



المدرسة الوطنية للعلوم التطبيقية - بني ملال
ⵜⴰⵎⴰⵔⵜ ⵜⴰⵎⴰⵏⴰⵢⵜ ⵜⴰⵖⴰⵏⴰⵏⵜ ⵜⴰⵙⴰⵢⵜ ⵜⴰⵎⴰⵏⴰⵢⵜ
Ecole Nationale des Sciences Appliquées - Béni Mellal

Module : Implementing Digital Projects
Filière : Transformation Digital Industriel
-TDI

**Étude de l'intégration et test
d'intégration d'un projet digital**

Réalisé par :	Encadré par :
Hacob Wissal	Professeur Mounaim Aquil

REMERCIEMENT

Nous tenons à remercier sincèrement Monsieur AQIL qui s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce rapport, ainsi pour l'inspiration, l'aide, la richesse, la qualité de son enseignement et surtout son expertise dans le domaine de l'implémentation des projets. On n'oublie pas nos parents pour leur contribution, leur soutien et leur patience. Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragés.

MERCI INFINIMENT

Table de matières

Introduction Générale :Contexte et annonce de la problématique :	4
Introduction du projet :	5
1.Objectif du projet et résultats visé.....	5
2.Méthodologie de recherche.....	5
Partie 1 : Généralités :	7
I. Introduction à l'intégration.....	8
1.Définition de l'intégration :	8
2.Les méthodes d'intégration :	8
• L'intégration continue :.....	8
Fonctionnement de l'intégration continue :.....	8
Les avantages de l'intégration continuent :	8
• L'intégration ascendante (Bottom-up) :.....	8
• L'intégration descendante (top-down) :.....	9
• L'intégration simultannée :.....	9
II. Les tests :.....	9
1.Définition des tests :.....	9
2.Importance des tests et impact sur la qualité.....	9
3.Les modes, les modalités d'exécution des tests et les méthodes de tests.....	10
Les modes :	10
Les modalités d'exécution des tests	10
Les méthodes de tests.....	10
III. Les tests d'intégration :	10
1.Définition	10
2.Outils d'intégration et de test populaires	11
• Jenkins	11
• Continuum apache	12
• TeamCity.....	12
3.benchmarking entre les différents outils :	14

<u>Partie 2: Partie Pratique</u>	16
I. L’outil d’intégration :Jenkins	17
• Conditions préalables :	17
• Étapes d’installation.....	17
II. Création de l'application : Java	17
Liaison avec Git et GitHub	24
III. Les tests d’intégration par Jenkins	27
<u>Conclusion Générale :</u>	34
Références :	35

Introduction générale

Contexte et annonce de la problématique :

Contexte : Dans le domaine du développement de logiciels, les applications web sont devenues de plus en plus courantes en raison de leur accessibilité et de leur portabilité. Cependant, les applications web modernes sont souvent complexes, avec de multiples composants et dépendances, ce qui rend leur intégration et leur déploiement difficiles. Pour surmonter ce défi, les équipes de développement utilisent des outils d'intégration pour automatiser le processus d'intégration et de déploiement.

Annonce problématique : Face à la complexité croissante des applications web modernes, l'intégration de leurs différents composants est devenue un défi majeur pour les équipes de développement. Comment automatiser efficacement l'intégration et le déploiement de ces composants pour accélérer la mise sur le marché de l'application tout en garantissant la qualité et la stabilité du système ? Comment s'assurer que les différentes parties de l'application fonctionnent ensemble de manière cohérente et fiable ? Ce sont des questions clés que nous allons explorer dans ce projet d'intégration des différents composants d'une application web. Nous chercherons à proposer des solutions efficaces pour faciliter l'intégration et le déploiement de ces composants, en utilisant des outils d'intégration continue tels que Jenkins, Continuum ou TeamCity

Introduction du projet

1. Objectif du projet et résultats visé

Objectif du projet : L'objectif de ce projet est de faciliter l'intégration des différents composants d'une application de gestion hospitalière. Cette application comporte de multiples fonctionnalités telles que la gestion des dossiers médicaux, la gestion des ressources humaines. Nous chercherons à automatiser le processus d'intégration et de déploiement de ces différents composants afin de garantir la qualité et la stabilité du système.

Résultats visés : Les résultats attendus de ce projet sont multiples.

- Améliorer l'efficacité du processus de développement en réduisant les temps d'intégration et de déploiement.
- Réduire les erreurs humaines en éliminant les risques d'erreur de configuration manuelle.
- Augmenter la qualité de l'application en assurant la cohérence et la fiabilité de l'ensemble du système.
- Réduire les coûts de maintenance de l'application à long terme en facilitant les mises à jour et les corrections de bugs.

2. Méthodologie de recherche

	Mode Tâche	Nom de la tâche	Début	Fin	Début au plus tard	Fin au plus tard	Marge libre	Marge totale
1		Phase1: Collection d'information	Mar 07/03/23	Mar 28/03/23	Mar 07/03/23	Mar 28/03/23	0 jour	0 jour
2		Tache1: Collection d'information generale (Integration Definition + Types + Outils)	Mar 07/03/23	Lun 13/03/23	Mer 19/04/23	Mar 25/04/23	31 jours	31 jours
3		Tache2 : Recherche approfondis (11 Methodes d'integration)	Mar 14/03/23	Lun 20/03/23	Mer 19/04/23	Mar 25/04/23	26 jours	26 jours
4		Tache3: Benchmark (Choix de 5 methodes pour les presenté)	Mar 21/03/23	Lun 27/03/23	Mer 19/04/23	Mar 25/04/23	21 jours	21 jours
5		Suivi1	Mar 28/03/23	Mar 28/03/23	Mar 25/04/23	Mar 25/04/23	20 jours	20 jours
6		Phase2: Pratique	Mer 29/03/23	Sam 08/04/23	Mer 29/03/23	Sam 08/04/23	0 jour	0 jour
7		Tache1 : Benchmark des methodes et des outils d'integrations (Adaptation du projet avec les recherches)	Mer 29/03/23	Mer 29/03/23	Mar 25/04/23	Mar 25/04/23	19 jours	19 jours
8		Tache2 : Installation d'outils Jenkins et tutorial d'utilisation	Mer 12/04/23	Lun 24/04/23	Jeu 13/04/23	Mar 25/04/23	1 jour	1 jour
9		Suivi2	Mar 25/04/23	Mar 25/04/23	Mar 25/04/23	Mar 25/04/23	0 jour	0 jour

Réunions d'équipe :

- Organisé des réunions d'équipe régulières pour discuter des approches de l'intégration continue, évaluer les progrès et les obstacles, et prendre des décisions en équipe pour avancer efficacement dans le projet.

Définition de l'intégration :

- Recherche approfondie pour comprendre les différentes définitions et concepts liés à l'intégration dans le développement de logiciels.
- Étudié les différentes approches d'intégration telles que l'intégration continue, l'intégration ascendante et descendante, l'intégration hybride et l'intégration simultanée.
- Examiné et étudié les avantages et les inconvénients des différentes méthodes d'intégration.

Outils d'intégration :

- Étude comparative des différents outils d'intégration continue disponibles sur le marché, tels que Jenkins, Continuum et TeamCity.
- Examiné les fonctionnalités et les avantages de chaque outil.

Benchmarking et choix des méthodes d'intégration et outils :

- Effectué un benchmarking pour évaluer les différentes méthodes d'intégration et outils disponibles sur le marché en comparant leurs avantages et inconvénients.
- Sélectionné les méthodes et outils qui correspondent le mieux à nos besoins.

Phase de réalisation :

- Installer Jenkins sur nos serveurs, intégré notre code source, configuré les tests d'intégration et effectué des tests pour évaluer l'efficacité de notre processus d'intégration continue.
- Effectué des ajustements et des améliorations en fonction des résultats obtenus pour garantir une intégration continue efficace et sans heurts.

Partie 1: **Généralités**

I. Introduction à l'intégration

1. Définition de l'intégration :

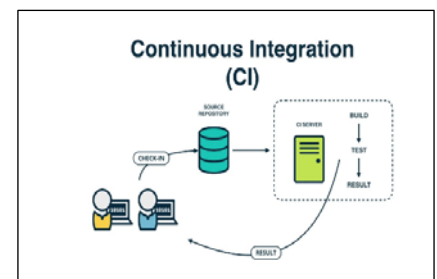
L'intégration informatique, ou intégration des systèmes d'information, désigne le fait de connecter les données, les applications, les API et les appareils au sein de votre service informatique, dans le but d'augmenter l'efficacité, la productivité et l'agilité de votre entreprise. L'intégration informatique permet à tous les éléments d'un environnement informatique de fonctionner ensemble. Ainsi, elle représente un aspect clé de la transformation d'une entreprise, c'est-à-dire de son adaptation face à l'évolution du marché.

Tests d'intégration : des tests effectués pour s'assurer que les modules logiciels s'intègrent correctement les uns avec les autres et qu'ils fonctionnent comme prévu.

2. Les méthodes d'intégration :

- L'intégration continue :

Consiste à tester fréquemment et automatiquement chaque modification de code avant de le déployer en production. Le but de l'intégration continue est de détecter rapidement les erreurs de code et de les corriger avant qu'elles ne deviennent des problèmes plus importants.



Fonctionnement de l'intégration continue :

Chaque modification du code est intégrée dans une branche principale de développement, qui est ensuite testée automatiquement. Si les tests réussissent, le code peut être déployé dans un environnement de production.

