TP3 - pipeline jenkins

Installation de Jenkins

1. Téléchargez Jenkins depuis le site officiel :
   https://www.jenkins.io/download

2. Choisissez Windows Installer et Suivre le guide d'installation pour windows:
   https://www.jenkins.io/doc/book/installing/windows/

3. Laissez le port par défaut : 8080.

4. Jenkins s'installe comme un service Windows.

5. Une fois installé, ouvrez un navigateur et accédez à :

   http://localhost:8080

6. Récupérer le mot de passe initial dans :

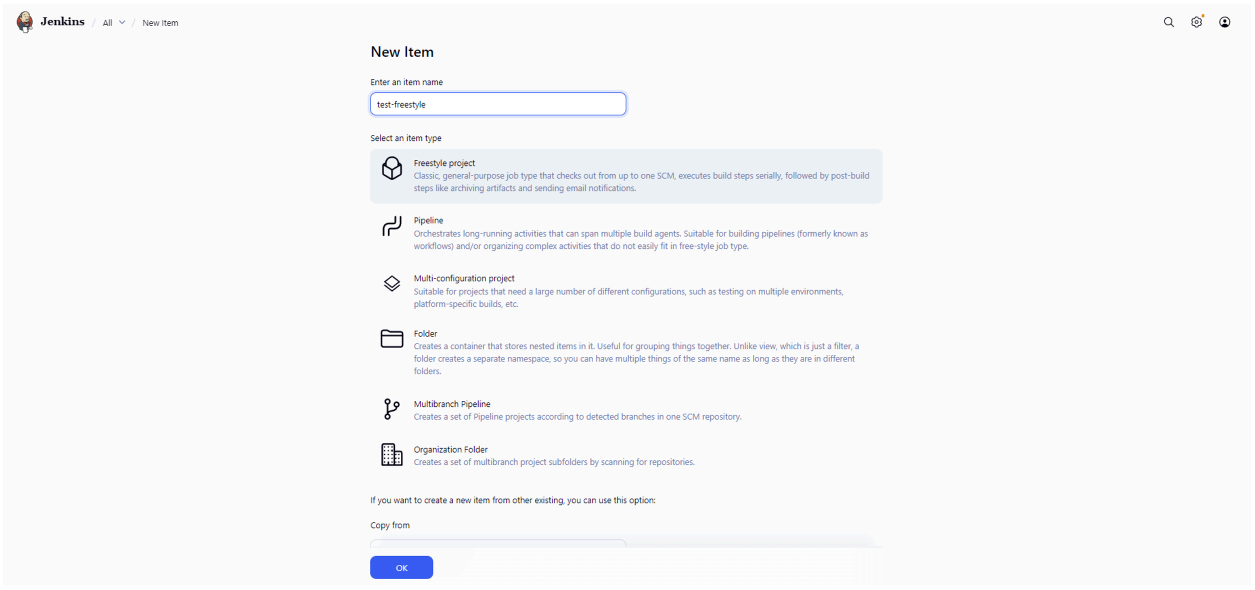
   C:\Program Files\Jenkins\secrets\initialAdminPassword

7. Collez la clé dans l'interface Jenkins pour déverrouiller l'installation.
8. Installer les plugins recommandés par défaut.

Exercice 1

Création d'un job Freestyle

1. Ouvrir Jenkins: http://localhost:8080.
2. Cliquer sur "New Item".
3. Entrer un nom, par exemple:
   test-freestyle
4. Choisir le type "Freestyle project".
5. Cliquer sur OK.



