

## 

| **SOMMAIRE** |
| --- |
| * **Introduction.** * **Qu'est-ce que le DevOps ?** * **Roadmap.** * **Scénario.** * **Préparer l'environnement de travail.**   **- Installer VirtualBox.**  **- Installer Vagrant.**  **- Installer Jenkins, utils, ansible, java, debian via** Vagrantfile**.**  **- Mettre en place trois serveurs applicatifs (** EnsetservDev, EnsetservStage, EnsetsrvProd **).**  **- Installer le serveur Base de données Postgres**  **- Créer Trois Base de données (** EnsetdbDev, EnsetdbStage, EnsetdbProd **).**  **- Installer Docker Registry, Docker-compose.**  **- La chaîne DevOps , CI /CD.**  **- Git Flow : le WorkFlow dont on a besoin.**  **- Installer Git.**  **- CI / CD en pratique**  **- Lancer un Conteneurs Docker.** |

**Introduction :**

## En ingénierie logicielle, les pratiques agiles mettent en avant la collaboration entre des équipes auto-organisées et pluridisciplinaires et leurs clients. Elles s'appuient sur l'utilisation d'un cadre méthodologique léger mais suffisant centré sur l'humain et la communication. Elles préconisent une planification adaptative, un développement évolutif, une livraison précoce et une amélioration continue, et elles encouragent des réponses flexibles au changement. et puisque les changements fréquents survenant toutes les semaines ou même tous les jours, imposent une nouvelle approche et de nouveaux rôles. C’est ainsi que l’approche **DevOps** a vu le jour.

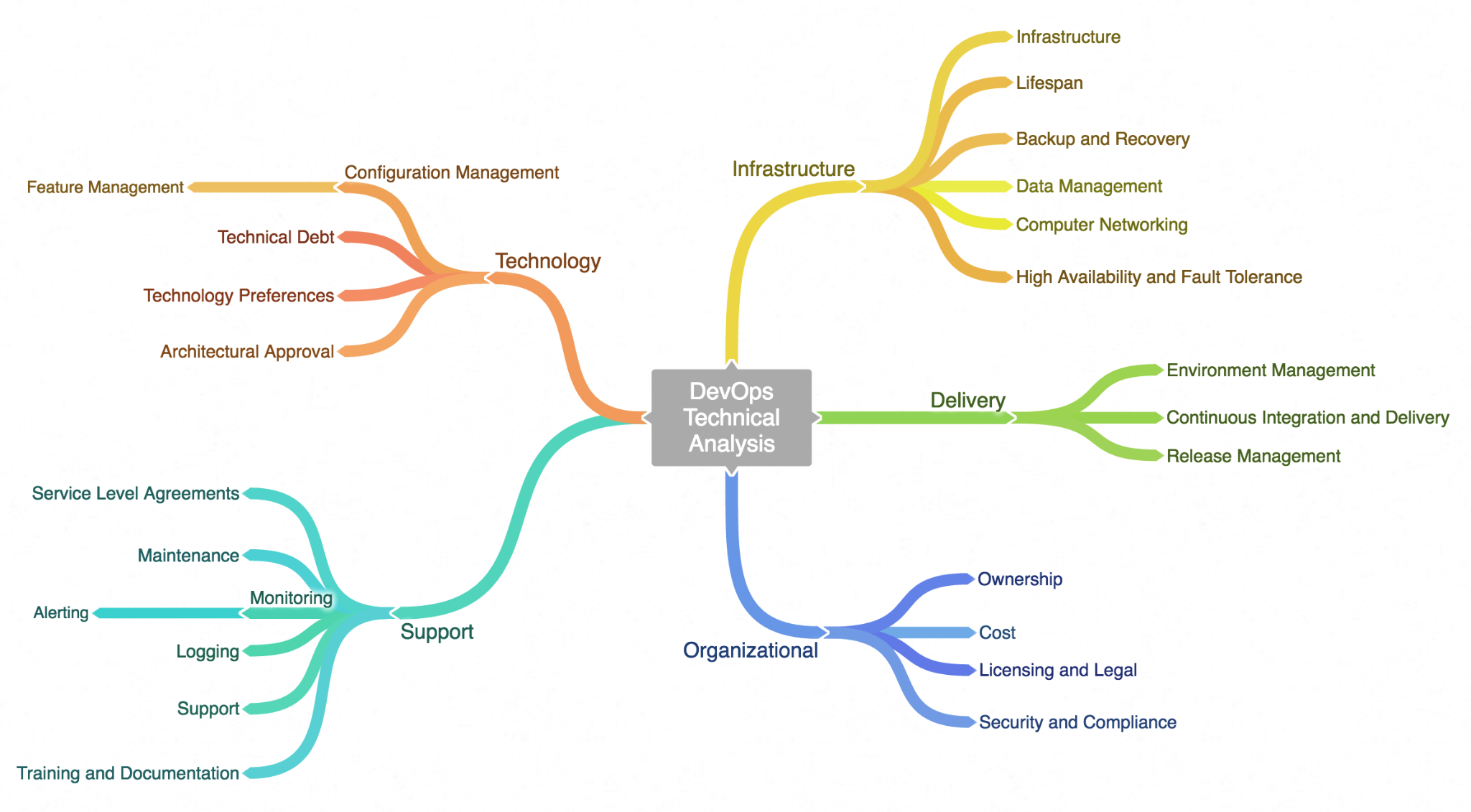
donc, notre application sera basée sur ce principe de **DevOps**.en utilisant toutes les étapes d’intégration et de livraison continue (CI/CD).

**Qu'est-ce que le DevOps ?**

Le terme **DevOps** est composé des termes “**développement**” et “**opérations**”. Il s’agit d’une pratique visant à fusionner le développement, l’assurance qualité, et les opérations à savoir le déploiement et l’intégration en un unique ensemble de processus continus.

Cette méthodologie nouvelle est une extension naturelle **des approches Agile** et de livraisons continues. En adoptant le **DevOps**, les entreprises profitent de trois avantages principaux.

1. Accélérer les lancements de produits.
2. Une meilleure réponse aux besoins.
3. Créer un meilleur environnement de travail.

**Roadmap.**

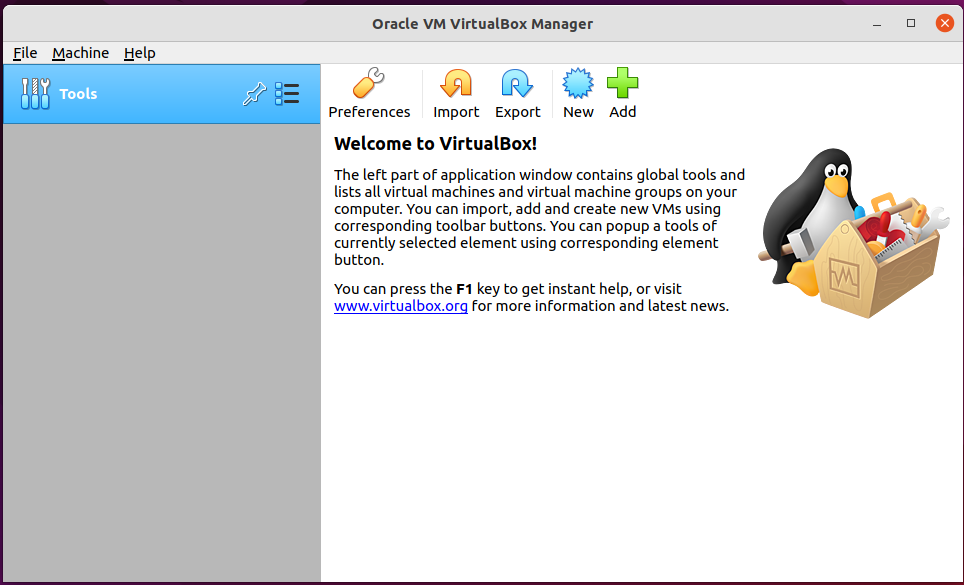
Roadmap est une étape très importante pour mettre en place le concept de DevOps, Construire une équipe DevOps performante est un défi de taille. Construire une équipe Agile DevOps performante peut être un énorme défi.

