TP2 – Création/configuration projet Maven/Jenkins

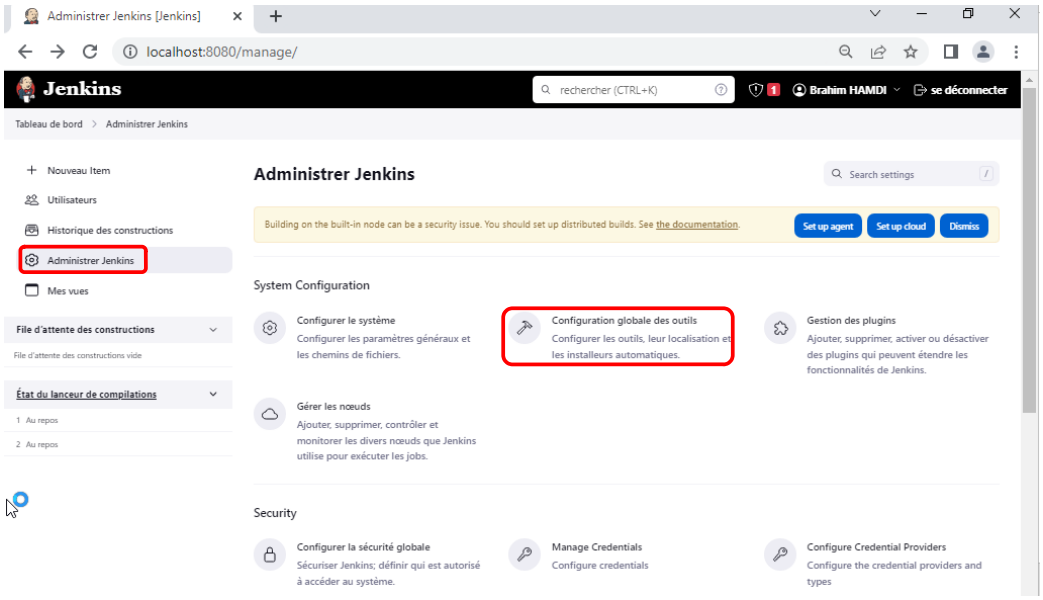
Brahim HAMDI

Objectifs :

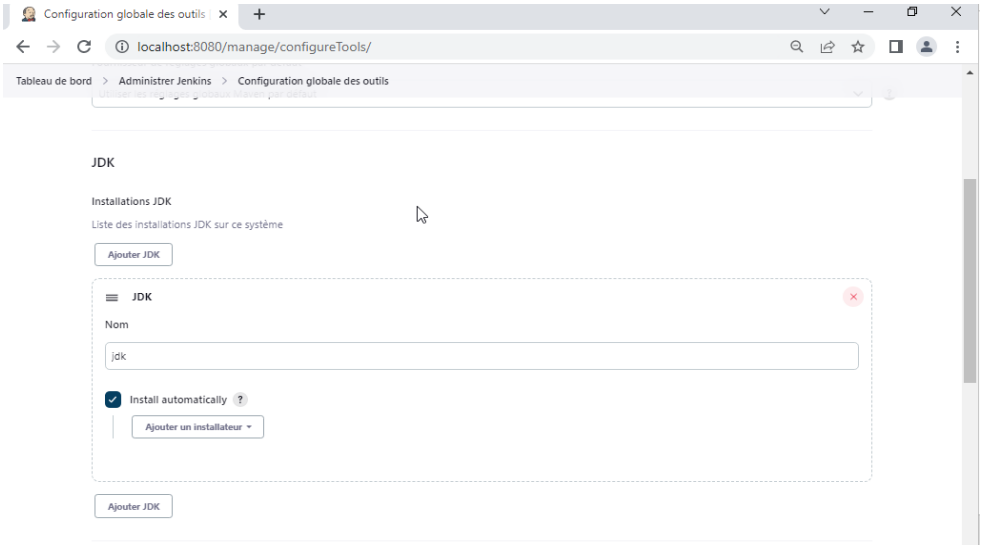
- Créer et configurer un projet java free-style
- Savoir écrire un pipeline déclaratif