Les avantages de l'intégration continuent :

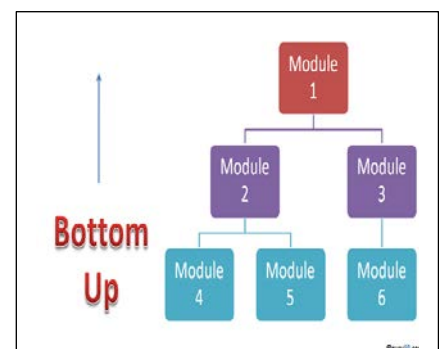
- ✓ Éviter les problèmes liés à la livraison du code
- ✓ Collaboration plus efficace entre les développeurs
- ✓ Augmentation de la qualité globale du code.

- L'intégration ascendante (Bottom-up) :

Consiste à intégrer d'abord les modules de plus bas niveau, tels que les composants logiciels les plus élémentaires, puis à intégrer progressivement des modules de plus haut niveau, jusqu'à l'intégration du système final.

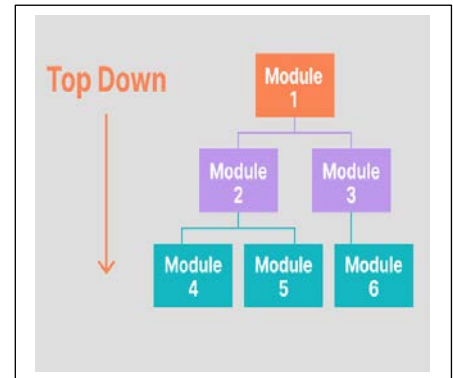
Fonctionnement de l'intégration ascendante :

Les modules logiciels les plus basiques sont intégrés en premier, afin de s'assurer que leur fonctionnement est correct et qu'ils ne créent pas d'interférences avec les autres modules. Les modules logiciels de niveau supérieur sont ensuite intégrés, en s'appuyant sur les modules de niveau inférieur qui ont déjà été testés.



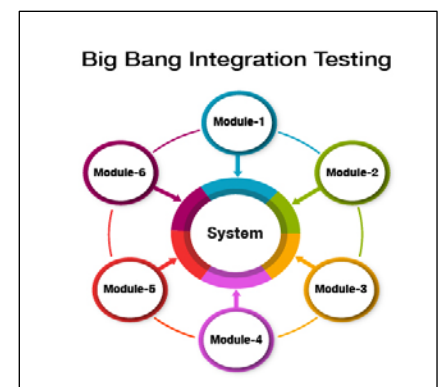
- **L'intégration descendante (top-down) :**

Consiste à intégrer d'abord les modules de plus haut niveau, tels que les interfaces utilisateur et les modules fonctionnels les plus importants, puis à intégrer progressivement des modules de plus bas niveau, jusqu'à l'intégration des composants logiciels les plus élémentaires. Cette méthode peut être efficace pour garantir que les interfaces Utilisateur et les fonctionnalités principales du système fonctionnent correctement dès le départ, mais elle peut prendre plus de temps pour détecter les erreurs dans les modules de plus bas niveau.



- **L'intégration simultanée (Big Bang):**

Les tests d'intégration simultanée, également appelés tests de type "Big Bang", font référence à une approche de test où tous les modules ou composants d'un projet digital sont intégrés simultanément, sans suivre une séquence spécifique. Dans cette approche, l'ensemble du système est intégré en une seule fois, et les tests sont effectués pour évaluer le bon fonctionnement global du système.



II. Les tests :

1. Définition des tests :

Il existe plusieurs définitions des tests, voici quelques définitions de tests à partir des références différentes.

IEEE (Institute of Electrical and Electronics Engineers) :

« Le test est l'exécution ou l'évaluation d'un système ou d'un composant, par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus. »

SWEBOK:

« Software testing consists of the dynamic verification that a program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite execution domain. »

“The art of software testing” de Glenford J. Myers, Corey Sandler, Tom Badgett:

« Un test réussi n'est pas un test qui n'a pas trouvé de défauts, mais un test qui a effectivement trouvé un défaut. »

On peut conclure que la définition des tests est l'étude d'un comportement pour un logiciel informatique sous l'ensemble des activités dynamique, afin de vérifier la conformité à son cahier de charge, et aussi aux attentes des parties prenantes.

2. Importance des tests et impact sur la qualité

Dans un cycle de développement d'un logiciel, les activités des tests représentent une partie très importante qu'il assure la présence, et validité d'un besoin ou un niveau de qualité défini par les parties prenantes. La qualité ce n'est pas les tests, mais les tests et d'autres processus de validation (Sondages, Audit annuel, etc.) peuvent prouver que la qualité est présente.

3. Les modes, les modalités d'exécution des tests et les méthodes de tests

Les modes :

- **Manuel** : ce sont les tests qu'ils s'effectuent par les testeurs, par l'entrée des données, voir les résultats selon les différents scénarios possibles qu'ils sont définis ou les effets attendus.
- **Automatique** : Dans ce mode, l'utilisation d'un outil ou plusieurs outils informatique est nécessaire pour exécuter ce mode de test, toujours avec l'intervention d'un testeur.

Les modalités d'exécution des tests

- **Dynamique** : il s'agit des tests de tous les composants et les caractéristiques d'un système.
- **Statique** : il s'agit des tests effectués par des testeurs càd de trouver des erreurs seulement en lisant le code.

Les méthodes de tests

Il existe deux méthodes de test : Boite blanche (structurel) ou Boite noire (fonctionnelle), la première méthode consiste à analyser le code source et la structure logicielle, et pour la deuxième méthode c'est plus tôt basé sur les spécifications logiciel (informelle et formelle), cette méthode permettra de bien écrire les tests avant la programmation.

III. Les tests d'intégration :

1. Définition

Les tests d'intégration sont une méthodologie de test de logiciels utilisée pour tester des composants logiciels individuels ou des unités de code pour vérifier l'interaction entre divers composants logiciels et détecter les défauts d'interface. Les composants sont testés en tant que groupe unique ou organisés de manière itérative.

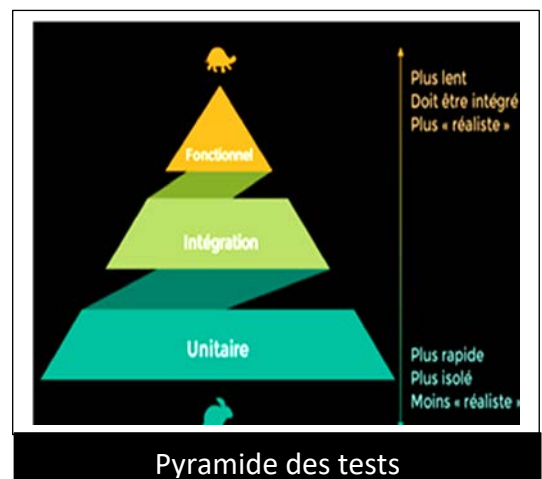
il existe deux types de tests d'intégration :

- **Les tests d'intégration composants :**

Permettent de vérifier si plusieurs unités de code fonctionnent bien ensemble, dans un environnement de test assez proche du test unitaire, c'est-à-dire de manière isolée, sans lien avec des composants extérieurs et ne permettant pas le démarrage d'une vraie application.

- **Les tests d'intégration système :**

Permettent de vérifier le fonctionnement de plusieurs unités de code au sein d'une configuration d'application, avec éventuellement des liens avec des composants extérieurs comme une base de données, des fichiers, ou des API en réseau.



2.Outils d'intégration et de test populaires

IV. Jenkins



Jenkins est un serveur d'automatisation open source avec un plugin inégalé écosystème pour prendre en charge pratiquement tous les outils dans le cadre de votre prestation Pipelines.

	<i>Caractéristiques</i>	<i>Fiabilité</i>	<i>Longévité</i>	<i>Environnement scibles</i>	<i>Facilité d'utilisation</i>
Jenkins	L'outil support: Ant, Maven1, Maven2, et Shell Unix ou Batch Windows.	Créé par le fondateur d'Hudson en 2011.	Support communautaire très active, et une large banque de plugins offerts par Jenkins. Plugins de « pipelines » peut survivre des pannes d'infrastructure.	Linux, Mac OS X, Solaris, et Win32.	Configuration très simple, gestionnaire d'intégration très facile .

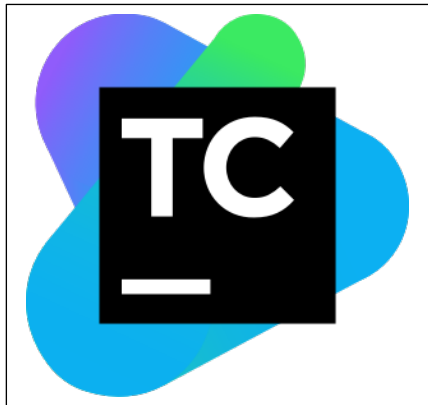
- Continuum apache



Continuum est développé par Apache, il offre une gestion de distribution du build, et la possibilité de regrouper les unités de développement au sein d'un groupe mutualisé.

<i>Caractéristiques</i>	<i>Caractéristiques</i>	<i>Fiabilité</i>	<i>Longévité</i>	<i>Environnements cibles</i>	<i>Facilité d'utilisation</i>
Continuum	L'outil support : Ant, Maven1, Maven2, et Shell.	Sorti en 2005. Donc plus de 10 ans de services donc on va voir Continuum plus en plus à travers Apache	Support communautaire très active. et une très bonne documentation à travers Apache Maven	Linux, Mac OS X, Solaris, et Win32.	Facile à utiliser et à intégrer.

- TeamCity



TeamCity est un outil d'intégration continue basé sur l'environnement java, présenté par JetBrains, et c'est une solution commerciale, mais on peut avoir une « Freemium license » pour 20 « Build configuration » et 3 « Build agent », par contre si nous sommes en train de développer un projet open source on peut dans ce cas demander une licence gratuitement.