* **Scénario**

****

**Préparer l’environnement de travail.**

**- Installer VirtualBox.**

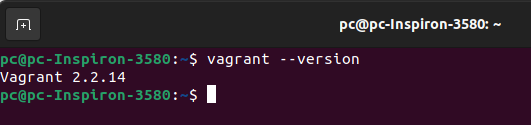
****

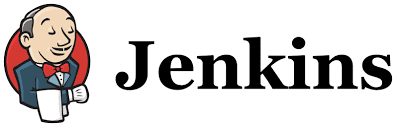
L’installation de la machine virtuel dont nous avons besoin

**- Installer Vagrant .**

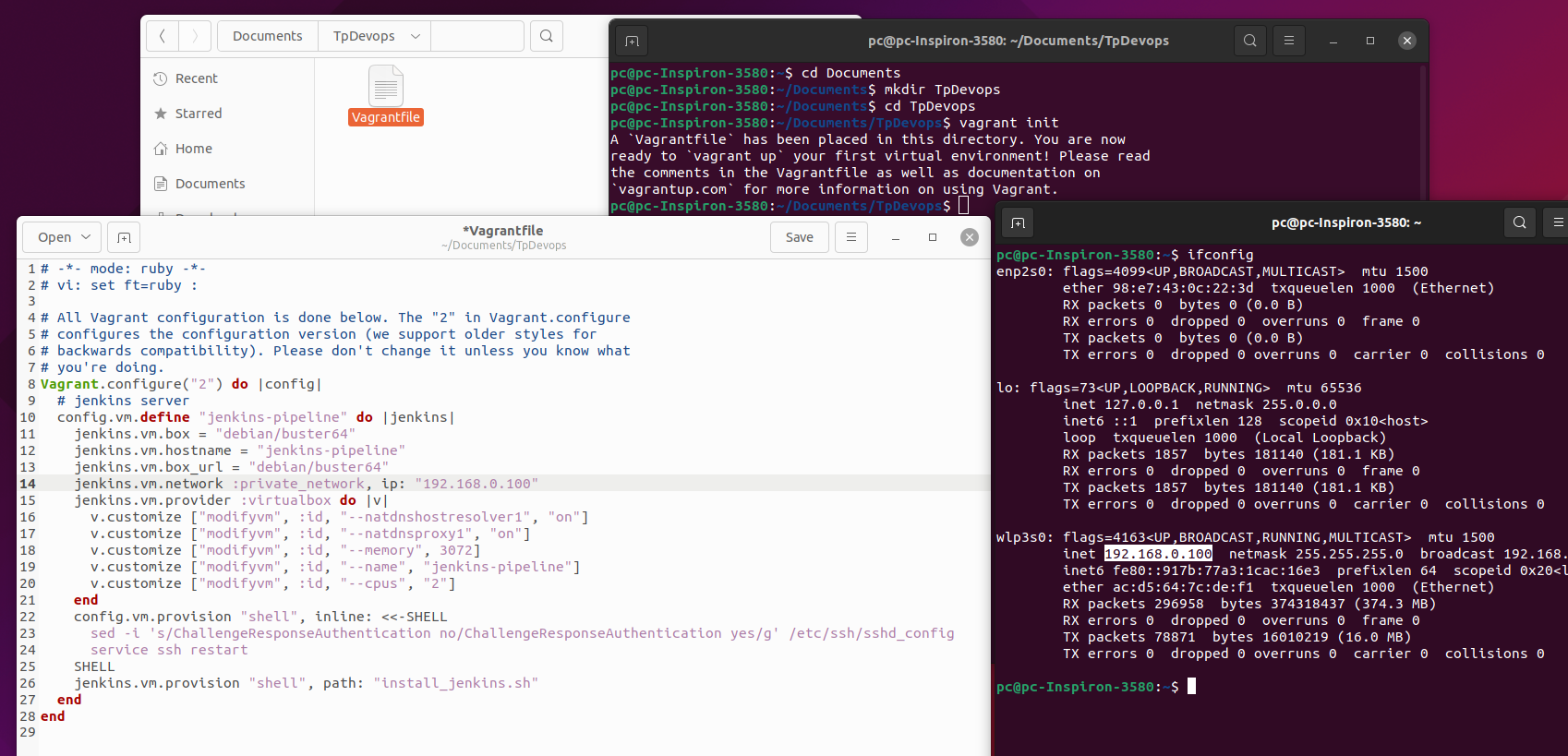
Vagrant est un logiciel libre et open-source pour la création et la configuration des environnements de développement virtuel. Il peut être considéré comme un wrapper autour de logiciels de virtualisation comme VirtualBox.

Nb: installer Vagrant dans la machine hôte.



**- Installer le serveur Jenkins.**

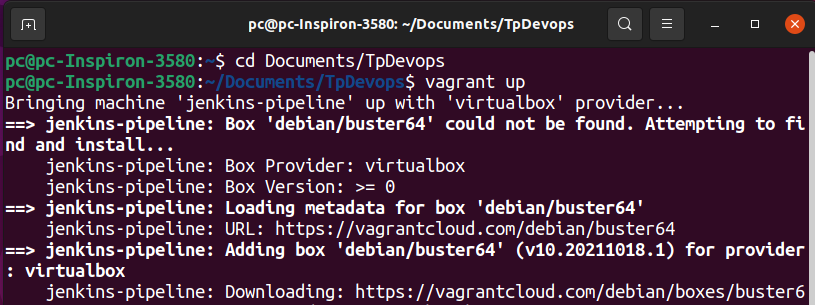
Jenkins est un outil open source de serveur d'automatisation. Il aide à automatiser les parties du développement logiciel liées au build, aux tests et au déploiement, et facilite l'intégration continue et la livraison continue.

Nous allons procéder à l'installation de notre serveur jenkins via vagrantfile écrit avec le langage ruby, comme illustré dans la figure ci-dessous.

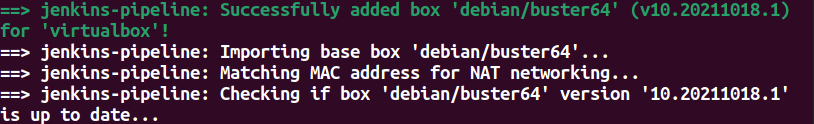
lien vers le fichier source vagrantfile et [install\_jenkins.sh](https://github.com/biko2020/Devops_Project/tree/master/Vagrant_Files)

Se déplacer au répertoir qui contient le fichier vagrantfile et puis

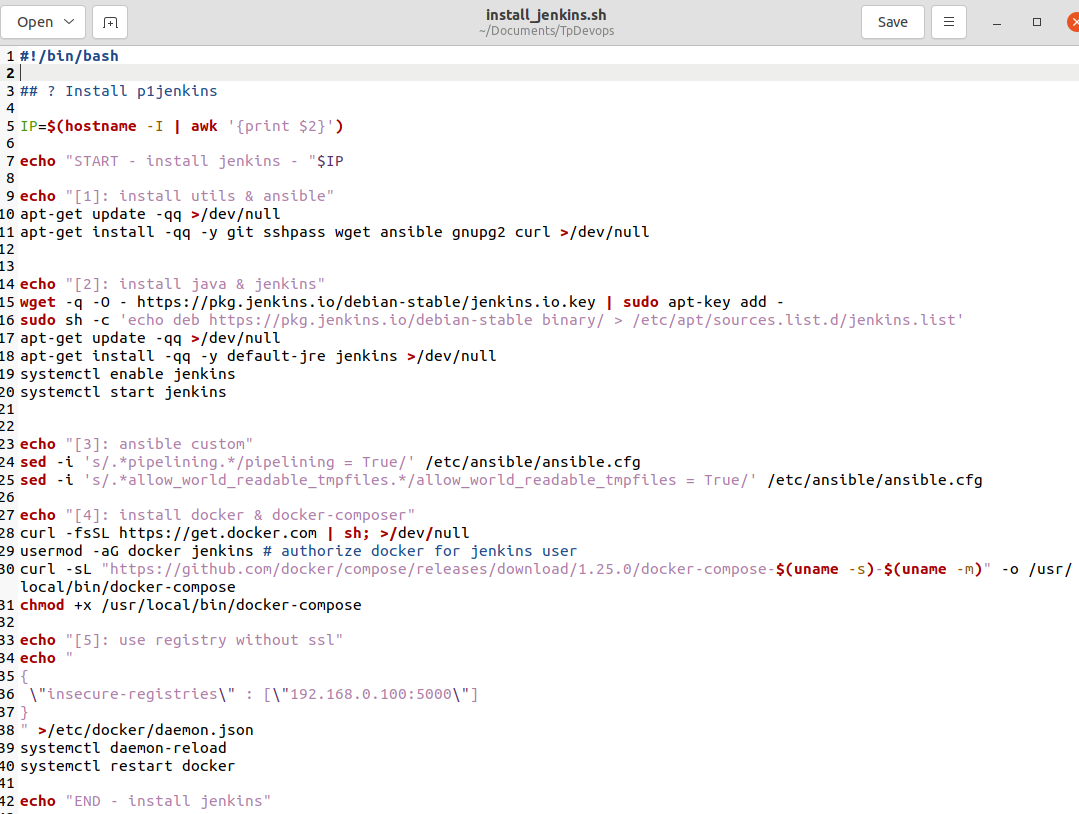
lancer la commande →*vagrant up* , vagrant détecte automatiquement la virtualbox



l’installation de jenkins et java ,install utils et ansible dans une machine debian est terminée avec succès.