Installation/Configuration de JDK et Maven

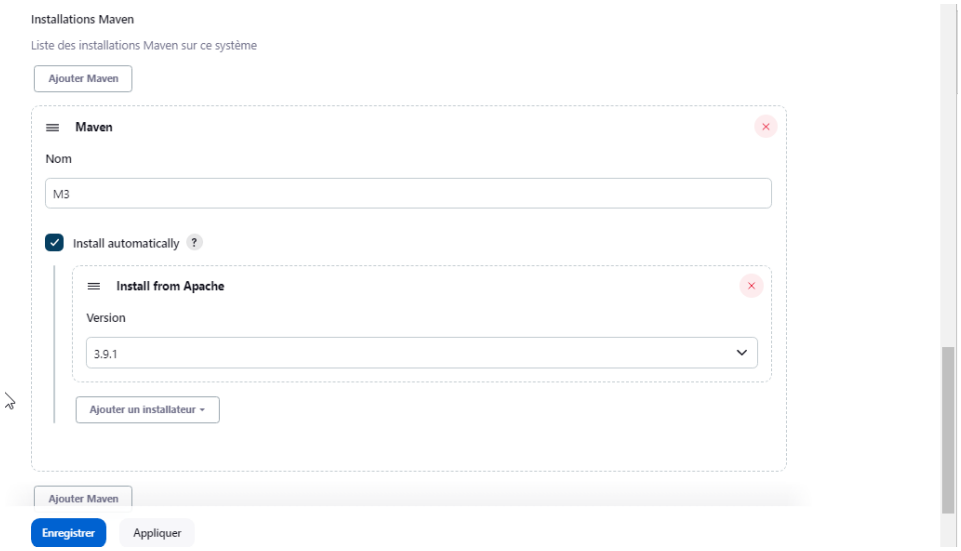
1. Connectez-vous à l'interface d'administration de Jenkins via votre navigateur web : http://192.168.56.120:8080
   - Naviguez vers la vue "Administrer Jenkins > Configuration globale des outils" et cliquez sur le bouton "Ajouter JDK"



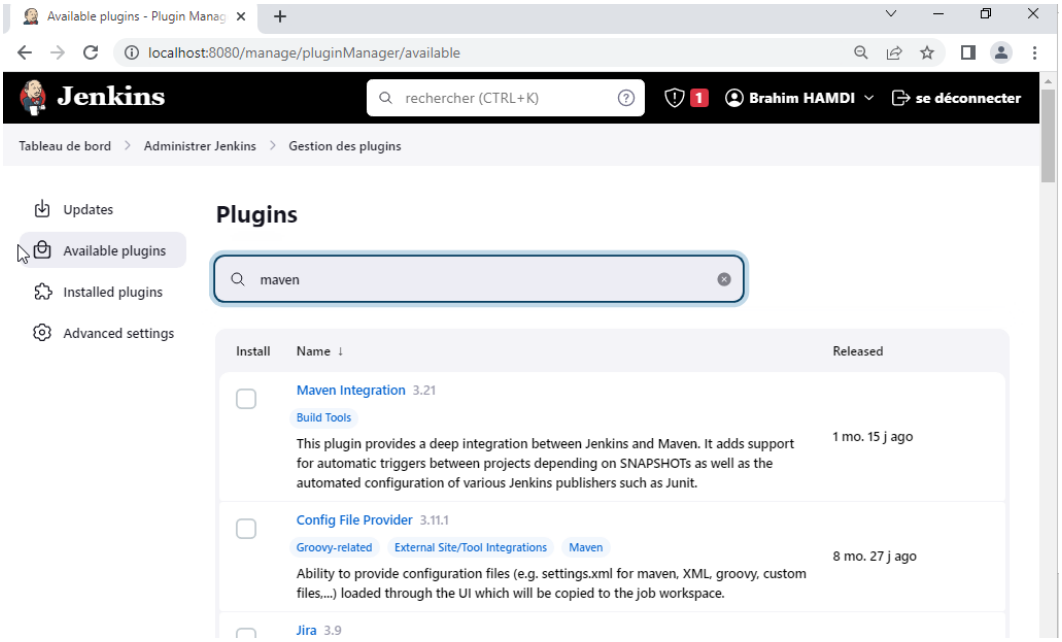
   - Dans la section de JDK, cliquez sur le bouton «Ajouter JDK », cochez la case «Installer automatiquement», puis saisir le nom et son chemin d'installation dans les champs "NOM" et cliquez sur le bouton "Appliquer".