New Item

Enter an item name

test-freestyle

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

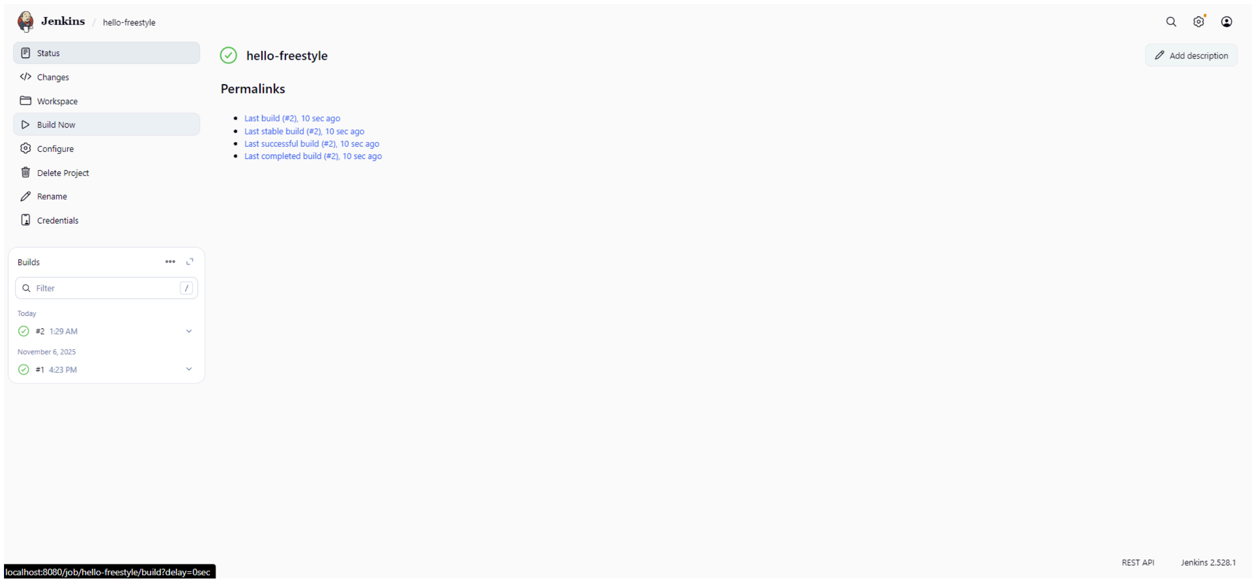
If you want to create a new item from other existing, you can use this option:

Copy from

6. Dans la page de configuration du job:
7. Descendre vers la section Build Steps.
8. Cliquer sur “Add build step”.
9. Sur Windows, choisir:

   “Execute Windows batch command”
10. Dans le champ commande, écrire:

```
echo Hello Jenkins depuis un job Freestyle
echo Date:
echo %DATE% %TIME%
```
11. Cliquer sur Save.
12. Lancer le job manuellement en cliquant sur “Build Now”.



hello-freestyle

Permalinks

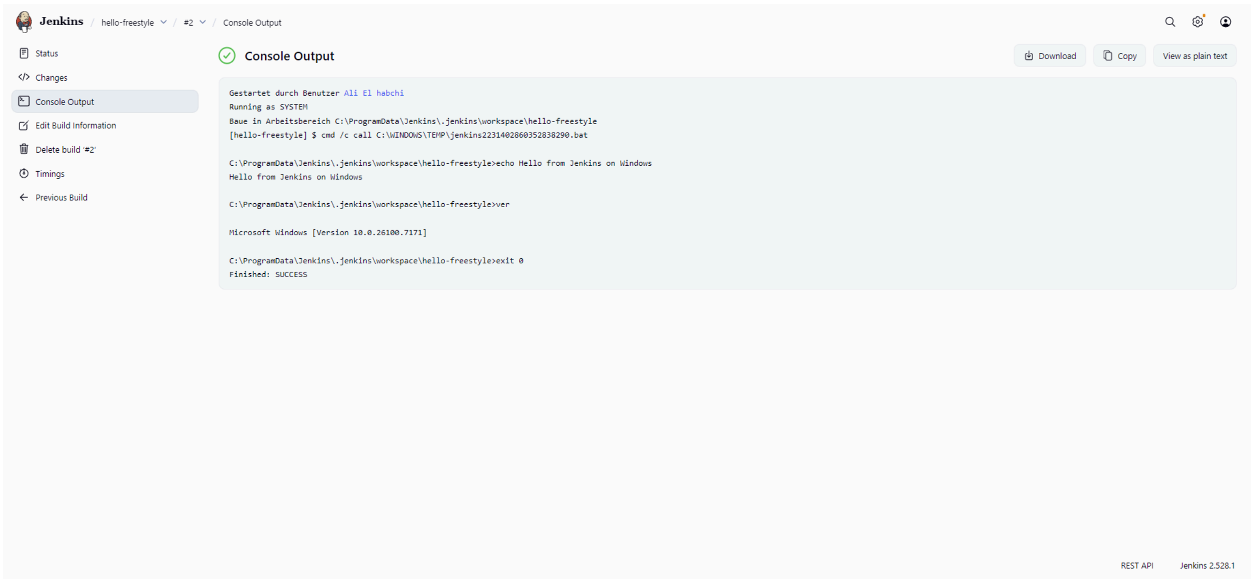
- Last build (#2), 10 sec ago
- Last stable build (#2), 10 sec ago
- Last successful build (#2), 10 sec ago
- Last completed build (#2), 10 sec ago