Des entreprises qui utilisent TeamCity :



	<i>Caractéristiques</i>	<i>Fiabilité</i>	<i>Longévité</i>	<i>Environnements cibles</i>	<i>Facilité d'utilisation</i>
TeamCity	L'outil support: Cloud intégration (Amazon EC2, Microsoft Azure...). Il donne des résultats avant même que l'intégration finisse.	October 2, 2006 [10]	JetBrains est une entreprise très active, et il travaille sur tous les nouvelles technologies, cela garantira une bonne continuité.	Java, .NET et Ruby. IDEs: Eclipse, IntelliJ IDEA, Visual Studio	Facile à mettre en œuvre avec un peu d'expérience, très graphique et une interface très sympathique pour mettre un processus de build de base.


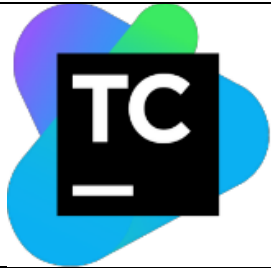

Après avoir examiné les fonctionnalités et les avantages de Jenkins, Continuum et TeamCity, il est clair que Jenkins est un choix solide pour les équipes d'intégration continue. Avec sa grande communauté de développeurs et ses innombrables plugins, Jenkins offre une grande flexibilité et une personnalisation facile. Bien que Continuum et TeamCity offrent également des fonctionnalités impressionnantes, Jenkins est plus largement utilisé et a une longue histoire de succès dans la communauté de développement de logiciels. En fin de compte, le choix entre ces outils dépendra des besoins spécifiques de chaque équipe, mais Jenkins reste une option populaire et fiable pour l'intégration continue.

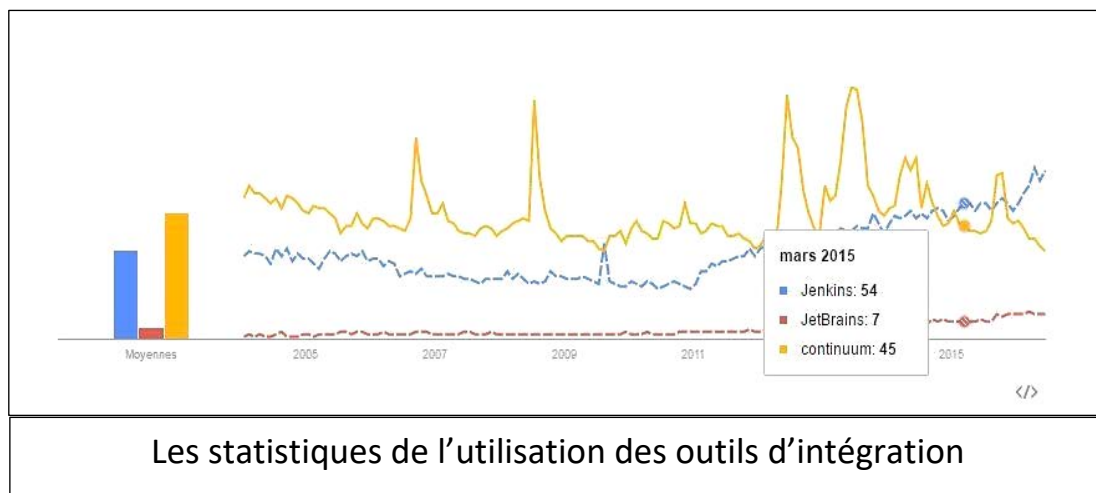
3.benchmarking entre les différents outils :

Les Critères du Benchmark :

- **Temps d'intégration** : Le temps qu'il faut pour intégrer une modification de code dans le projet.
- **Fiabilité** : La capacité de la méthode d'intégration à garantir que les modifications de code sont intégrées sans erreurs et sans conflits.
- **Coût** : Les coûts associés à l'implémentation et à la maintenance de la méthode d'intégration, tels que les coûts de matériel et de logiciel, les coûts de personnel et les coûts de formation.
- **Complexité** : La complexité de la mise en œuvre de la méthode d'intégration et la difficulté pour les développeurs à s'y conformer.
- **Flexibilité** : La capacité de la méthode d'intégration à s'adapter aux besoins changeants du projet et de l'entreprise.
- **Efficacité** : La capacité de la méthode d'intégration à détecter rapidement les conflits de code et les erreurs de code.
- **Type d'application** : Le type d'application, comme une application web ou mobile, peut avoir une incidence sur les méthodes d'intégration utilisées.
- **Framework** : Le Framework utilisé pour construire l'application peut avoir un impact sur les méthodes d'intégration.
- **Langage de programmation** : Le langage de programmation utilisé pour construire l'application peut avoir un impact sur les méthodes d'intégration. Certains langages de programmation peuvent avoir des bibliothèques ou des outils spécifiques pour l'intégration.

Critères /Méthodes d'intégration	Continue	Ascendante	Descendante	Simultanée
Temps d'intégration minimale	✓	✓	✓	
Fiabilité	✓			
Coût	✓	✓	✓	✓
Flexibilité	✓			
Efficacité	✓	✓	✓	✓
Complexité de l'application				✓

USAGE\OUTILS	 Jenkins		
Gratuit et Open Source	✓	✓	✓
Utilisé mondialement	✓		
Documentation riche	✓		
Facile à utiliser et intégrer	✓	✓	
Multiples tests sur différentes branches	✓		
Validation individuelle	✓		
Global	✓	✓	✓
Socle technique, et l'extensibilité	✓	✓	



Partie 2 :
Partie pratique

I. L'outil d'intégration :

L'outil choisi : Jenkins

Jenkins est souvent choisi pour les tests d'intégration en raison de sa flexibilité, de sa facilité d'utilisation, de sa capacité d'automatisation, de sa large gamme de plugins et de son support continu par une communauté active. Ces avantages font de Jenkins un choix solide pour améliorer les processus de tests d'intégration.



L'installation sur Windows :

- **Conditions préalables :**

Configuration matérielle minimale :

- 256 Mo de RAM

- 1 Go d'espace disque (bien que 10 Go soient un minimum recommandé si vous exécutez Jenkins en tant que conteneur Docker)

Configuration matérielle recommandée pour une petite équipe :

- 4 Go + de RAM

- 50 Go+ d'espace disque

- **Étapes d'installation à l'aide du programme d'installation Windows MSI**

Reportez-vous à la section Windows de la page Téléchargement de Jenkins pour télécharger une version LTS ou une version hebdomadaire du programme d'installation de Windows. Une fois le téléchargement terminé, ouvrez le programme d'installation de Windows et suivez les étapes ci-dessous pour installer Jenkins.

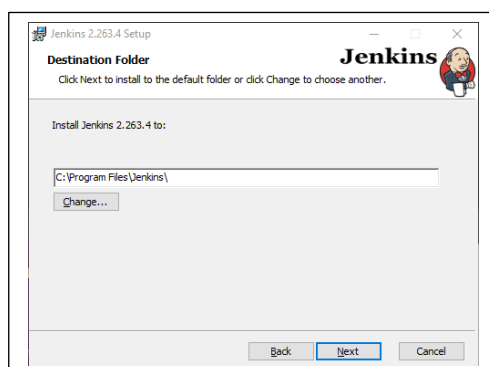
Étape 1 : Assistant de configuration

Lors de l'ouverture de Windows Installer, un assistant de configuration de l'installation apparaît, cliquez sur Suivant dans l'assistant de configuration pour démarrer votre installation.



Étape 2 : Sélectionnez le dossier de destination

Sélectionnez le dossier de destination pour stocker votre installation Jenkins et cliquez sur Suivant pour continuer.



Étape 3 : Identifiants de connexion au service

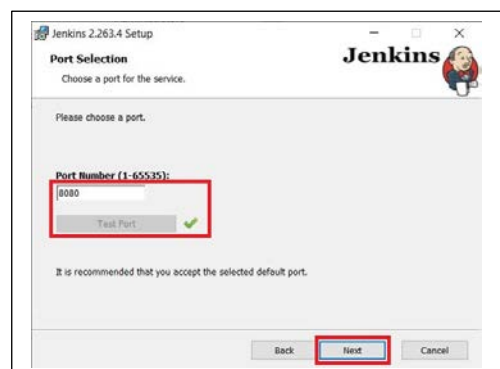
Lors de l'installation de Jenkins, il est recommandé d'installer et d'exécuter Jenkins en tant que service Windows indépendant à l'aide d'un utilisateur local ou de domaine car il est beaucoup plus sûr que d'exécuter Jenkins à l'aide de LocalSystem (équivalent Windows de root) qui accordera à Jenkins un accès complet à votre machine et services.

Pour exécuter le service Jenkins à l'aide d'un utilisateur local ou de domaine, spécifiez le nom d'utilisateur et le mot de passe du domaine avec lequel vous souhaitez exécuter Jenkins, cliquez sur Test Credentials pour tester vos informations d'identification de domaine et cliquez sur Next.



Étape 4 : sélection du port

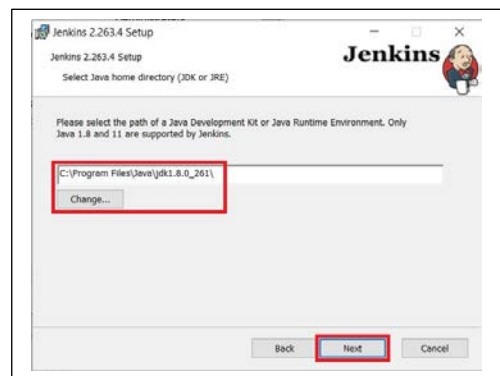
Spécifiez le port sur lequel Jenkins sera exécuté, bouton Test Port pour valider si le port spécifié est libre sur votre machine ou non. Par conséquent, si le port est libre, il affichera une coche verte comme indiqué ci-dessous, puis cliquez sur Suivant.