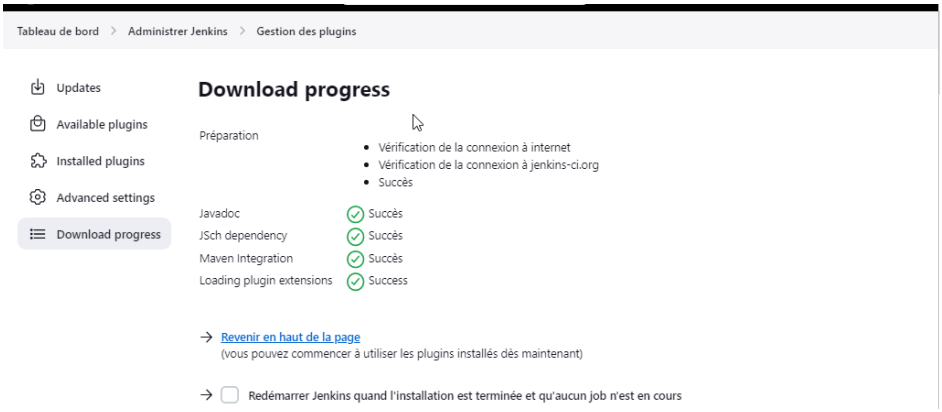
2. Dans la vue "Administrer Jenkins > Configuration globale des outils", cliquez sur le bouton "Ajouter Maven".
    - Saisir le nom de Maven dans le champs "NOM" (M3 par exemple) et cliquez ensuite sur le bouton "Enregistrer".



3. Dans la vue "Administrer Jenkins > Gestion des plugins" et cliquez sur l'onglet "Disponibles" et filtrez avec le terme "Maven integration".



- Sélectionnez l'option "Maven integration plugin" et cliquez sur le bouton "Installer sans redémarrer".