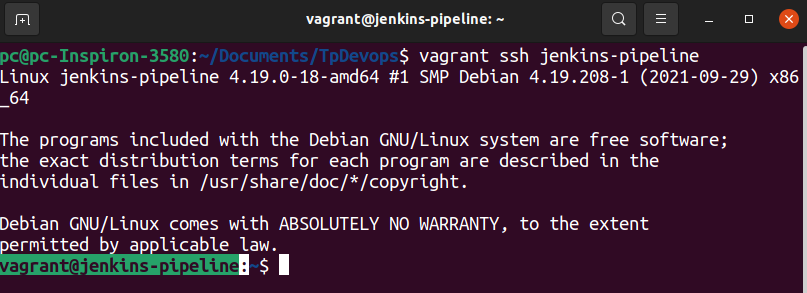


Ci-après le contenu du fichier [install\_jenkins.sh.](https://github.com/biko2020/Devops_Project/tree/master/Vagrant_Files)

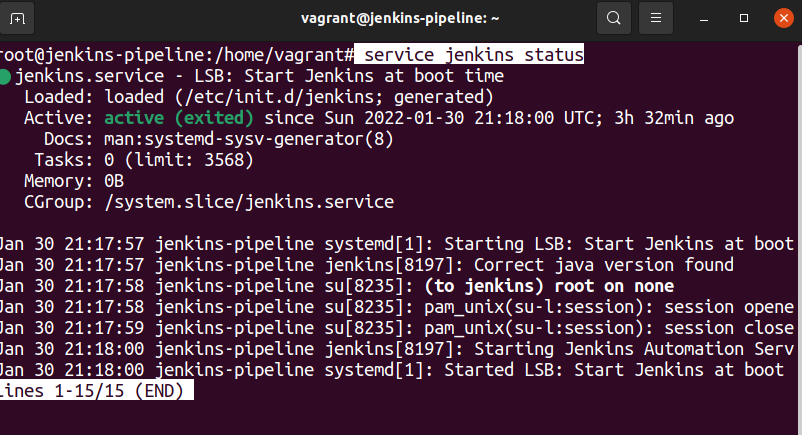


donc, je vais me connecter a ma machine virtuelle depuis ma machine hôte en mode ssh avec la commande

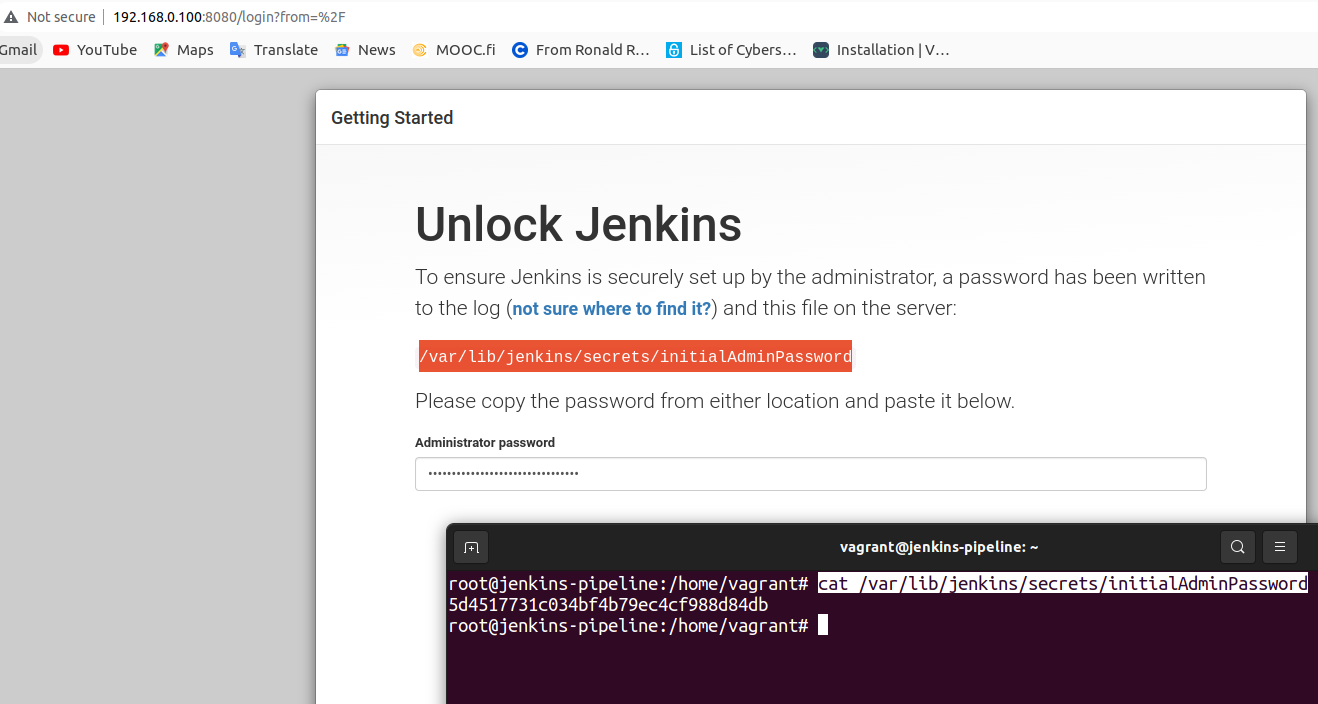
→ *vagrant ssh le nom de la machine* comme illustré dans la figure ci-dessous



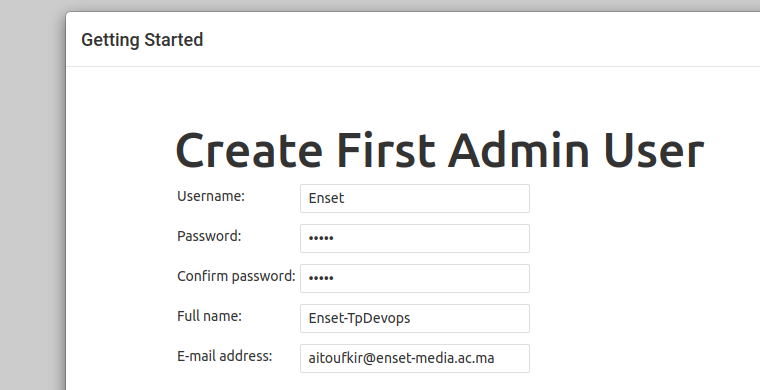
Après avoir démarré la machine, je vérifie l’état de jenkins avec la commande → *service jenkins status* . donc, jenkins a bien démarré comme montré dans la figure ci-dessous.



