Installation du docker Jenkins

- Dans un terminal Ubuntu, exécuter les commandes suivantes :

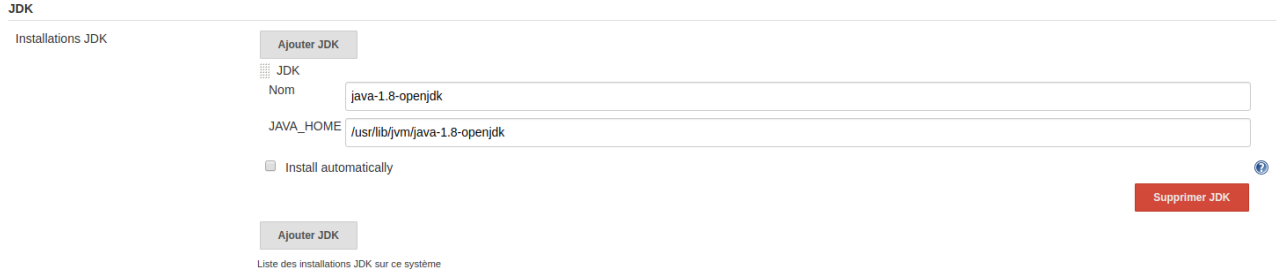
```
docker pull frouland/myjenkins:0.2
docker run -d --name JenkinsCI -e http_proxy='proxy.insa-rouen.fr:3128' -e https_proxy='proxy.insa-rouen.fr:3128' -p
8080:8080 -p 50000:50000 frouland/myjenkins:0.2
```

- Dans un navigateur, ouvrir l'URL : http://10.0.0.100:8080/
- Ajout du mot de passe admin pour débloquer Jenkins :

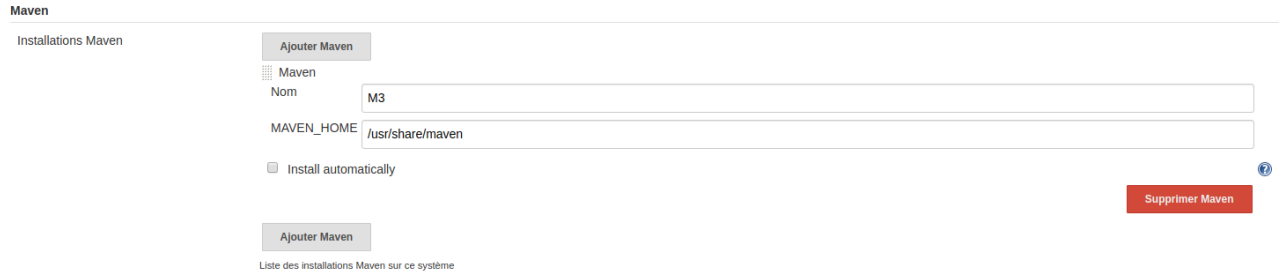
```
docker exec -it JenkinsCI cat /var/jenkins_home/secrets/initialAdminPassword
```

- Mise à jour du proxy dans Jenkins pour les plugins : http://10.0.0.100:8080/pluginManager/advanced
- Installation des plugins par défaut
- Aller dans "**Administrer Jenkins > Configuration globale des outils**" pour configurer les outils suivants :

1. JDK *(décocher Install automatically)* :
   - Nom : java-1.8-openjdk
   - JAVA_HOME : /usr/lib/jvm/java-1.8-openjdk



2. Maven *(décocher Install automatically)* :
   - Nom : M3
   - MAVEN_HOME : /usr/share/maven



Configuration du proxy pour Maven dans le docker JenkinsCI

- Dans un terminal Ubuntu, exécuter les commandes suivantes :

```
docker exec -it JenkinsCI bash
dos2unix /usr/share/maven/conf/settings.xml
```

- Editer le fichier "**/usr/share/maven/conf/settings.xml**" et ajouter les lignes suivantes dans la section **proxies** :

```
<proxy>
        <id>optional</id>
        <active>true</active>
        <protocol>http</protocol>
        <username></username>
        <password></password>
        <host>proxy.insa-rouen.fr</host>
        <port>3128</port>
```

```
        <nonProxyHosts>local.net|some.host.com|10.0.0.100</nonProxyHosts>
</proxy>
```