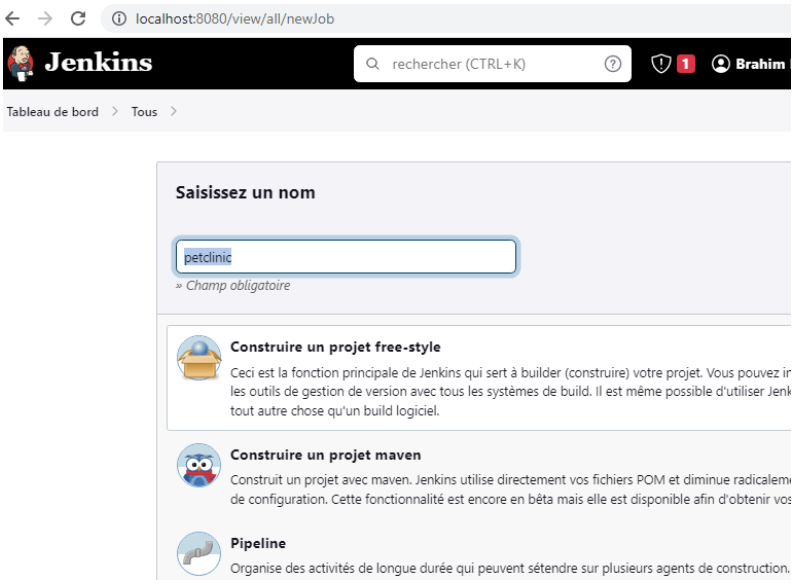


Création et exécution d'un job de build

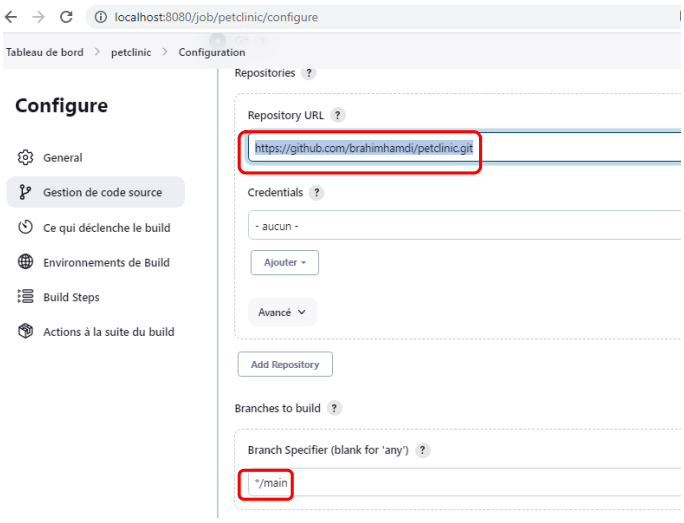
Dans cette partie, nous allons créer et configurer un projet de build de notre application « ***petclinic*** » (https://github.com/brahimhamdi/petclinic).

4. Dans le tableau de bord, cliquez sur ***Nouveau Item***. Une nouvelle fenêtre est apparue.

- Sur l'interface de la nouvelle fenêtre, saisir le nom de projet et sélectionner « ***Construire un projet free-style*** » et cliquez ensuite sur le bouton ***OK*** :

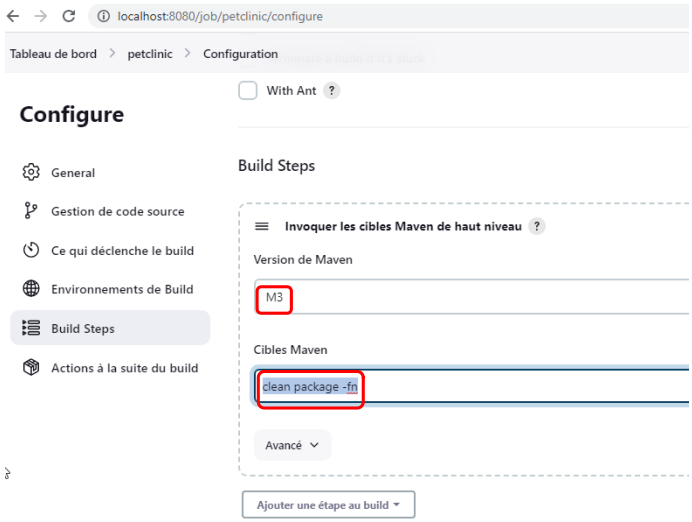


5. Le nouveau projet est créé, et son interface de configuration est affichée.
   - Dans la section ***Gestion de code source***, sélectionnez ***Git*** et tapez l'URL du dépôt Git où se trouve le code source de notre projet et changez la branche vers main (par défaut master), ensuite cliquez sur le bouton « ***Apply*** » :

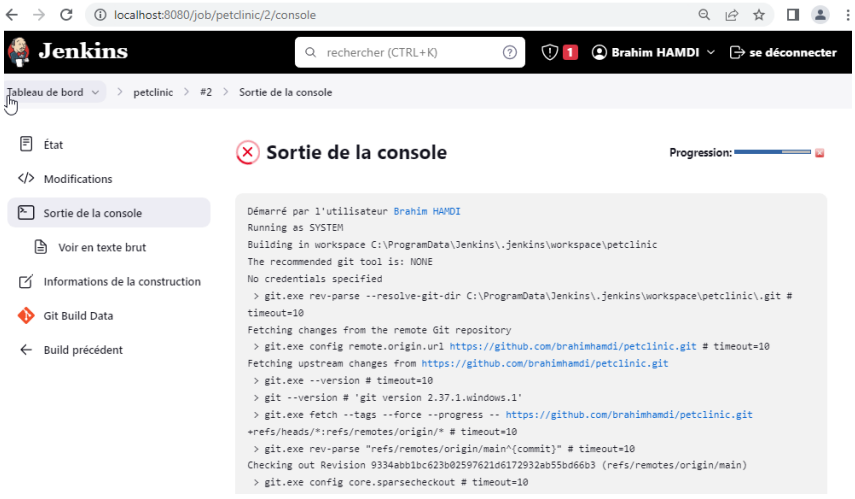


6. Dans la section **Build Steps**, nous ajouterons les étapes de build.
   - Sélectionnez « ***Invoker les cibles Maven de haut niveau*** ».
   - Nous voulons que cette étape exécute un cycle de vie de build Maven jusqu'à la création de package jar.
     - Définir la Version Maven sur l' outil ***M3*** que nous avons configuré précédemment.
     - Définir le champ Cibles Maven sur : ***clean package -fn***

       L' option *-fn* indique à Maven de ne pas échouer si certains tests échouent.