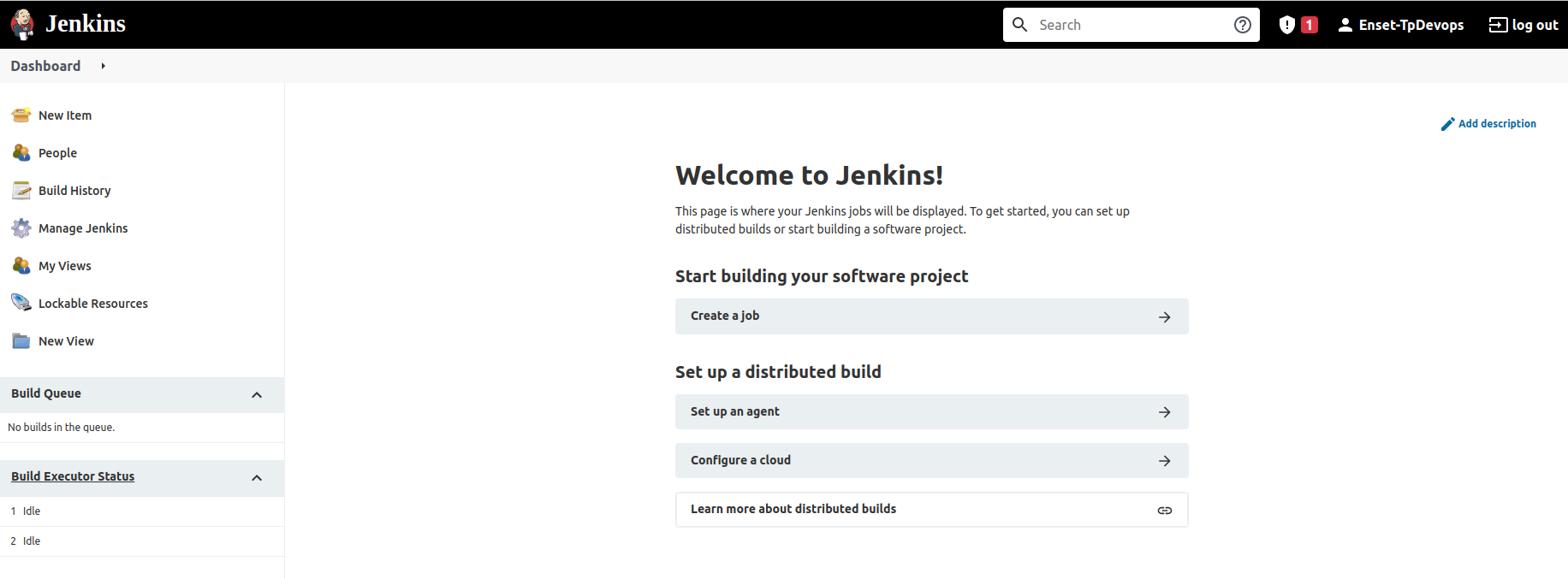
Pour démarrer jenkins taper sur le navigateur l’adresse ip de la machine suivi par le port :8080 ex “<http://192.168.0.100:8080/>” vous aurez l’interface ci-dessous et pour activer jenkins copier le lien /var/lib/jenkins/secrets/initialAdminPassword sur votre terminal précédé par la commande → *cat /var/lib/jenkins/secrets/initialAdminPassword*



Inscrivez-vous



Voilà à quoi ressemble l’interface de jenkins.

- **Mettre en place trois serveurs applicatifs (** EnsetservDev, EnsetservStage, EnsetsrvProd **).**

donc, après l’installation de notre serveur jenkins on va procédé a l’installation de nos trois serveurs (*ensetsrvdev)* pour le développement de notre application et le serveur (*ensetsrvstage*) pour faire les testes et enfin le *master* qui sera notre serveur de production(*ensetsrvprod*) , bien sur la création de nos serveurs sera effectué d’une manière automatique via l’ajout du code suivant au fichier *vagrantfile.*



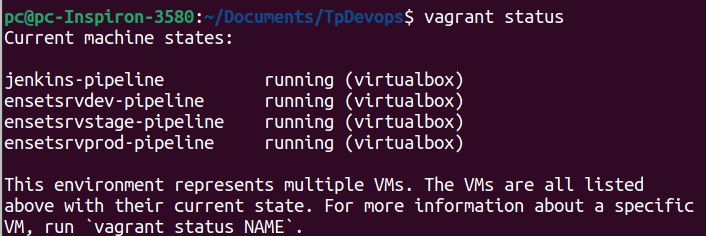
le lien vers le fichier[*vagrantfile*](https://github.com/biko2020/Devops_Project/tree/master/Vagrant_Files)

Pour démarrer l’installation des trois serveurs(*dev- stage- pro* ) au terminal j’exécute la commande suivante

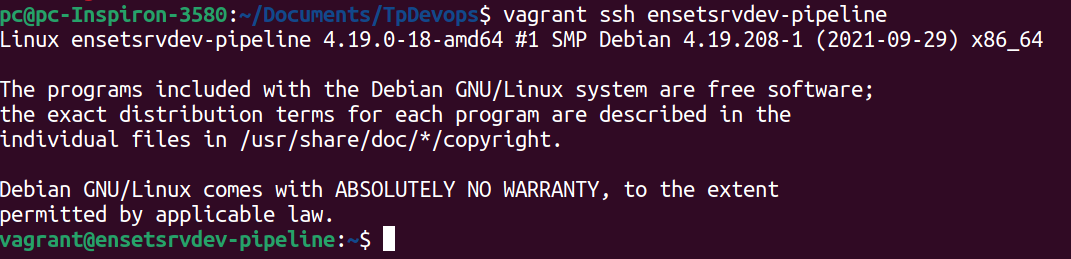
→*vagrant up*



afficher les serveurs installé avec la commande → *vagrant status*

**

démarrer le serveur ensetsrvdev → *vagrant ssh ensetsrvdev-pipeline*

**

→$ *logout* pour se déconnecter du server

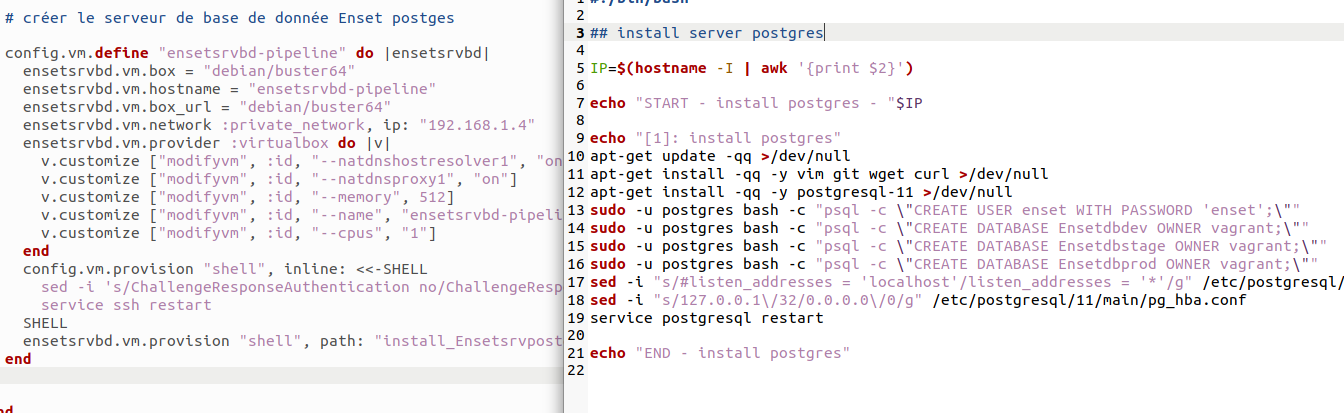
→$ *grep -ri ip:* lister les adresses ip des serveurs.

→$ *ssh vagrant@192.168.1.1* permet de se connecter au serveur via son ip.

**- Installer le serveur base de données Postgres et créer Trois Base de données (** EnsetdbDev, EnsetdbStage, EnsetdbProd  **).**

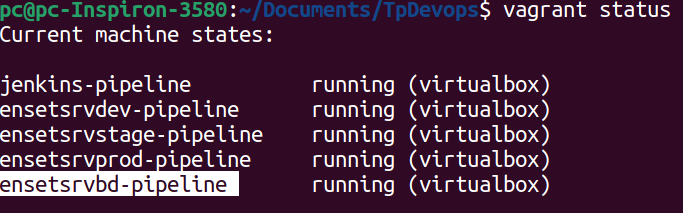
Donc, je vais créer un serveur base de données postgres avec trois base de données pour chacun de nos serveurs applicatifs.

→$ *vagrant up*

**

le lien vers le fichier[*install\_Ensetsrvpostgres.sh*](https://github.com/biko2020/Devops_Project/tree/master/Vagrant_Files)

→ $ *vagrant status* l’installation de notre serveur base de donnée Postgres

**

**- Installer Docker Registry, Docker-composer.**

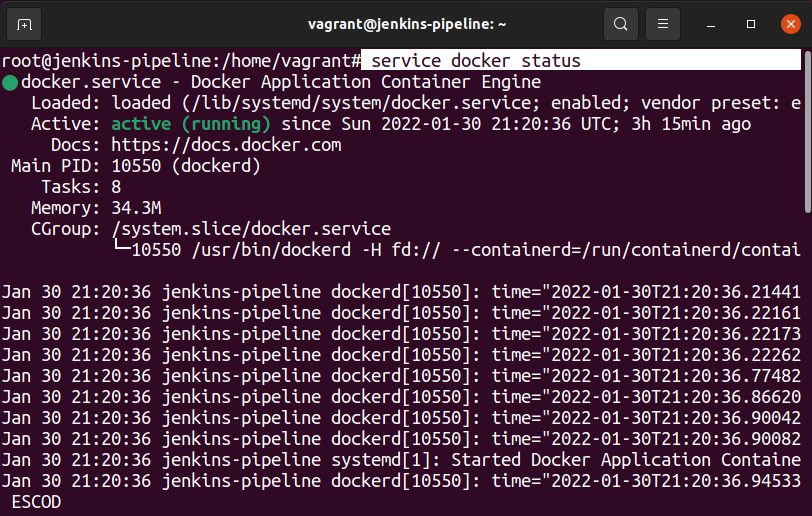
Docker Registry : est un moyen ordinaire de stocker et de distribuer des images Docker, est un référentiel open source sous licence d'autorisation Apache. il permet également d'améliorer le contrôle d'accès et la sécurité des images Docker stockées dans son référentiel.