Étape 5 : Sélectionnez le répertoire d'accueil Java

Le processus d'installation recherche Java sur votre machine et pré-remplit la boîte de dialogue avec le répertoire de base Java. Si la version Java nécessaire n'est pas installée sur votre machine, vous serez invité à l'installer.

Une fois votre répertoire d'accueil Java sélectionné, cliquez sur Suivant pour continuer.



Étape 6 : Configuration personnalisée

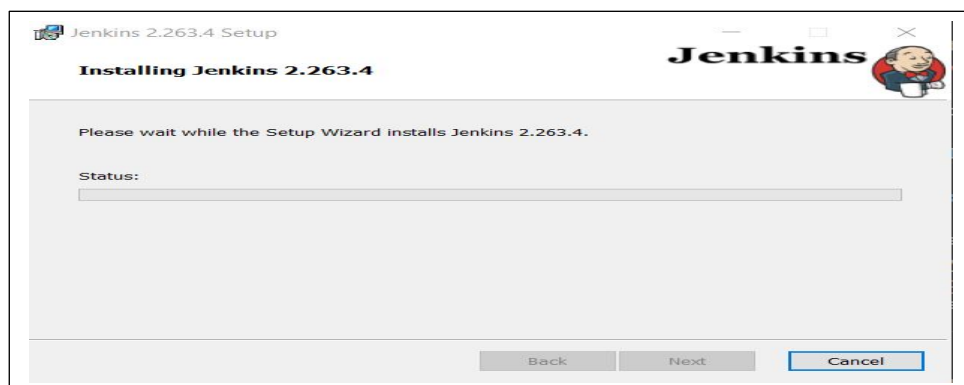
Sélectionnez les autres services qui doivent être installés avec Jenkins et cliquez sur Suivant.



Étape 7 : Installer Jenkins

Cliquez sur le bouton Installer pour lancer l'installation de Jenkins.

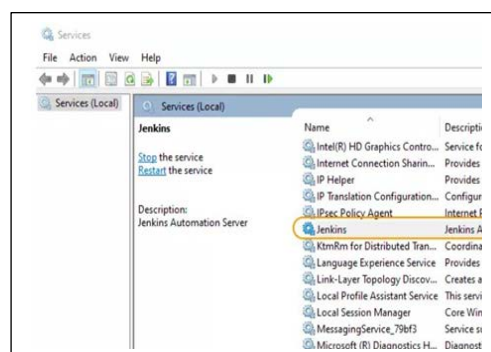
De plus, cliquer sur le bouton Installer affichera la barre de progression de l'installation, comme indiqué ci-dessous :



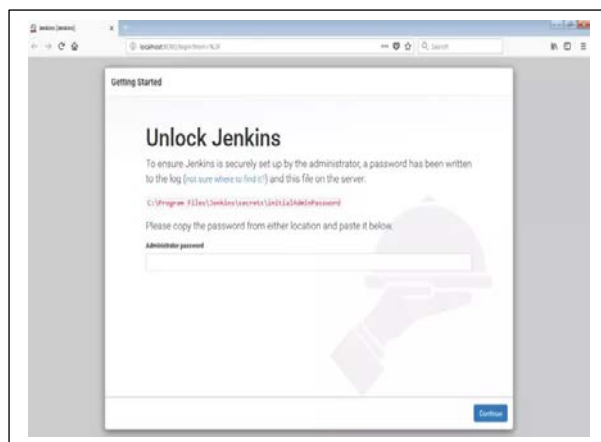
Étape 8 : terminer l'installation de Jenkins

Une fois l'installation terminée, cliquez sur Terminer pour terminer l'installation.

Jenkins sera installé en tant que service Windows. Vous pouvez le valider en parcourant la section des services, comme indiqué ci-dessous :



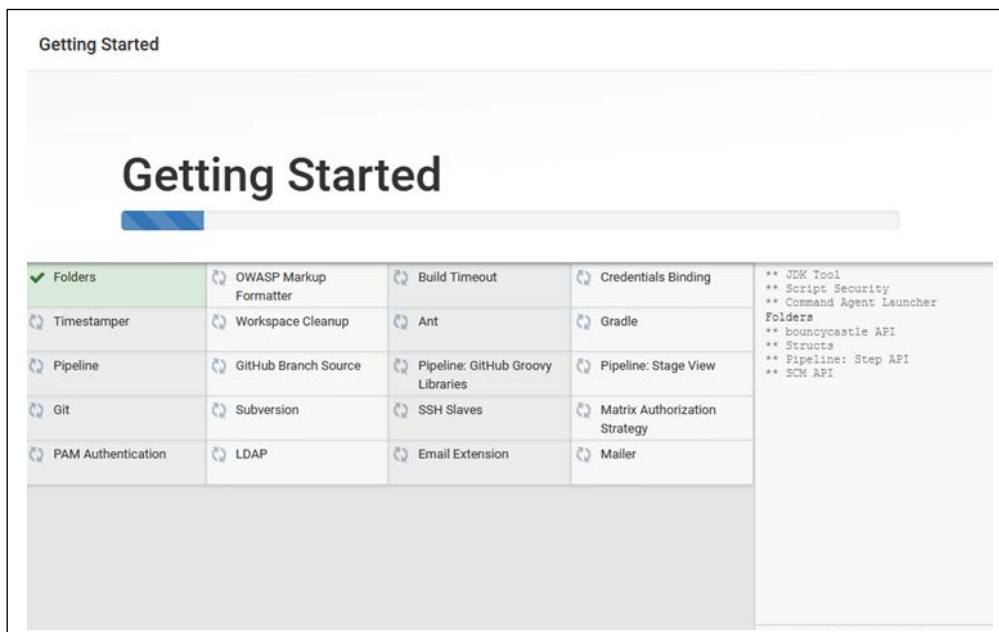
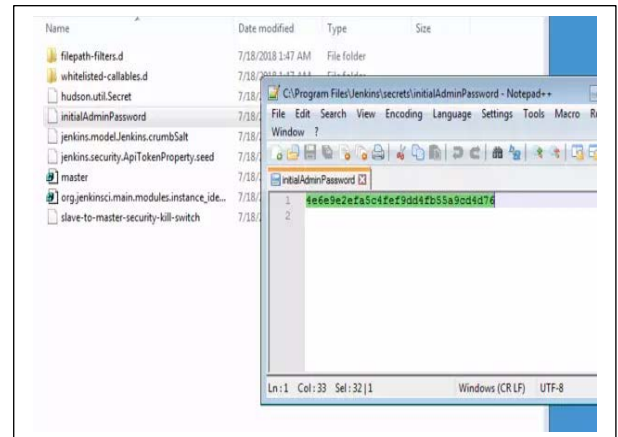
L'installation ne devrait prendre que quelques secondes. Immédiatement après, le programme ouvre localhost :8080 dans votre navigateur par défaut. Cela donne accès à une interface Web, que vous pouvez utiliser pour configurer Jenkins. Mais vous devez d'abord prendre une mesure de sécurité : Jenkins a généré un mot de passe aléatoire pour vous. Celui-ci se trouve dans le répertoire Jenkins, dans le dossier Secrets et enfin dans le fichier initialAdminPassword. Ce fichier peut être ouvert avec n'importe quel éditeur de texte. Copiez la chaîne de caractères et collez-la dans la zone ad hoc de l'interface Web.



Avant de pouvoir commencer à utiliser Jenkins, l'application vous invite à entrer un code généré dans le masque.

Le fichier de mot de passe peut être ouvert avec n'importe quel éditeur.

Maintenant, il est temps de configurer Jenkins. L'assistant de configuration vous demandera si vous souhaitez choisir les plugins à installer ou si vous préférez utiliser le paramétrage par défaut intégrant déjà toutes les améliorations majeures. Si vous utilisez Jenkins pour la première fois, ce choix par défaut vous conviendra certainement. Et ne vous inquiétez pas : des plugins supplémentaires peuvent être facilement installés plus tard.

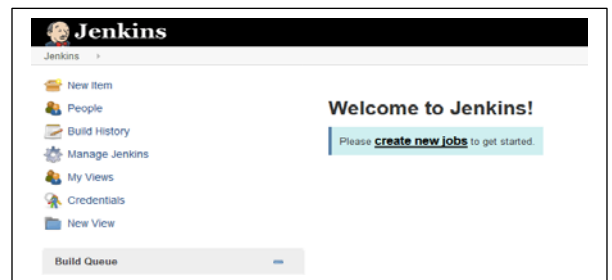


La sélection par défaut de plugins inclut les améliorations les plus importantes.

Créez ensuite un premier utilisateur. Si vous travaillez seul sur votre projet, vous pouvez ignorer cette étape et utiliser simplement Jenkins en tant qu'administrateur. Dans les paramètres de Jenkins, vous pourrez toujours créer ultérieurement de nouveaux utilisateurs avec des droits différents. Lors de la dernière étape de la configuration, vous avez encore la possibilité de spécifier une URL Jenkins. Le champ correspondant comporte par défaut localhost :8080. Si vous avez installé Jenkins sur un serveur (ce qui devrait être le cas dans un environnement de travail professionnel), indiquez ici le bon chemin d'accès au répertoire Jenkins. Enregistrez les données saisies et terminez la configuration.