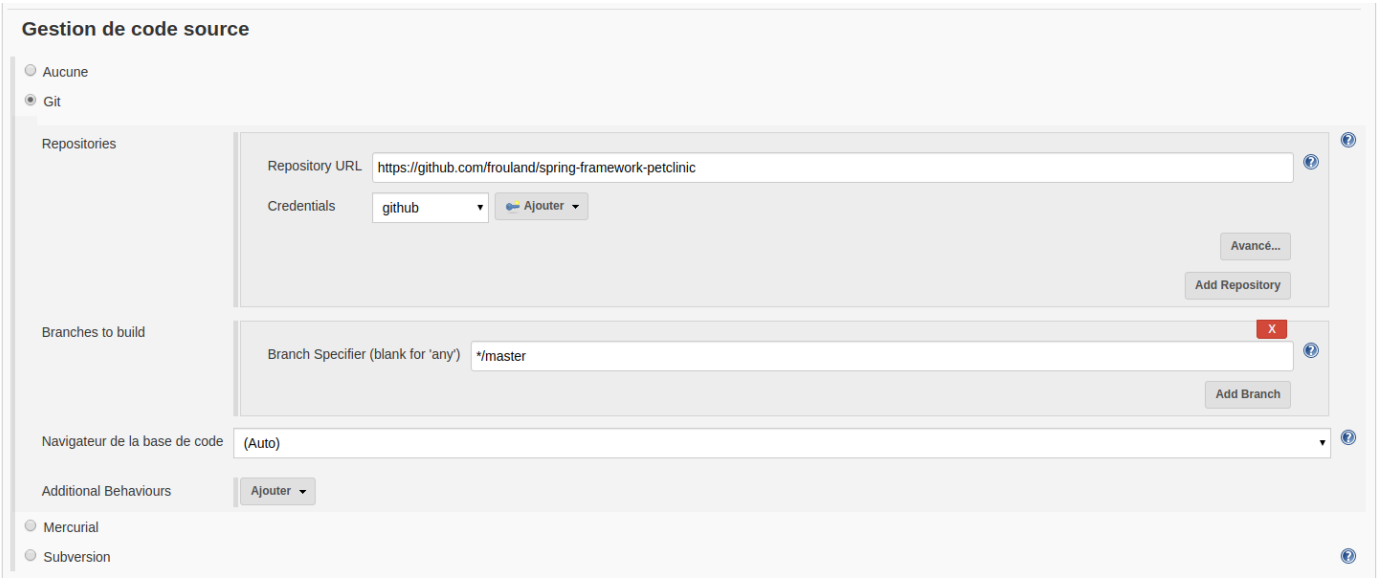
Création d'un projet free-style

- Créer un projet free-style nommé "**Petclinic**" qui devra :

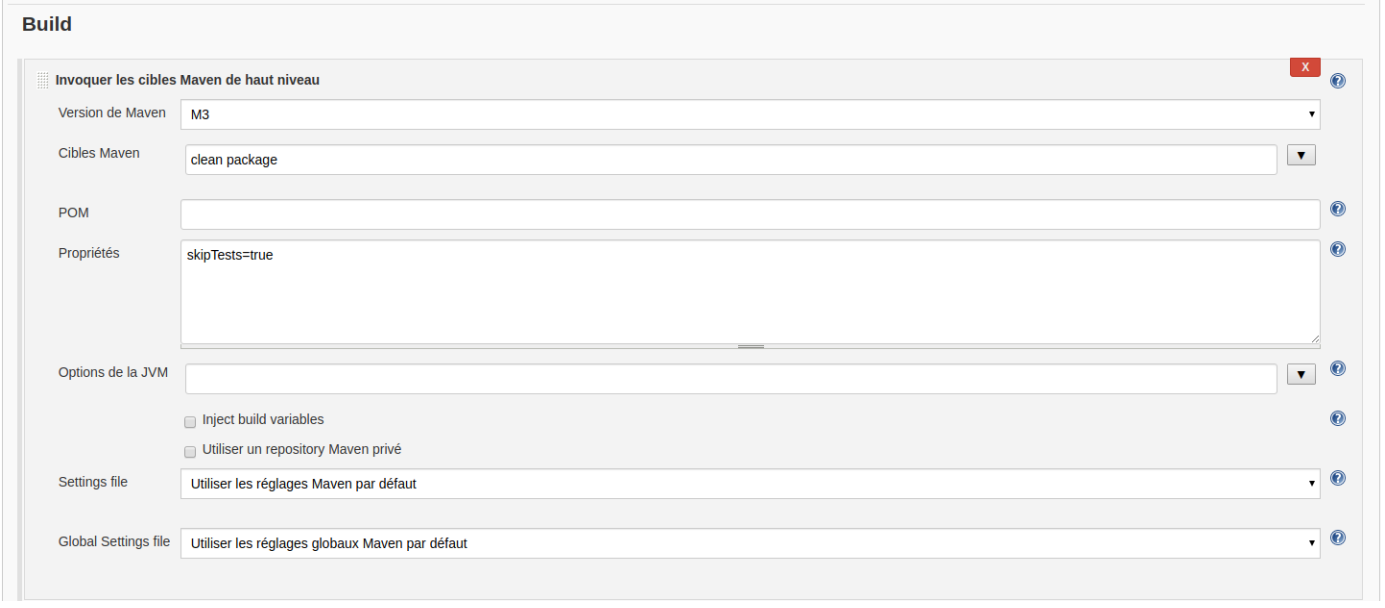
1. Récupérer les sources de **spring-framework-petclinic** dans votre GitHub
    - Créer une clé SSH dans le docker JenkinsCI avec la commande :

```
ssh-keygen -t rsa -b 4096 -C "<votre_adresse_mail>"
```