Ajouter le fichier registry responsable à l'installation de conteneur docker.



le lien vers le fichier[*install\_registry.sh*](https://github.com/biko2020/Devops_Project/tree/master/Vagrant_Files)

De même, je vérifie l’état de docker avec la commande → *service docker status*, le résultat de la vérification est figuré dans l’image ci-après.



**La chaîne DevOps , CI /CD**

****

Pour mieux comprendre l'importance de la coordination et de la communication, nous devons comprendre comment chaque phase est liée à l'autre, créant une chaîne de production pour le logiciel.

ci-après l’architecture logiciel adoptée pour notre TP.

donc, nous allons passer au pratiqué pour tester les techniques d'intégration continue et celle de déploiement continu via le distribution de notre projet microservice.

**Git flow : préparer notre environnement de travail sur gitHub.**

Git Flow sépare sur des branches isolées le développement et le code validé et testé. Pour cela, il s’appuie sur la création de plusieurs branches dont le cycle de vie est bien défini. Voici une table contenant leurs noms, leurs cycles de vie et leurs fonctions :

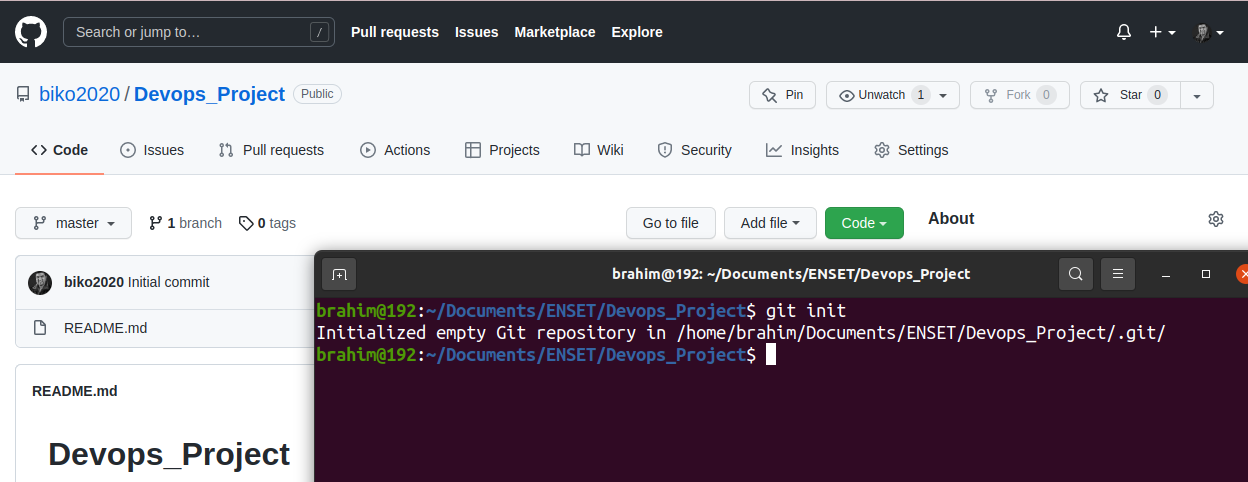
| **Branche** | **Nombre** | **Branche**  **d’origine** | **Durée de vie** | **Fonction** |
| --- | --- | --- | --- | --- |
| master | Unique |  | Permanente | Code stable, testé et validé potentiellement  éligible pour une MEP (Mise En Production) |
| feature | Plusieurs | develop | Développement  d’une feature | Code en cours de développement qui implémente  une fonctionnalité à embarquer  dans la prochaine version de l’application |
| develop | Unique | master | Permanente | Code de la prochaine version de l’application.  Une fois que le développement d’une fonctionnalité  (feature)est fini, le code est fusionné sur cette branche |
| release | Unique | develop | Recette | Branche sur laquelle on corrigera les bugs  détectés pendant la phase de recette |
| hotfix | Aucune/  Plusieurs | master | Correction d’un bug | Branche où on fait les corrections des bugs sur  le code présent sur la branche master (production) |

**- Installer Git.**

voila le script

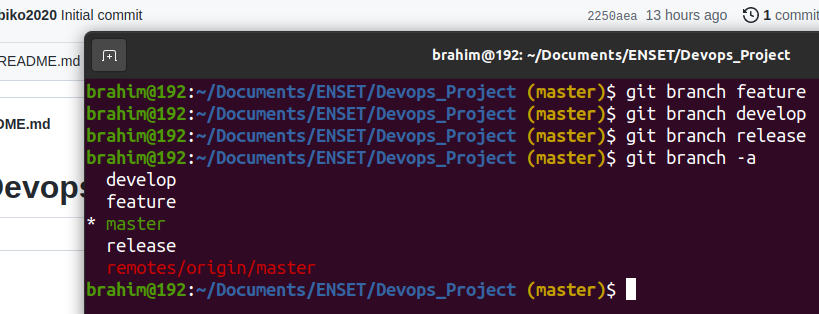


Donc, après l’installation de git , j’ai créer et initialiser et ajouter l’application java (Microservice) au répertoire du projet sur gitHub.



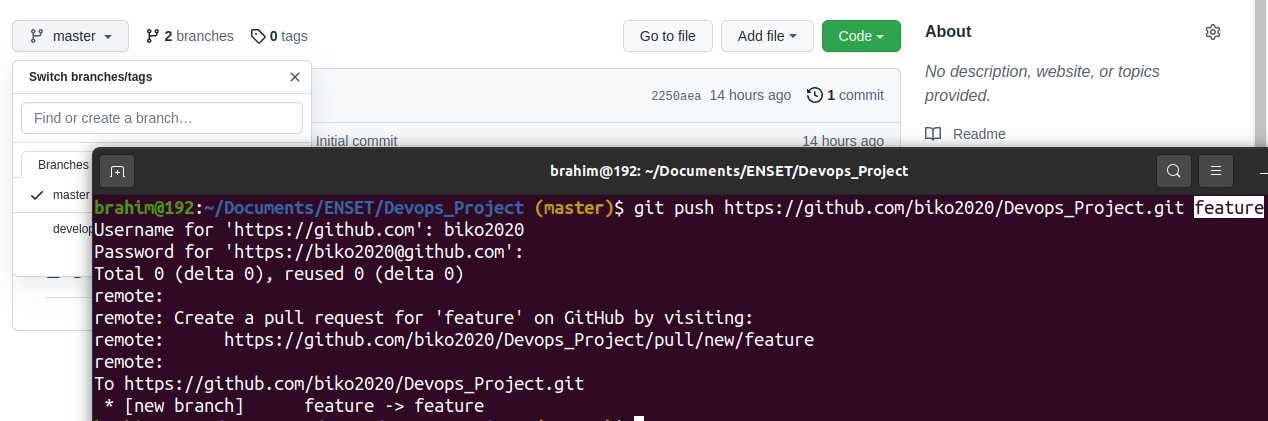
Créer les branches nécessaires via ligne de commande pour notre projet.

*→ git branch* feature



push les branches vers mon dépôt à distance(sur gitHub).