7. Cliquez sur le bouton « ***Sauver*** » pour sauvegarder et quitter l'interface de configuration du projet.
   - Ensuite lancez l'exécution du projet manuellement en cliquant sur « ***Lancer un build*** » (à gauche).
   - Observez l'exécution sur la sortie de console du projet.



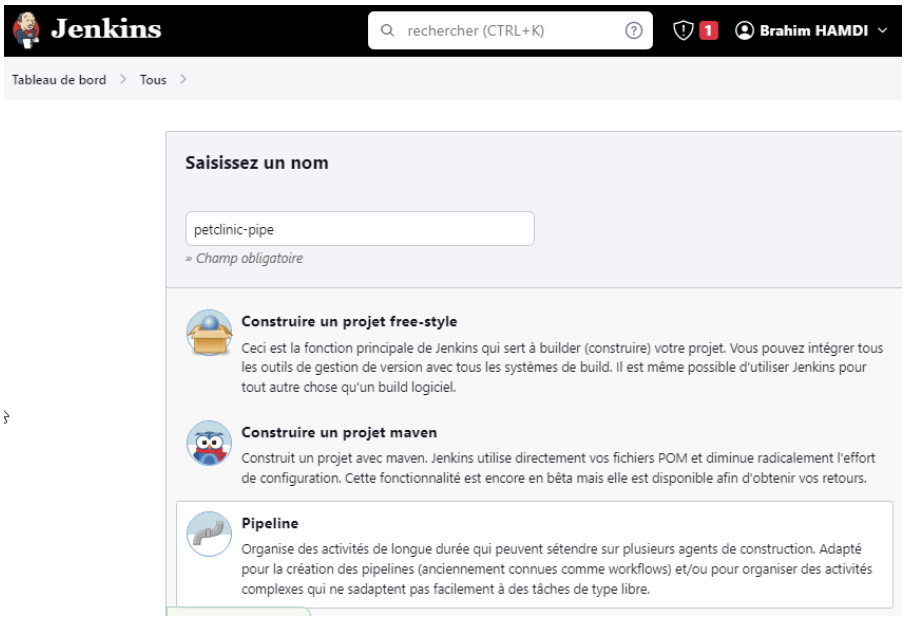
- A la fin, Jenkins affiche l'état de l'exécution du *build*. Dans notre cas, le build a passé avec succès.

```
[INFO] Building jar: C:\ProgramData\Jenkins\.jenkins\workspace\petclinic\target\spring-petclinic-
3.0.0-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.0.4:repackage (repackage) @ spring-petclinic ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  07:50 min
[INFO] Finished at: 2023-04-19T13:43:02+02:00
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

Création du pipeline déclaratif

Dans la partie précédente nous avons crée un projet d'intégration continue de type free-style. A partir de la version 2 de jenkins, il est possible de créer des pipeline sous forme d'un programme appelé *jenkinsfile*, en utilisant le langage *groovy* avec une syntaxe scriptée ou déclarative.
Dans cette partie, nous allons refaire le même travail mais avec un projet de type pipeline (le plugin *pipeline* doit être installé).

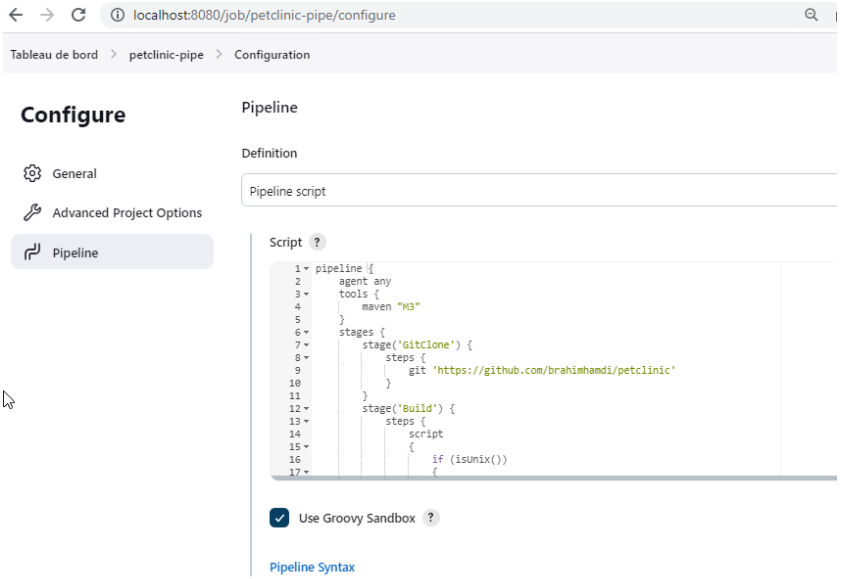
8. Sur le tableau de bord, cliquez sur Nouveau Item. Une nouvelle fenêtre est apparue.
   - Sur la nouvelle fenêtre, saisir le nom de projet, sélectionnez Pipeline et cliquez sur le bouton OK :



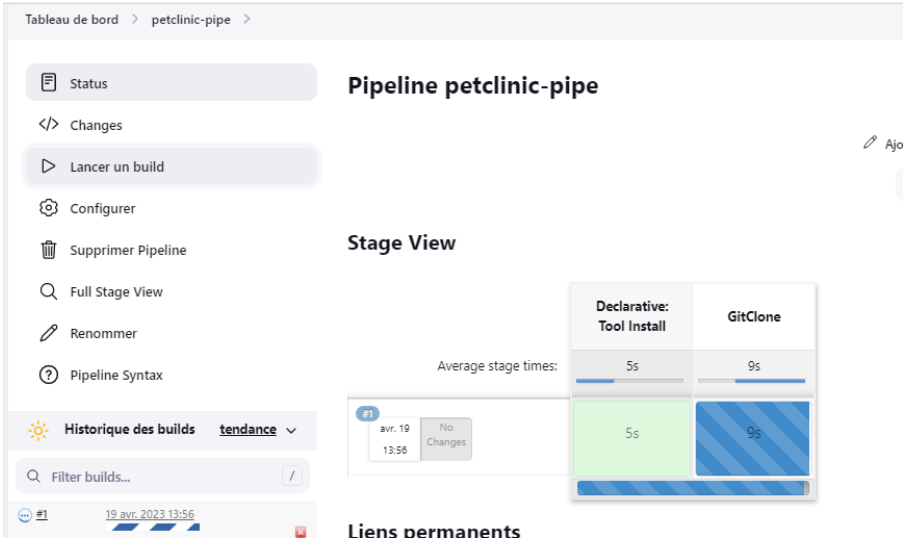
Le nouveau projet est créé, et son interface de configuration est affichée, elle est différente de celle du projet free-style.