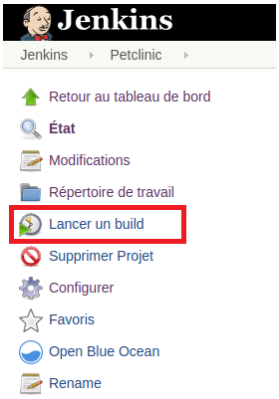
```
- Ajouter la nouvelle clé SSH "**~/.ssh/id_rsa.pub**" dans votre compte GitHub
- Créer un credential de type "**SSH Username with private key**"
- Renseigner un ID, un username, la "**Private Key**" et la passphrase (si nécéssaire)
```



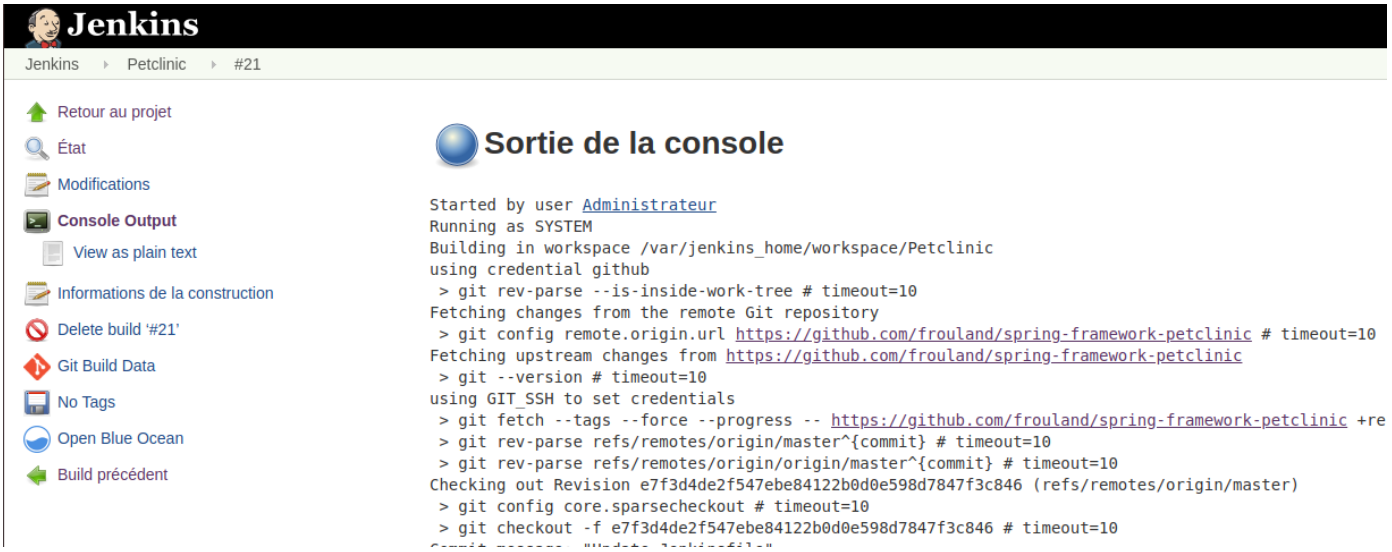
2. Créer une tâche du build avec le type "**Invoquer les cibles Maven de haut niveau**"
3. Sélectionner la version de maven
4. Appel des cibles "**clean package**"
5. Dans "**Avancé...**", ajouter la propriété "**skipTests=true**" pour éviter le lancement des tests