Presentation de Jenkins :

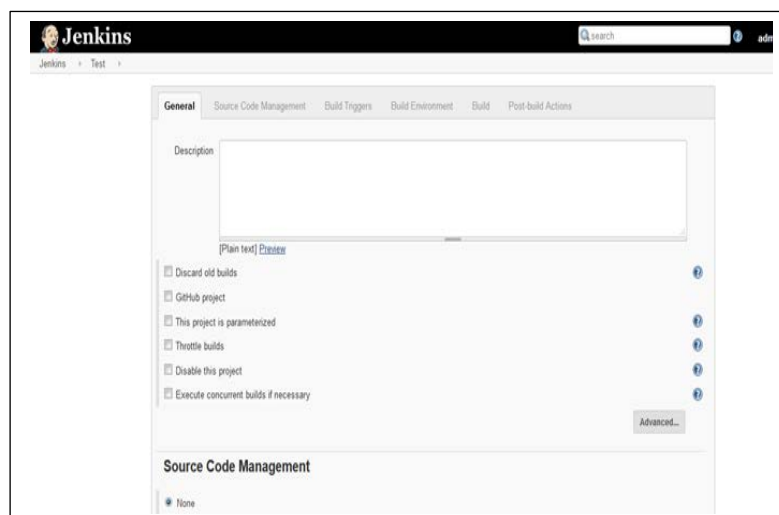
Vous démarrez Jenkins avec un environnement de travail complètement vide. Pour démarrer un nouveau projet d'intégration continue, vous devez créer un nouveau job. Pour ce faire, utilisez l'icône visible au milieu de la fenêtre (« create new jobs ») ou l'option de menu « New Item », que vous trouverez à gauche.



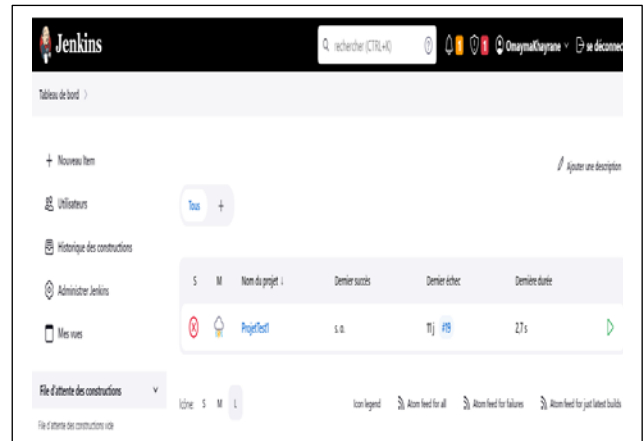
Ensuite, il est nécessaire de donner un nom à votre projet et de choisir ce que vous voulez atteindre :

- Freestyle project : Jenkins associe la gestion des versions à un système de build.
- Pipeline : créez un pipeline entre plusieurs agents de build.
- Projet multi-configuration : Choisissez cette option si vous avez un projet qui nécessite une variété de paramètres, par exemple parce que vous utilisez plusieurs environnements de test.
- Folder : un dossier est un conteneur dans lequel vous pouvez stocker des objets imbriqués.
- GitHub Organization : cette option fait une recherche dans tous les référentiels d'un compte sur GitHub.
- Multibranch Pipeline : vous permet de créer directement plusieurs pipelines.

À la page suivante, vous avez de nombreuses options de paramétrage. Et de configuration du projet.



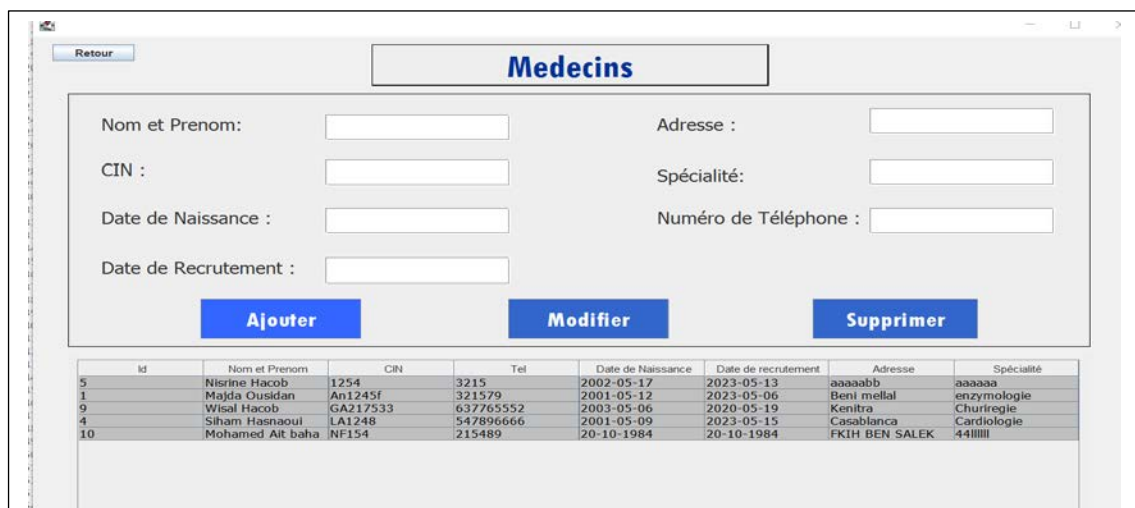
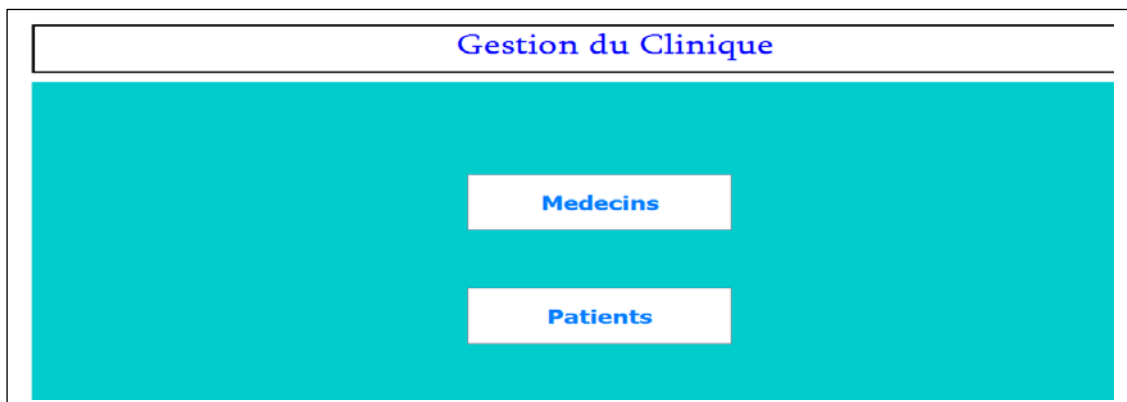
Dans le tableau de bord de Jenkins, vous pouvez voir tous les projets sur lesquels vous travaillez. Ici aussi, le programme matérialise l'état du projet par une couleur. Vous obtenez également des informations sur la Stabilité du build sous la forme d'un bulletin météo. Il s'agit d'une statistique sur la stabilité moyenne des builds du projet. Si plus de 80 % de vos builds réussissent, vous verrez un soleil. En dessous de cette valeur, la météo symbolique se dégrade.



II. Création de l'application :

Le langage choisi : Java(JavaSwing)

Java et Java Swing ont été choisis pour créer l'application en raison de la polyvalence de Java et de la richesse de Java Swing pour créer une interface utilisateur conviviale. Ensemble, ils offrent la portabilité du code, des composants graphiques prêts à l'emploi et une flexibilité de personnalisation. De plus, ils bénéficient d'une grande communauté de développeurs pour le support et les ressources en ligne.



id	Nom et Prenom	CIN	Tel	Date de Naissance	Maladie	Medecin Traitant
9	nada salid	LA154	125488	2023-05-16	insomnie	LA1248
11	lkz	rffrr	254	fr	1254	
13	nada salid	LA154	125488	2023-05-16	insomnie	LA1248

L'application développée en utilisant Java Swing est une solution de gestion pour une clinique. Elle comprend trois principales classes : Home, Médecins et Patient.

La classe Home est la classe principale de l'application. Elle offre aux utilisateurs la possibilité de choisir entre deux rôles : médecin ou patient. En fonction de leur choix, ils seront redirigés vers l'interface correspondante.

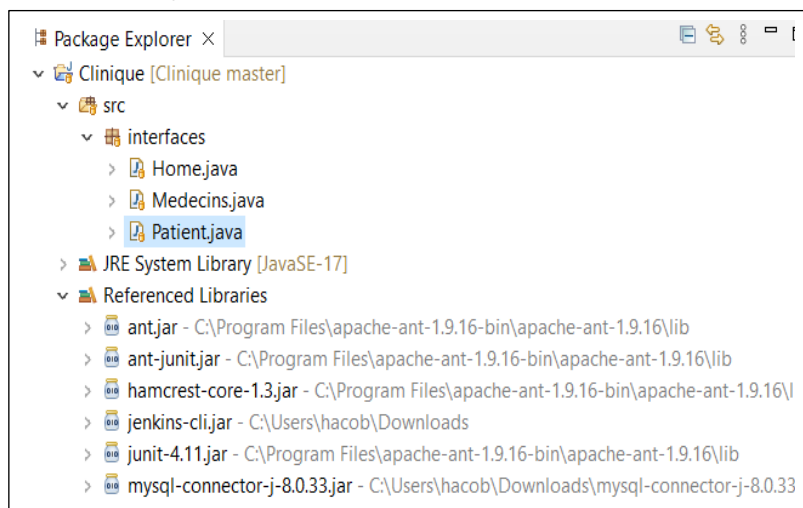
La classe Médecins représente les fonctionnalités destinées aux administrateurs de la clinique. Elle leur permet d'ajouter, de modifier et de supprimer les données des médecins. Les informations concernant chaque médecin incluent le nom, l'adresse, le numéro de téléphone, etc.