→ *git push https://github.com/biko2020/Devops\_Project.git* feature



—---------------------------------------------------------------------------------------------

To show branch name in "git log" outputs

→ git config --global log.decorate auto

To show current branch name in Ubuntu Terminal type below commands to related lines in the → vim ~/.bashrc

Find “PS1” in .bashrc like:

if [ "$color\_prompt" = yes ]; then

PS1= ...

else

PS1= ...

fi

instead of above lines, paste blow lines

# git branch info if present

parse\_git\_branch() {

git branch 2> /dev/null | sed -e '/^[^\*]/d' -e 's/\* \(.\*\)/ (\1)/'

}

if [ "$color\_prompt" = yes ]; then

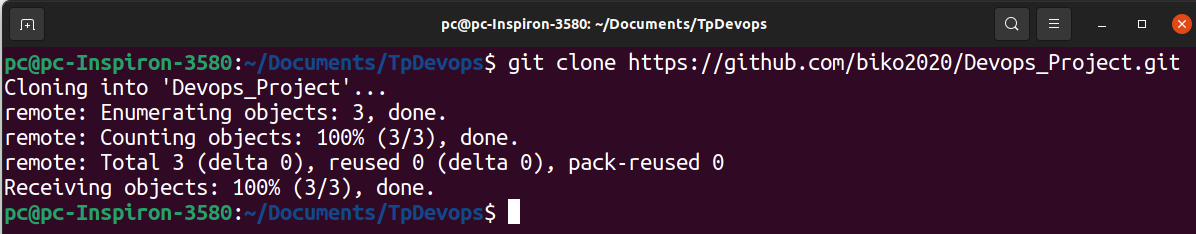
PS1='${debian\_chroot:+($debian\_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[33m\]$(parse\_git\_branch)\[\033[00m\]\$ '

else

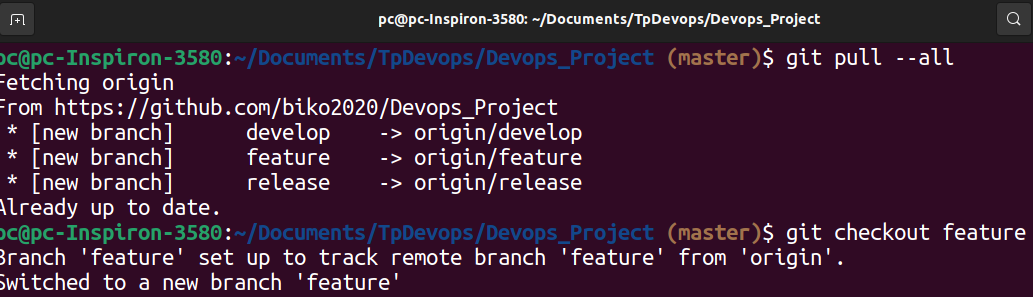
PS1='${debian\_chroot:+($debian\_chroot)}\u@\h:\w$(parse\_git\_branch)\$ '

fi

je vais cloner mon dépôt gitHub dans une autre machine afin que je puisse travailler de dont.

****

Commencer le travailler sur la branche feature

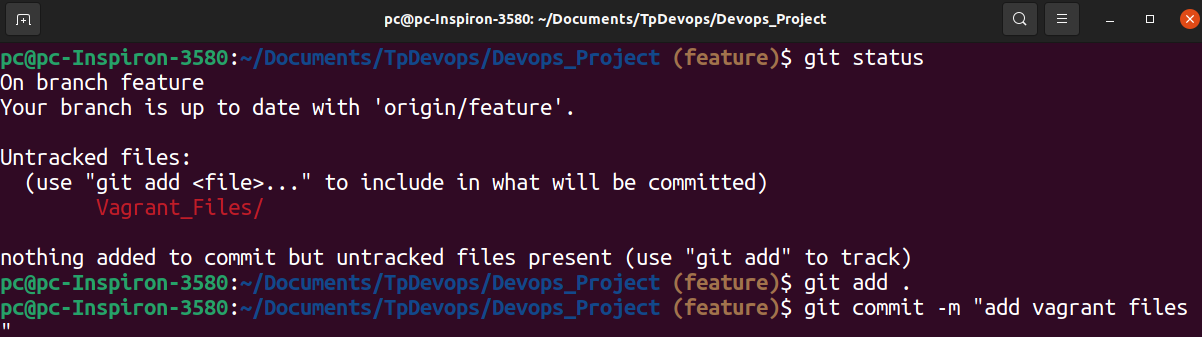


*→ git status // vérifier l’état de mon dépôt changements*

*→ git add . // ajouter les changements*

*→ git commit -m “” // faire des commentaires*

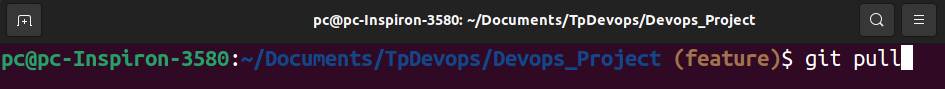
*→ git push -u origin feature // envoyer les modifications e*

****

*→ git checkout feature // se déplacer vers la branche feature*

*(feature) → git merge master // merger feature avec master*

* **CI / CD en pratique**

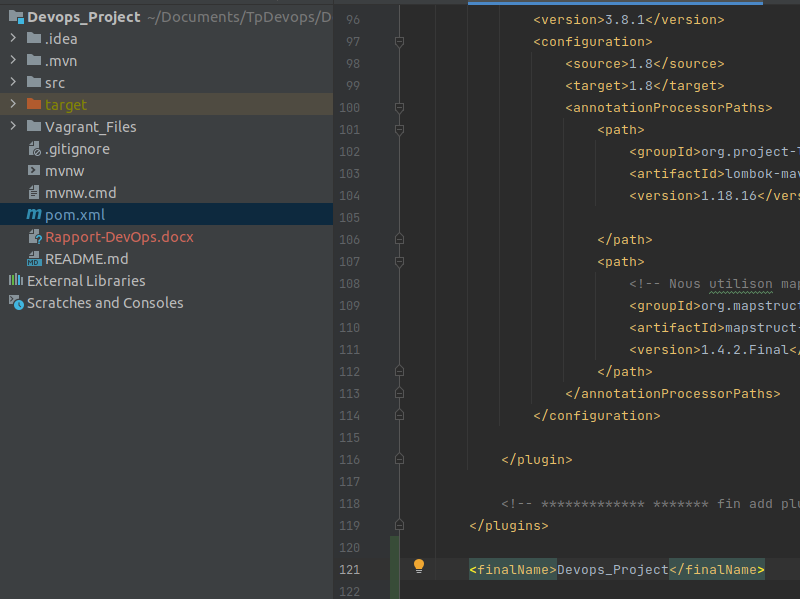
****

Référence le nom du dépôt gitHub sur l’application java , exactement sur le fichier *pom.xml.*