9. Descendez vers la section Pipeline, et ajoutez le contenu suivant dans la partie script :

```
pipeline {
    agent any
    tools {
        maven "M3"
    }
    stages {
        stage('GitClone') {
            steps {
                git branch: 'main', url: 'https://github.com/brahimhamdi/petclinic'
            }
        }
        stage('Build') {
            steps {
                script
                {
                    if (isUnix())
                    {
                        sh 'mvn -fn clean package'
                    }
                    else
                    {
                        bat 'mvn -fn clean package'
                    }
                }
            }
        }
    }
}
```



10. Lorsque vous cliquez sur « Sauver », Jenkins sauvegarde le pipeline et affiche l'interface d'accueil du projet. Cliquez sur « *Lancer un build* » pour démarrer l'exécution du pipeline. Vous pouvez voir la progression du pipeline via l'interface de Jenkins :



Et voilà le résultat de build :

```
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ spring-petclinic ---
[INFO] Building jar: C:\ProgramData\Jenkins\.jenkins\workspace\petclinic-pipe\target\spring-petclinic-3.0.0-
SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.0.4:repackage (repackage) @ spring-petclinic ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  49.605 s
[INFO] Finished at: 2023-04-19T14:08:40+02:00
[INFO] ------------------------------------------------------------------------
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```