La classe Patient offre des fonctionnalités similaires à celles de la classe Médecin, mais spécifiquement pour les patients. Les administrateurs peuvent ajouter, modifier et supprimer les informations des patients, telles que le nom, l'adresse, le numéro de téléphone, etc.

Une fonctionnalité importante de l'application est la liaison entre les classes Médecin et Patient. Dans les données d'un patient, il y a une liste déroulante permettant de choisir le médecin traitant en utilisant le numéro d'identification national (CIN) du médecin. Cela permet de garder une trace du médecin responsable du suivi et du traitement de chaque patient.

Grâce à cette application, les administrateurs de la clinique peuvent gérer efficacement les données des médecins et des patients, en ajoutant, en modifiant ou en supprimant les informations nécessaires. La liaison entre les médecins et les patients facilite la coordination et le suivi médical au sein de la clinique.

l'insertion des fichiers .jar nécessaires :



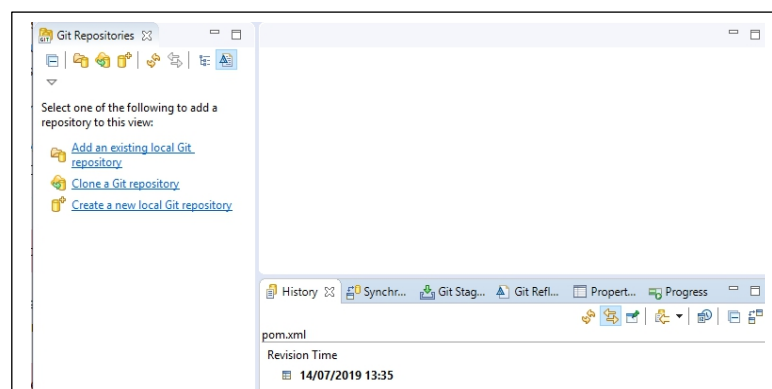
- ant.jar : C'est le fichier JAR principal d'Apache Ant, un outil de construction de logiciels. Il contient les bibliothèques nécessaires pour exécuter et configurer des tâches de construction automatisées. Ant fournit une approche basée sur XML pour la compilation, le déploiement, les tests et d'autres opérations liées au développement de logiciels.
- ant-junit.jar : Ce fichier JAR est utilisé en conjonction avec Apache Ant pour exécuter des tests unitaires écrits avec JUnit. Il contient les classes et les dépendances nécessaires pour exécuter les tests JUnit lors de la construction du projet avec Ant.
- hamcrest-core.jar : Hamcrest est un framework de correspondance d'objets utilisé dans les tests unitaires. Le fichier JAR hamcrest-core.jar contient les classes centrales de Hamcrest, qui permettent de définir des assertions et des correspondances plus expressives lors de l'écriture de tests avec des frameworks tels que JUnit.
- jenkins-cli.jar : C'est le fichier JAR pour l'interface en ligne de commande (CLI) de Jenkins, un serveur d'intégration continue. Il permet d'interagir avec Jenkins à partir de la ligne de commande, ce qui facilite l'automatisation de certaines tâches administratives telles que la création de projets, le lancement de builds, etc.
- junit.jar : Ce fichier JAR contient les classes et les dépendances nécessaires pour exécuter des tests unitaires avec JUnit, l'un des frameworks de test les plus populaires pour Java. JUnit fournit des annotations, des assertions et des fonctionnalités pour l'écriture et l'exécution de tests unitaires de manière structurée.
- mysql-connector.jar : C'est le fichier JAR pour le pilote JDBC (Java Database Connectivity) de MySQL. Il permet à une application Java de se connecter à une base de données MySQL et d'interagir avec elle en utilisant des requêtes SQL. Le fichier JAR contient les classes nécessaires pour établir une connexion, exécuter des requêtes et récupérer des résultats à partir de la base de données.

Git et github :

Comment utiliser Git dans Eclipse ?

Git est un système de versionning de code, c'est-à-dire un logiciel qui va garder l'historique des modifications d'un code source. Cet historique est riche, car il contient des commentaires, les auteurs des modifications, et souvent plusieurs historiques parallèles (branches) lorsque plusieurs versions d'une même application sont développées en même temps.

En allant dans le menu **Window -> Perspective -> Open Perspective -> Other...** on peut voir l'item Git.. Le package explorer est remplacé par un explorateur de dépôts Git . maintenant on arrive à la fenêtre ci-dessous.



En cliquant sur Create a new local Git repository on peut créer un répertoire locale.

Comment utiliser GitHub avec git ?

GitHub est un dépôt public Git, accessible sur Internet. Tout le monde peut créer son projet gratuitement, pourvu qu'il soit open source et accessible aux autres. Sur GitHub, on peut trouver du code utilisé par Facebook, Red Hat, et d'autres entreprises bien connues. Néanmoins, tous les projets GitHub ne sont pas nécessairement open source ni même publics. Certains projets sont privés ou accessibles à certaines personnes seulement. GitHub propose alors une tarification en fonction du nombre de projets et d'utilisateurs.

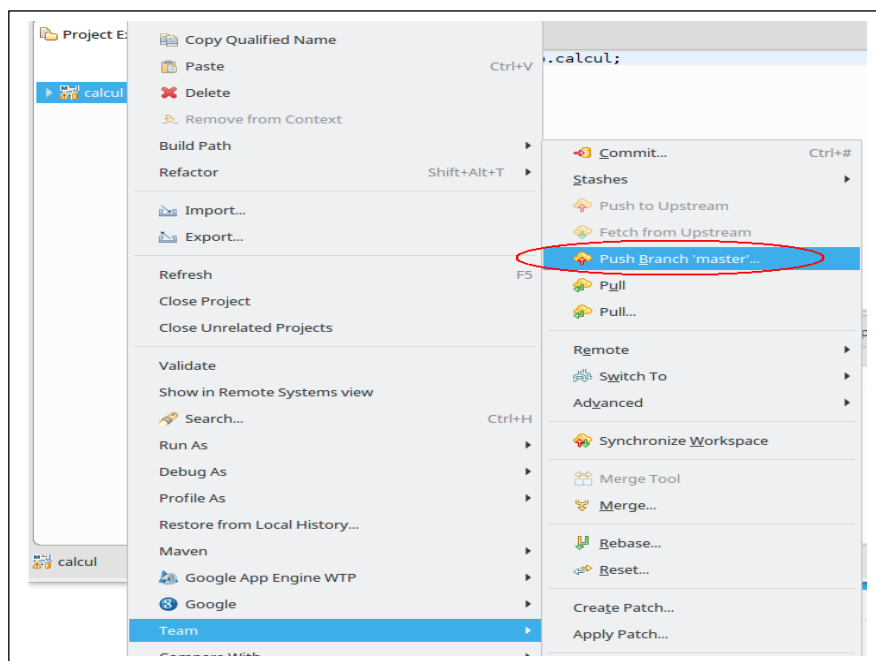
la différence entre Git et GitHub :

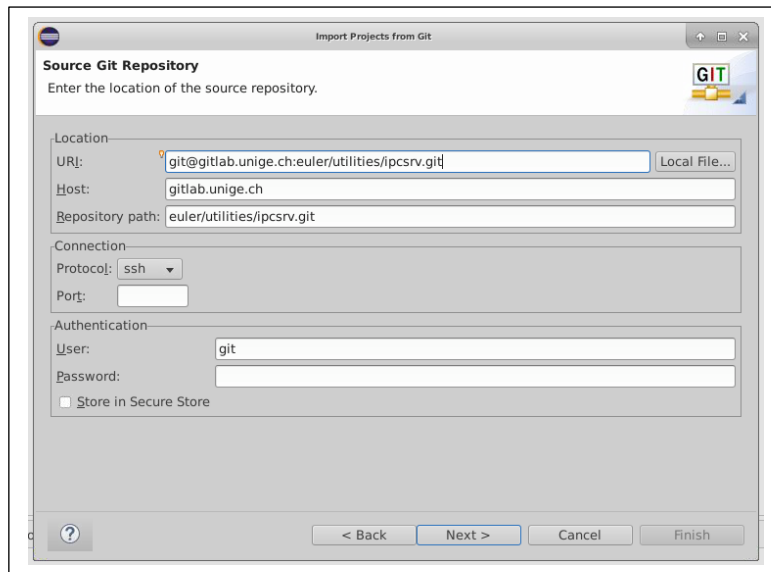
Git est un logiciel pour créer des **versions de code**. Il peut être utilisé directement sur votre ordinateur pour stocker l'historique du code. Mais tout l'intérêt réside dans le partage du code entre développeurs. Il faut donc disposer d'une ou plusieurs machines qui vont héberger de l'espace compatible avec Git, un serveur compatible avec Git ;

GitHub est le plus grand **serveur compatible Git**, accessible de manière publique sur Internet. Les entreprises disposent aussi souvent d'un serveur privé, hébergé parfois dans l'entreprise, pour partager le code entre développeurs.