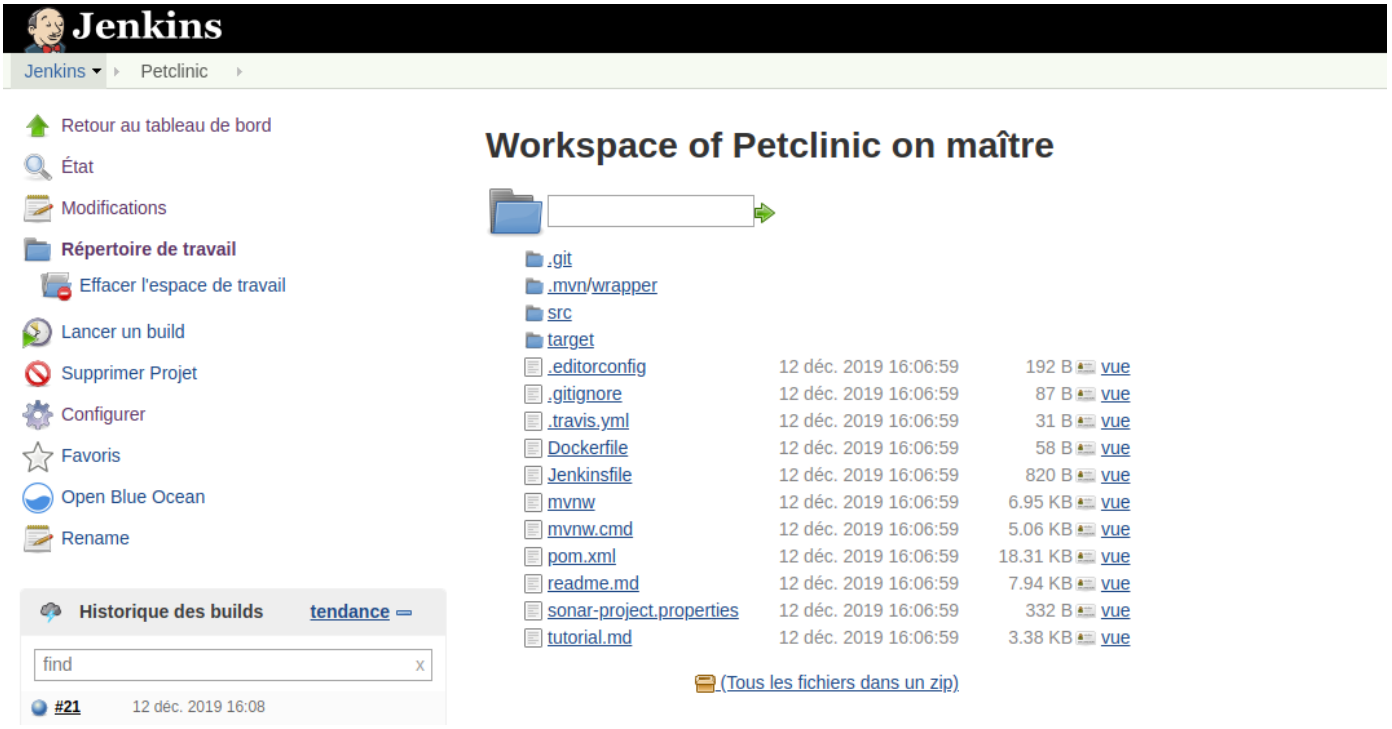


6. Lancer le build manuellement



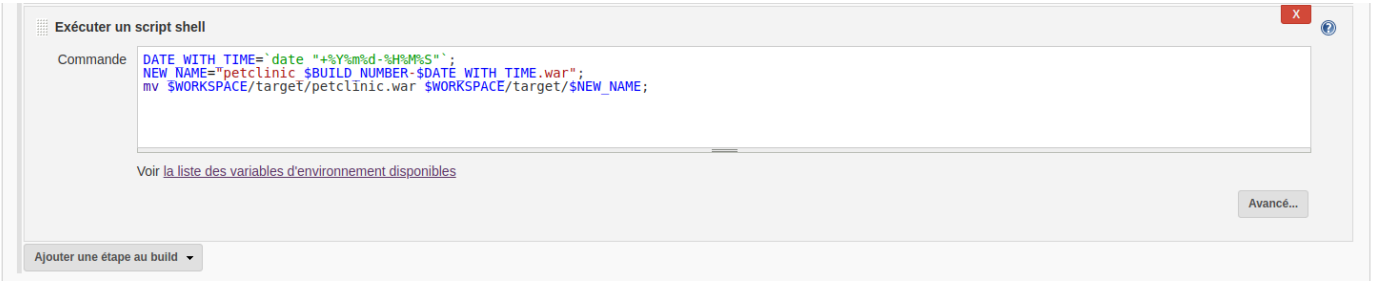
7. Consulter le log du build et l'espace de travail



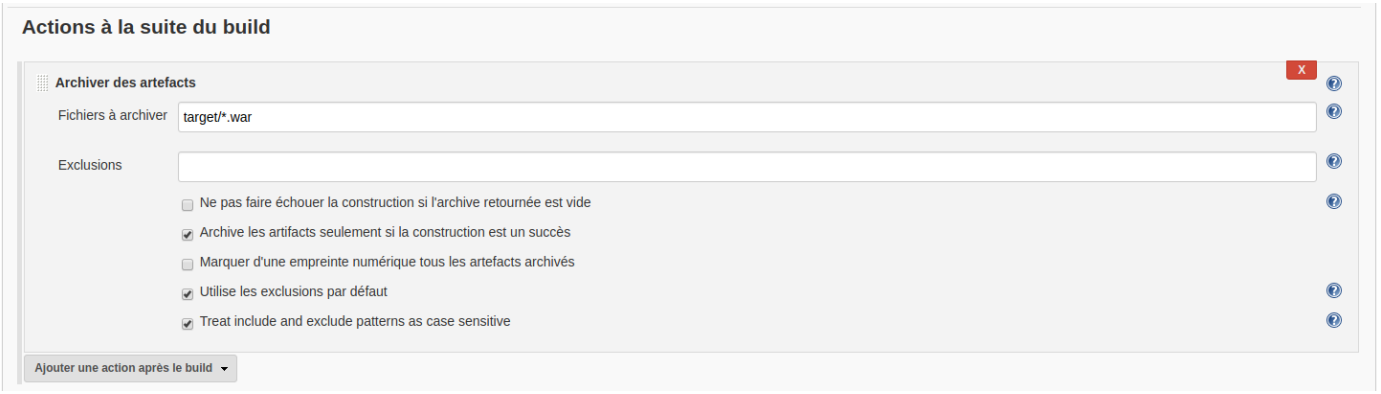


8. Ajouter une nouvelle tâche dans le build pour renommer le fichier **petclinic.war** généré dans le dossier target du workspace.
   Le fichier sera renommé de la façon suivante : petclinic-NUM_BUILD-TIMESTAMP.war
   Exemple : petclinic-4-20191210102322.war

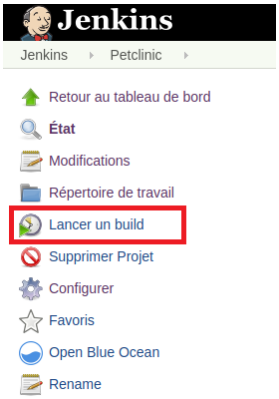
> Astuce :
> DATE_WITH_TIME=`date "+%Y%m%d-%H%M%S"`;
> NEW_NAME="petclinic_$BUILD_NUMBER-$DATE_WITH_TIME.war";



9. Ajouter une action à la suite du build pour archiver l'artifact généré (fichier à archiver : **target/*.war**)



10. Lancer le build manuellement



12. (En option) Ajouter une action pour supprimer le workspace une fois le build terminé