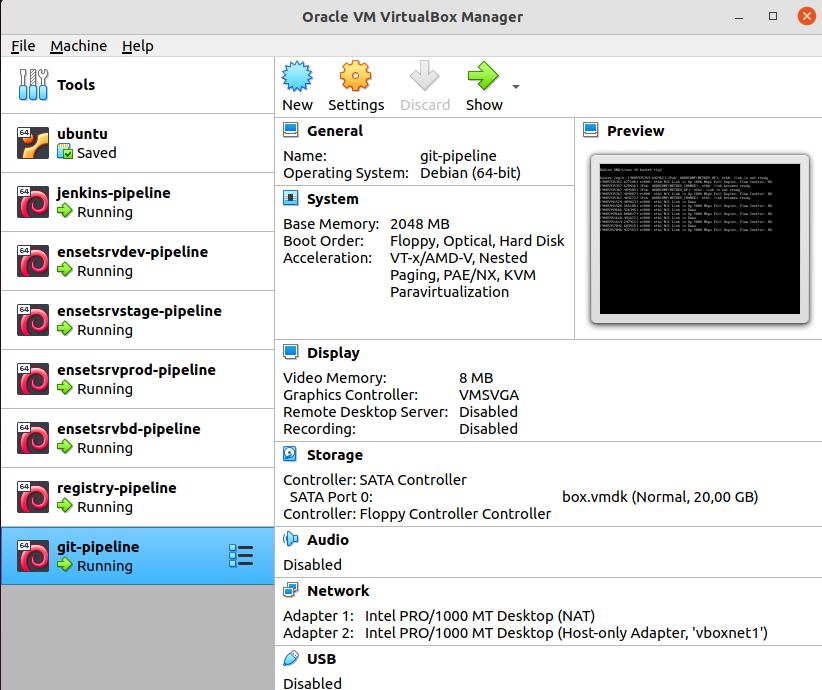
*<build>*

*<finalName>Devops\_Project</finalName>*

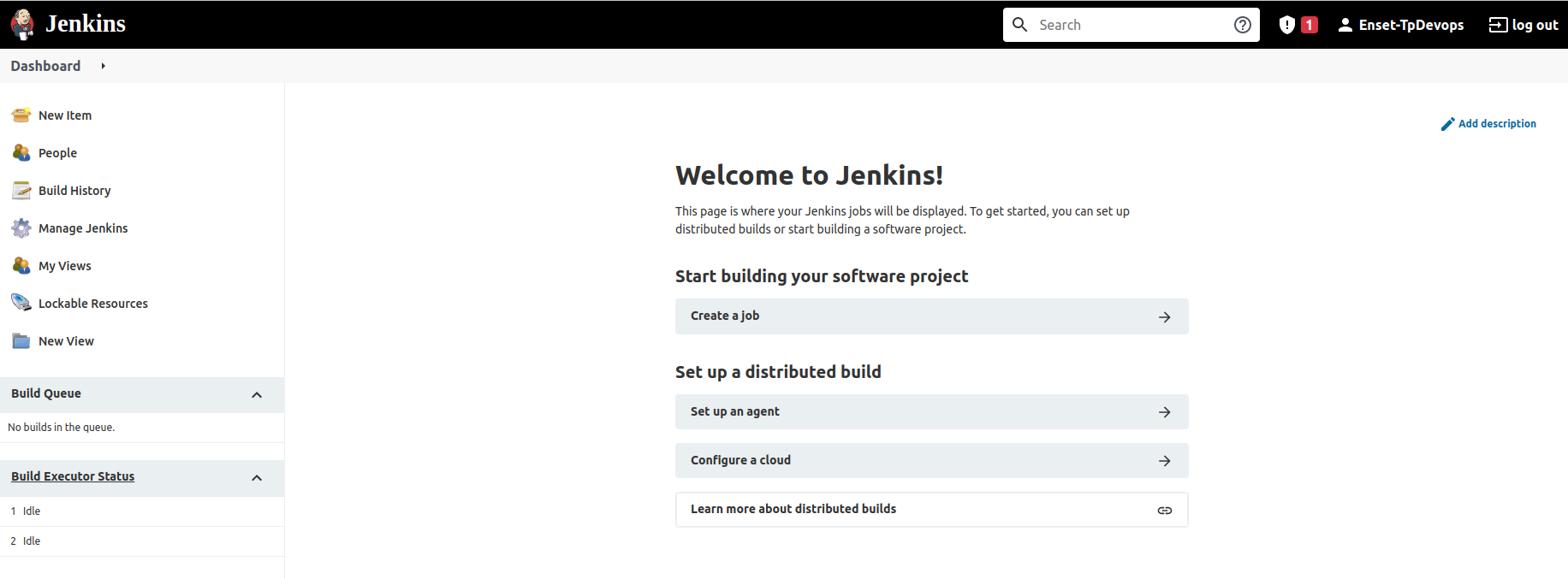
*</build>*

**

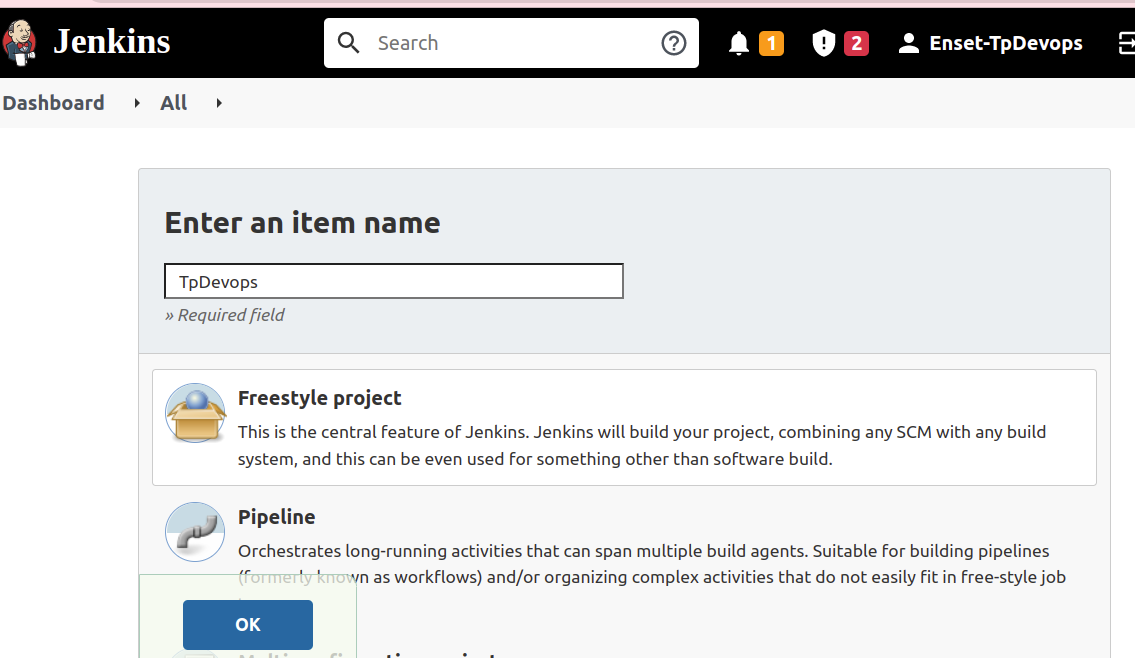
Lancer le serveur *jenkins-pipeline* depuis la machine virtuel avec la commande *vagrant up* et après on va configurer le serveur jenkins afin qu'il puisse communiquer avec notre dépôt gitHub.

**

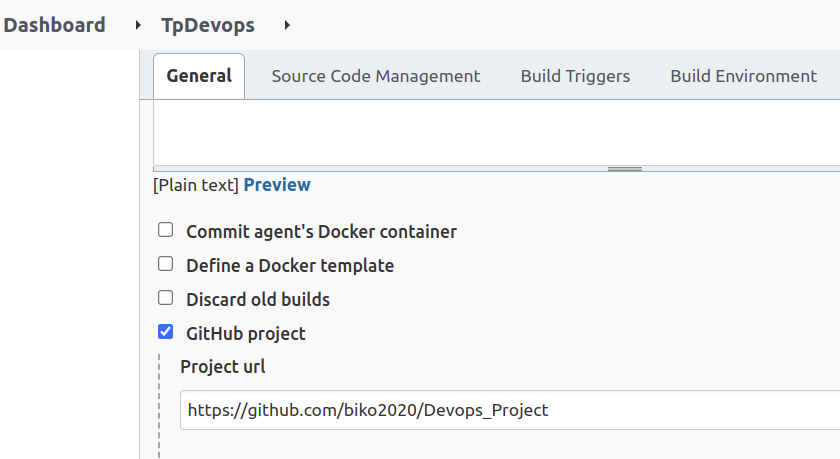
Pour démarrer jenkins taper sur le navigateur l’adresse ip de la machine suivi par le port :8080 ex *“*[*http://localhost:8080/*](http://192.168.0.100:8080/)*”* vous aurez l’interface ci-dessous et pour activer jenkins copier le lien.



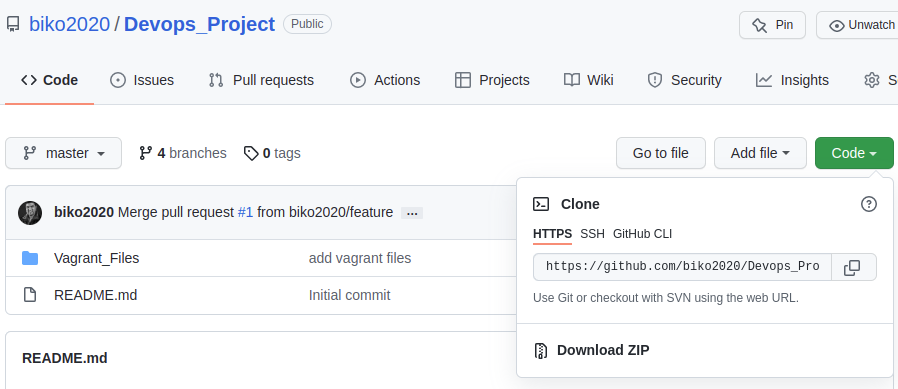
Créer new job:



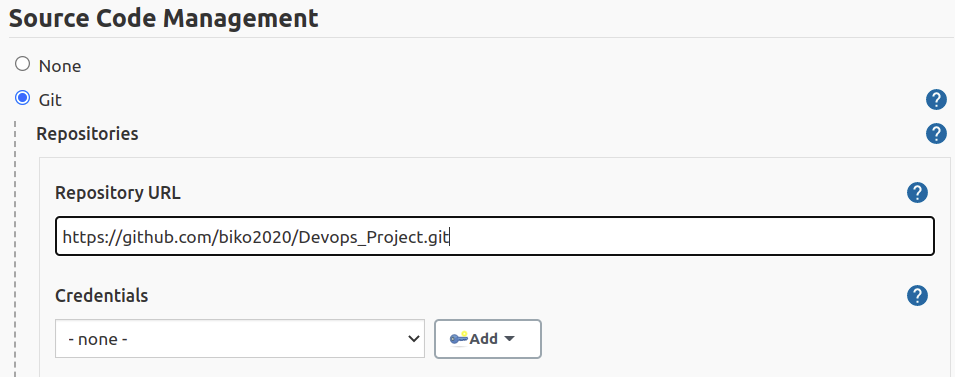
Entrez le lien du projet GitHub project



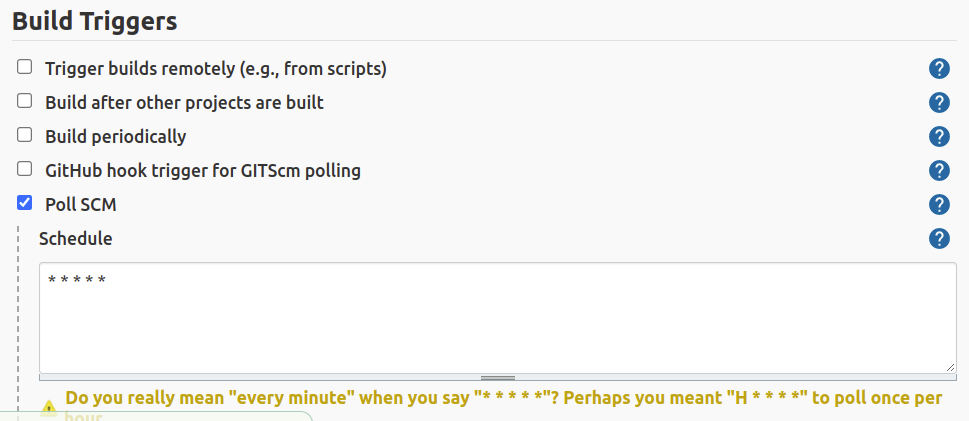
Entrez le lien de la source du dépôt gitHub



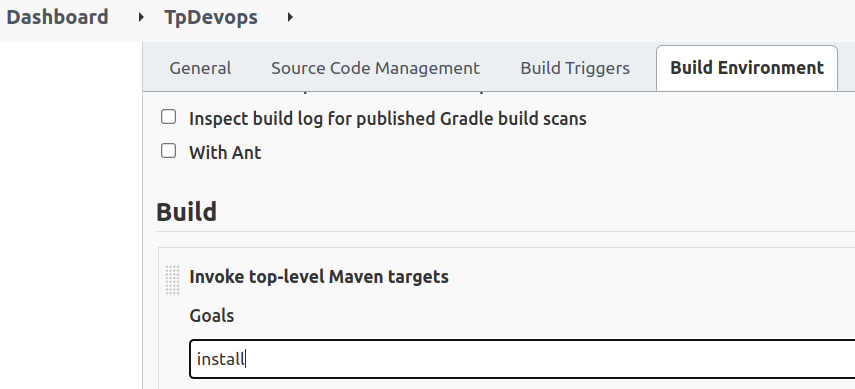
coller le lien au source code Management



Entrez le cron souhaité pour interroger le SCM. Ici, nous avons donné \* \* \* \* \* ce qui signifie que Jenkins interroge le SCM toutes les minutes. veuillez vous référer à la Figure ci-après.

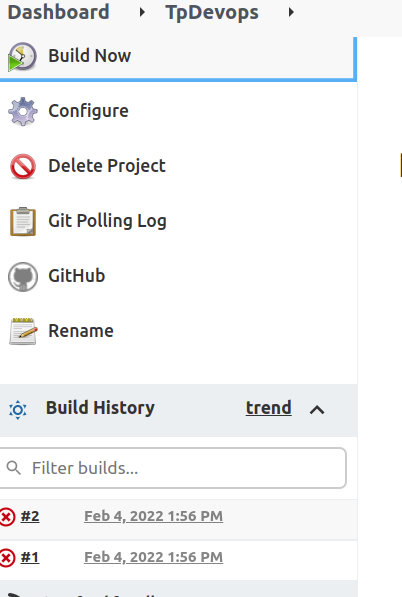


Entrez le nom de l'élément et sélectionnez le projet Maven et cliquez sur appliquer comme indiqué dans la figure ci-dessous.



Pour tester le build.

réussite de la construction



**- Lancer un Conteneurs Docker.**

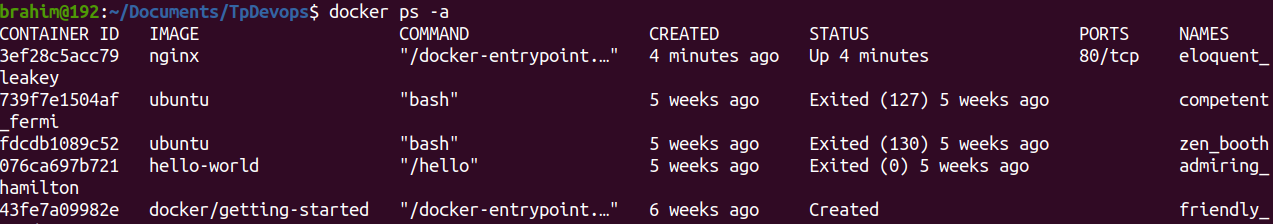
Lancer et manipuler le conteneurs docker

*→ docker ps // le conteneur active*

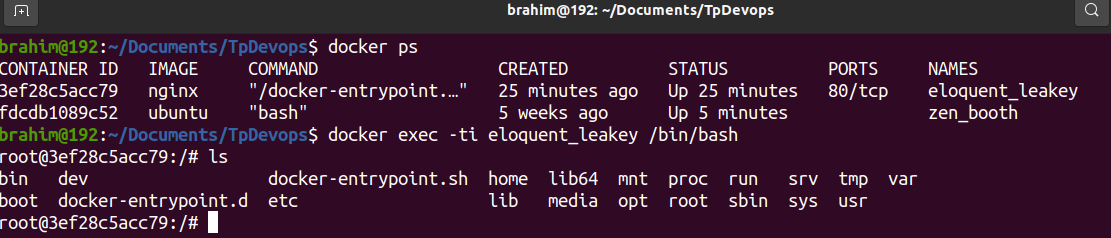
*→ docker ps -a // lister toutes les conteurs*

*→ docker start // démarrer un conteur*

*→ docker restart // recharger le conteure*

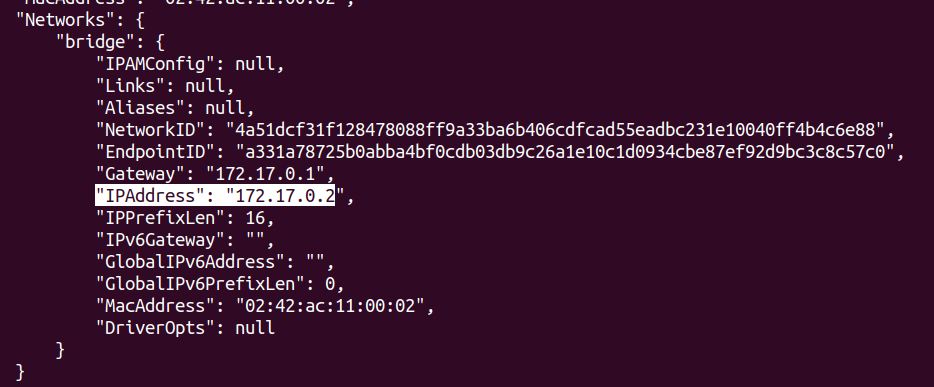


Pour manipuler le conteneur nginx je vais accéder à ce conteneur en tant que root.

****

Avec la commande inspect nous permet d’avoir toutes les informations sur le conteneur en question, mais ce qui nous intéresse de plus c’est l’aspect Réseaux (bridge) exactement l’IP du conteneur .

*→ docker inspect eloquent\_leakey*

****

*→ curl 127.17.0.2 // accéder la page home de nginx*

**Conclusion :**

Le mouvement DevOps tend à fusionner ces deux mondes en améliorant leur communication. 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.

## 