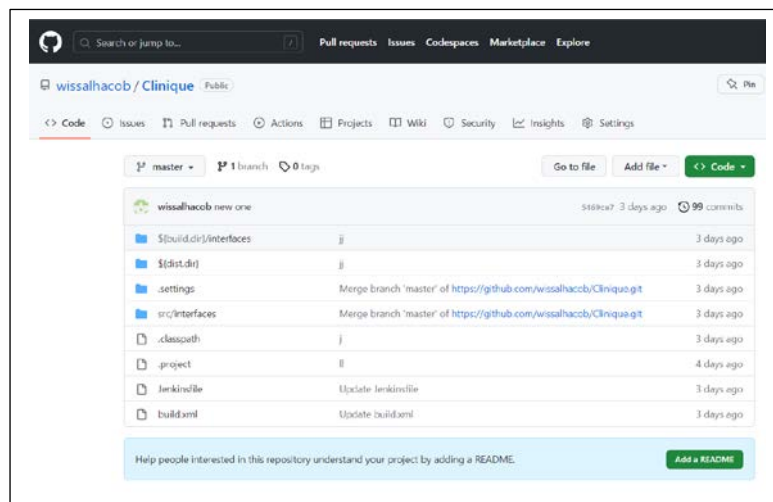
Jusqu'à présent les « **commit** » effectués, l'ont été dans le dépôt local sur la machine du développeur. Pour partager ce projet dans GitHub, il convient de le publier dans le dépôt distant du serveur. Pour le faire avec Git sous eclipse il faut sélectionner le projet dans l'explorateur et ensuite utiliser le menu contextuel (Click avec le bouton droit de la souris).

Utiliser le menu: **Team / Push Branch master**.





En ajoute l'url du repository créé dans GitHub et automatiquement le host et le repository s'ajoutent et on saisit le nom d'utilisation et le mot de passe de notre compte GitHub. On utilise le bouton « **Finish** » pour finaliser la publication. À ce stade le dernier commit du projet est publié sur le serveur.



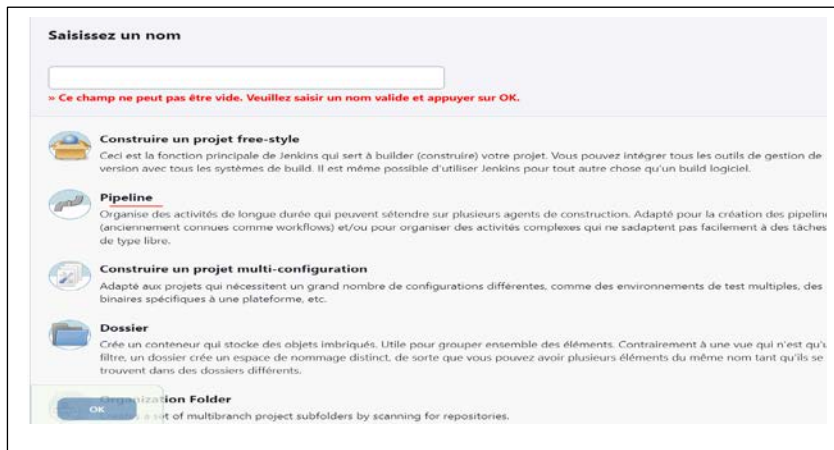
Le lien de notre repository : <https://github.com/wissalhacab/test-integration>

III. Les tests d'intégration continues :

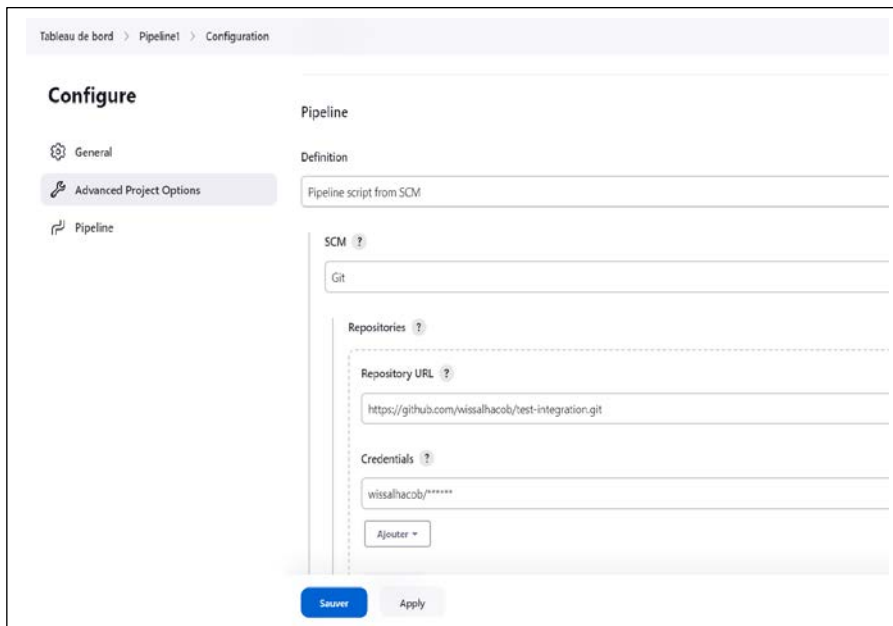
Article Jenkins :

On crée un article Jenkins de type pipeline :

Un pipeline Jenkins est représenté par un ensemble de scripts ou de fichiers de configuration qui décrivent les étapes et les actions à effectuer pour construire, tester et déployer une application. Il suit généralement une approche basée sur des fichiers de configuration, tels que Jenkinsfile, qui sont enregistrés dans le contrôle de version du projet.



On ajoute l'url de notre GitHub repository dans la configuration de l'article :



Dans la configuration de Jenkins en ajoute les installations nécessaires

Tableau de bord > Administrer Jenkins > Tools

Installations JDK

Installations JDK ^ Edited

Ajouter JDK

JDK Nom
jdk-11

JAVA_HOME
C:\Program Files\Java\jdk-11

☐ Install automatically ?

Ajouter JDK

Enregistrer Appliquer

Tableau de bord > Administrer Jenkins > Tools

Git installations

Name
Default

Path to Git executable ?
git.exe

☐ Install automatically ?

Add Git

Tableau de bord > Administrer Jenkins > Tools

Installations Ant

Installations Ant ^ Edited

Ajouter Ant

Ant Nom
ant

ANT_HOME
C:\Program Files\apache-ant-1.9.16-bin\apache-ant-1.9.16

☐ Install automatically ?

Ajouter Ant

Générer un fichier build.xml du projet

Le fichier "build.xml" est utilisé pour décrire les étapes de construction d'un projet Java, y compris les tests d'intégration. Il fournit une structure et des instructions pour automatiser le processus de construction, permettant ainsi de compiler le code, d'exécuter les tests et de générer les artefacts finaux du projet.

Ajouter :

```
<property name="classpath" value="${build.dir}:${lib.dir}/*;C:\Programmes  
\apache-ant-1.9.16-bin\apache-ant-1.9.16\lib\junit-4.13.2-javadoc.jar"/>
```

`${build.dir}` : Cette variable fait référence au répertoire de construction (build directory). Il est utilisé pour inclure les fichiers compilés ou générés lors de la construction du projet dans le classpath.

`${lib.dir}/*` : Cette expression fait référence à un répertoire appelé "lib.dir" contenant des fichiers JAR. L'étoile "*" est utilisée pour inclure tous les fichiers présents dans ce répertoire dans le classpath. Il est courant d'inclure les dépendances externes d'un projet dans un répertoire "lib" et de les ajouter au classpath de cette manière.

`C:\Programmes\apache-ant-1.9.16-bin\apache-ant-1.9.16\lib\junit-4.13.2-javadoc.jar` : Il s'agit d'un chemin absolu vers le fichier JAR "junit-4.13.2-javadoc.jar". Ce fichier JAR est ajouté au classpath pour inclure la documentation de JUnit lors de l'exécution des tests. Cela permet d'accéder à la documentation Javadoc associée à JUnit lors de l'exécution des tests.

la propriété "**classpath**" définit les éléments à inclure dans le classpath, y compris les fichiers générés lors de la construction, les dépendances du projet et un fichier JAR de documentation spécifique (junit-4.13.2-javadoc.jar).

```
<target name="test">
  <property name="classpath" value="${build.dir}:${lib.dir}/*"/>
  <property name="test.dir" value="C:\Users\hacob\Desktop\eclipse\Clinique/test"/>

  <mkdir dir="${build.dir}/reports"/>

  <junit printsummary="yes" haltonfailure="yes">
    <classpath>
      <path location="${classpath}"/>
      <pathelement location="${test.dir}"/>
    </classpath>
    <formatter type="xml"/>
    <batchtest fork="yes" todir="${build.dir}/reports">
      <fileset dir="${test.dir}">
        <include name="**/*Test.java"/>
      </fileset>
    </batchtest>
  </junit>
</target>
```

La propriété "classpath" est définie avec la valeur du répertoire de construction (build.dir) et du répertoire des bibliothèques (lib.dir) concaténés avec tous les fichiers présents dans le répertoire spécifié par la propriété "test.dir".

La propriété "test.dir" est définie avec le chemin du répertoire contenant les fichiers de test d'intégration. Dans cet exemple, le chemin est "C:\Users\hacob\Desktop\eclipse\Clinique/test".

Un répertoire "reports" est créé dans le répertoire de construction (build.dir). Ce répertoire sera utilisé pour stocker les rapports de test.

La tâche "junit" est utilisée pour exécuter les tests d'intégration. Les éléments suivants sont configurés dans la tâche :

Le chemin de classe (classpath) est spécifié en utilisant la propriété "\${classpath}" qui a été définie précédemment.

Un élément "formatter" est ajouté pour spécifier que le rapport de test doit être généré au format XML.

La tâche "batchtest" est utilisée pour définir les fichiers de test à exécuter. Elle utilise un élément "fileset" pour spécifier le répertoire "\${test.dir}" et inclut tous les fichiers correspondant au motif "**/*Test.java". Cela signifie que tous les fichiers de test se terminant par "Test.java" dans le répertoire "\${test.dir}" seront exécutés.

Les résultats des tests sont enregistrés dans le répertoire "\${build.dir}/reports" à l'aide de l'attribut "todir" de la tâche "batchtest".