Today
#2 1:29 AM
November 6, 2025
#1 4:23 PM

localhost:8080/job/hello-freestyle/build?delay=0sec

13. Cliquer sur le build #1, puis sur “Console Output”.

14. Vérifier que la sortie affiche bien les messages:



```
Gestartet durch Benutzer Ali El habchi
Running as SYSTEM
Baue in Arbeitsbereich C:\ProgramData\Jenkins\.jenkins\workspace\hello-freestyle
[hello-freestyle] $ cmd /c call C:\WINDOWS\TEMP\jenkins2231402860352838290.bat

C:\ProgramData\Jenkins\.jenkins\workspace\hello-freestyle>echo Hello from Jenkins on Windows
Hello from Jenkins on Windows

C:\ProgramData\Jenkins\.jenkins\workspace\hello-freestyle>ver

Microsoft Windows [Version 10.0.26100.7171]

C:\ProgramData\Jenkins\.jenkins\workspace\hello-freestyle>exit 0
Finished: SUCCESS
```

15. Revenir dans la configuration du job
16. Cliquer sur “Configure”.
17. Aller à la section “Build Triggers”.
18. Cocher “Build periodically”.
19. Dans le champ Schedule, saisir par exemple:

* * * * * pour toutes les minutes

H/5 * * * * pour toutes les 5 minutes

20. Cliquer sur Save.

Exercice 2 - pipeline déclaratif simple

1. New Item -> Nom: test-pipeline -> Type: “Pipeline”. -> OK.
2. Section “Pipeline”
3. Definition est sur “Pipeline script”.
4. Dans le champ Script, coller:

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello depuis un pipeline déclaratif'
      }
    }
```

```
                stage('Infos environnement') {
                    steps {
                        echo "Nom du noeud Jenkins: ${env.NODE_NAME}"
                        echo "Workspace: ${env.WORKSPACE}"
                    }
                }
            }
        }
```

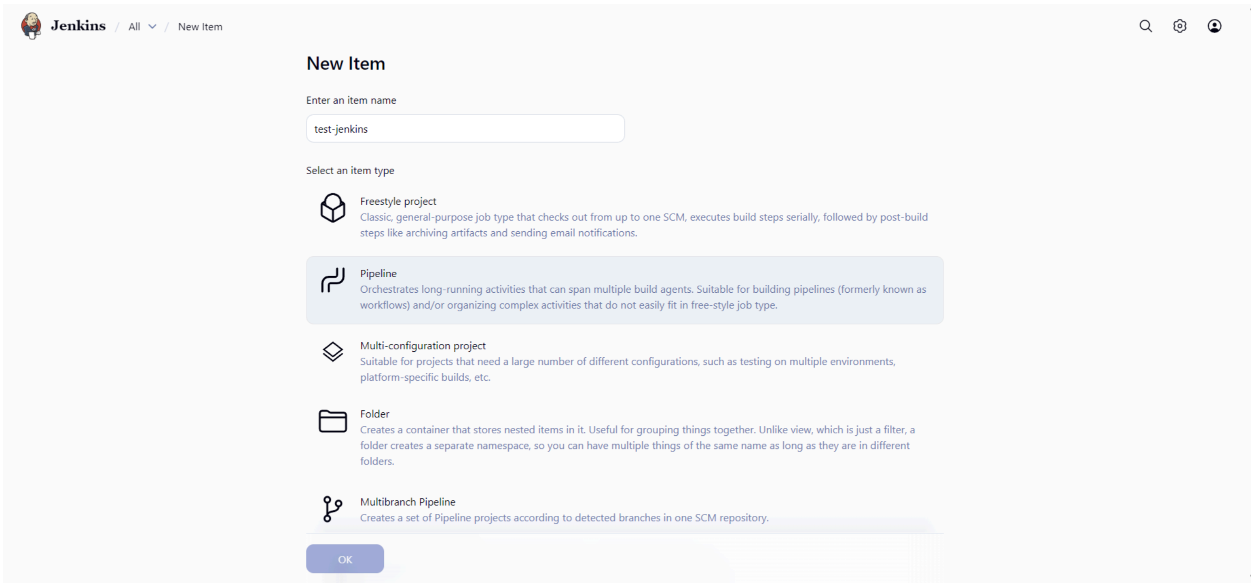
5. Build Now -> Console Output. Vérifier l'affichage des messages echo.

Exercice 3 - Pipeline déclaratif avec Jenkinsfile dans GitHub

- Créer un projet sur git avec un pipeline Jenkinsfile (utilisé même contenu de exercice 2)
- Commit et push:

```
git add Jenkinsfile README.md
git commit -m "Ajout Jenkinsfile simple"
git push origin main
```

- Créer un projet Jenkins :



New Item

Enter an item name

test-jenkins

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

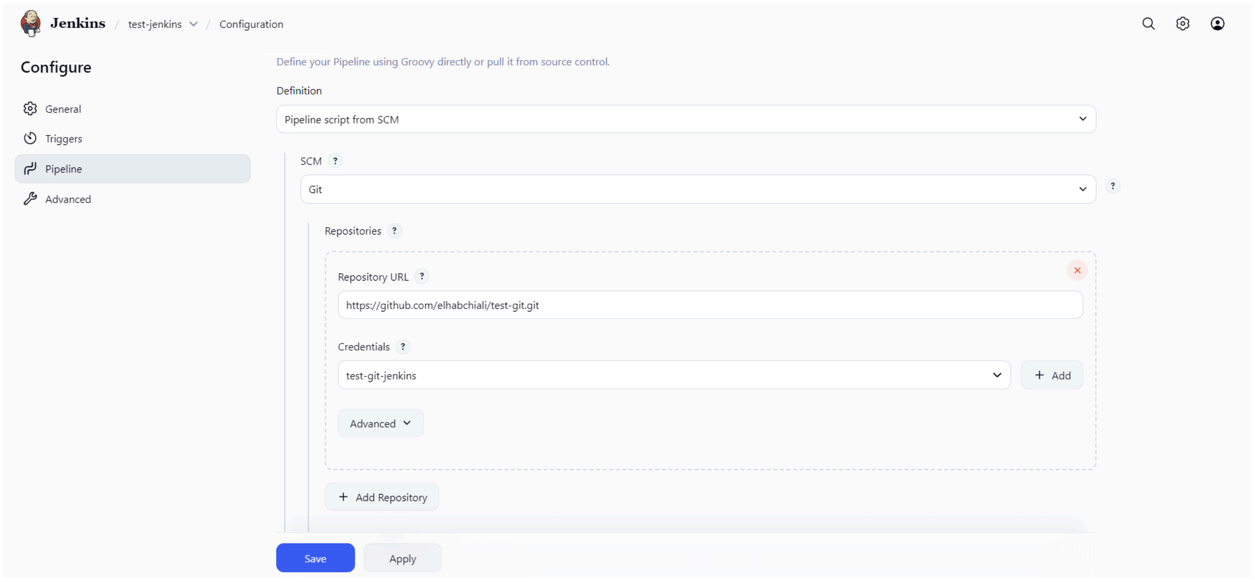
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

- New Item -> entrer un nom du projet (utiliser même nom du projet git pour simplifier)