Ce code exécute les tests d'intégration en utilisant JUnit comme framework de test. Les résultats des tests sont stockés dans le répertoire de construction et formatés en XML.

Créer un fichier Jenkinsfile dans le chemin du projet

```
pipeline {
    agent any

    environment {
        JAVA_HOME = tool 'jdk-11' // Spécifiez ici la version de Java que vous
        utilisez pour votre application }

    stages {
        stage('---clean---') {
            steps {
                bat "ant clean" // Pour nettoyer votre projet Java Swing    } }

        stage('--compile--') {
            steps {
                bat "ant compile" // Pour compiler votre projet Java Swing    } }
    }
}
```

```

stage('--Test--') {
    steps {
        bat 'ant test' }}
stage('Package') {
    steps {
        bat 'ant package' }}
stage('Build') {
    steps {
        bat 'ant build' // ou 'ant jar' }}
stage('Run') {
    steps {
        bat 'ant run' }
    }
}

```

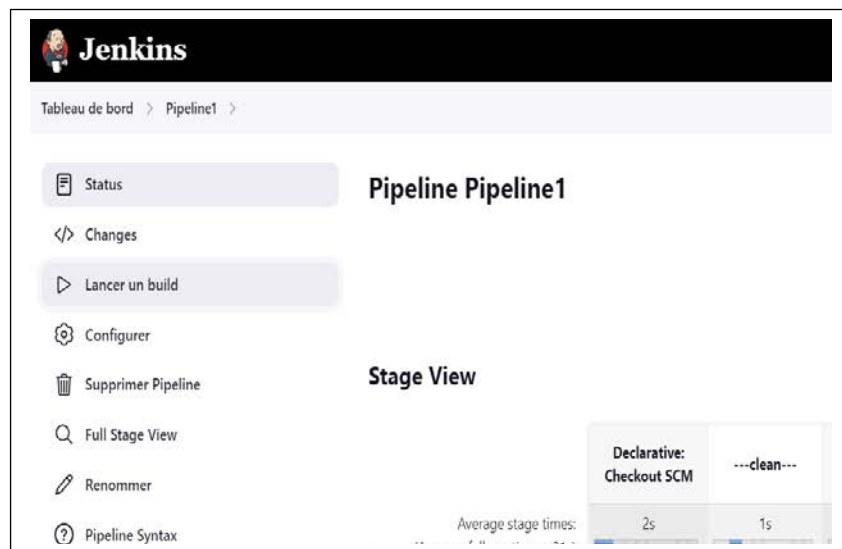
- **"clean"** : Cette étape utilise la commande "ant clean" pour nettoyer votre projet Java Swing. Cela peut inclure la suppression des fichiers générés lors de la compilation précédente ou tout autre nettoyage spécifique à votre projet.
- **"compile"** : Cette étape utilise la commande "ant compile" pour compiler votre projet Java Swing. Cela permet de générer les fichiers .class à partir des fichiers source de votre projet.
- **"Test"** : Cette étape utilise la commande "ant test" pour exécuter les tests de votre projet Java. Cette étape est destinée à l'exécution des tests unitaires ou d'autres tests automatisés pour vérifier le bon fonctionnement de votre application.
- **"Package"** : Cette étape utilise la commande "ant package" pour créer un package ou un artefact de votre projet Java. Cela peut être un fichier JAR ou tout autre format d'archive contenant les fichiers nécessaires à votre application.
- **"Build"** : Cette étape utilise la commande "ant build" ou "ant jar" pour construire votre projet Java. Cela peut inclure la compilation, les tests, le packaging et d'autres tâches nécessaires pour préparer votre projet pour le déploiement.

- **"Run"** : Cette étape utilise la commande "ant run" pour exécuter votre projet Java. Cela permet de lancer votre application et de vérifier son bon fonctionnement.

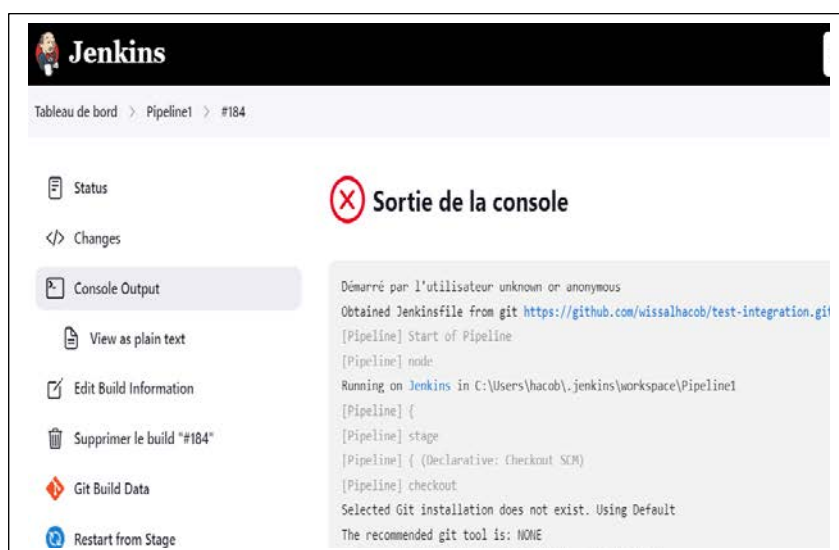
Chaque étape est définie dans un bloc "stage" avec une liste d'étapes spécifiques à exécuter dans le bloc "steps". Les étapes sont exécutées séquentiellement, ce qui signifie que chaque étape est exécutée après l'achèvement de l'étape précédente.

Il convient de noter que les étapes de "Build" et "Run" mentionnées dans le ne sont pas spécifiques aux tests d'intégration. Elles peuvent être utiles pour d'autres phases du processus de développement, telles que la construction de l'application finale et son exécution dans un environnement réel.

Lancer un Build :



S'il y a une erreur :



Sinon :

Jenkins rechercher (C

Tableau de bord > Pipeline1 > #190

Status

Changes

Console Output

View as plain text

Edit Build Information

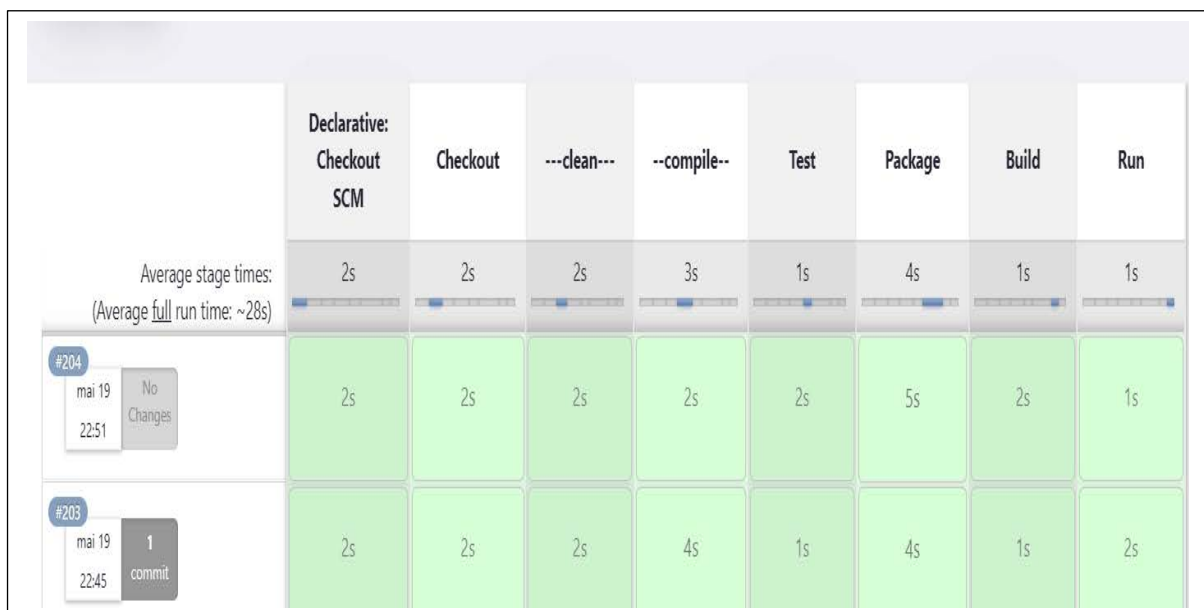
Git Build Data

Thread Dump

Sortie de la console

```
Démarré par l'utilisateur unknown or anonymous
Obtained Jenkinsfile from git https://github.com/wissalhabcb/test-integration.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\Users\hacob\.jenkins\workspace\Pipeline1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
```

Le but :



Conclusion générale

En conclusion, l'utilisation du test continu dans l'application de gestion d'une clinique a été bénéfique pour assurer une intégration continue et une validation régulière du système. Grâce à cette approche, les différentes fonctionnalités et modules ont été testés en continu, ce qui a permis de détecter rapidement les erreurs et les incompatibilités, garantissant ainsi une qualité et une fiabilité accrues du système. Le test continu a également facilité l'identification des problèmes potentiels lors de l'ajout de nouvelles fonctionnalités ou de modifications du code existant. En conclusion, le test continu a contribué à maintenir un système de gestion de clinique robuste, évolutif et performant.

Références :

<https://www.jenkins.io/>

<https://www.jenkins.io/doc/>

<https://www.redhat.com/fr/topics/integration/what-is-integration>

<https://openclassrooms.com/fr/courses/6100311-testez-votre-code-java-pour-realiser-des-applications-de-qualite/6616481-decouvrez-les-tests-dintegration-et-les-tests-fonctionnels>

<https://fr.theastrologypage.com/integration-testing>

<http://publicationslist.org/data/a.april/ref-545/Projet%20Anas%20Ouardirhi.pdf>