- Choisir Pipeline



Define your Pipeline using Groovy directly or pull it from source control.

Definition: Pipeline script from SCM

SCM: Git

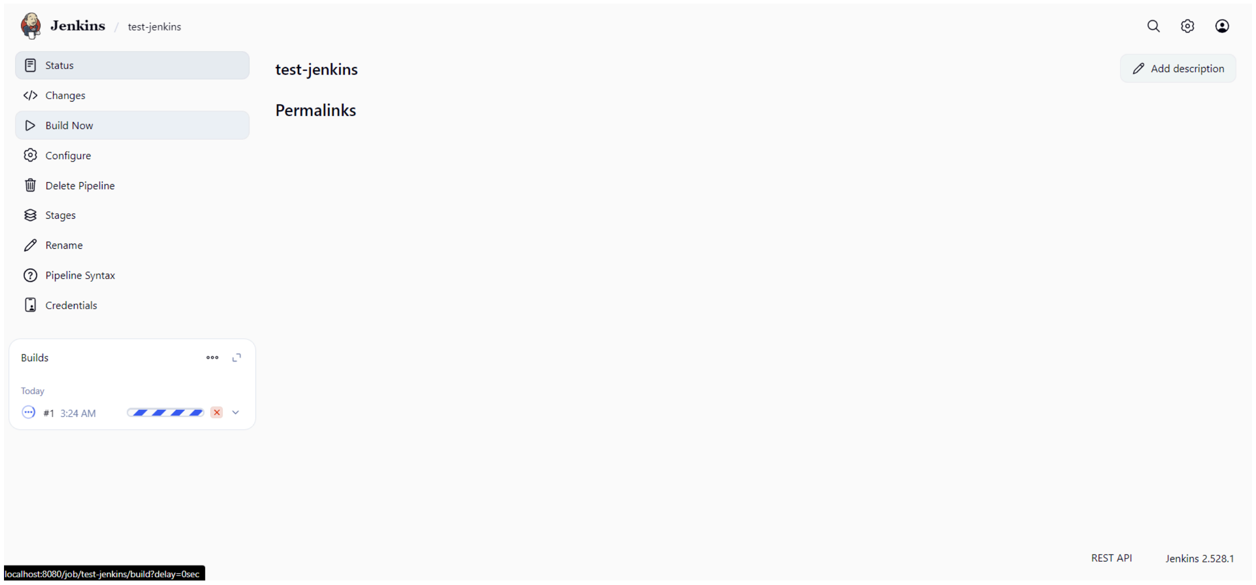
Repositories

Repository URL: https://github.com/elhabchiali/test-git.git

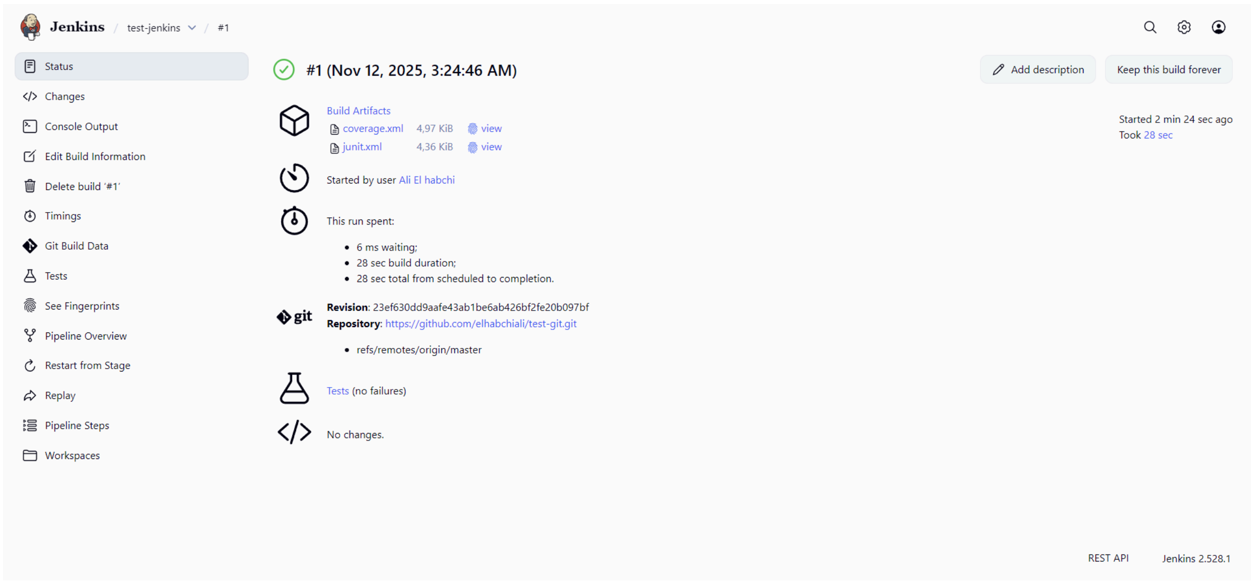
Credentials: test-git-jenkins

- Definition: Pipeline script from SCM.
- SCM: Git
- URL du repo + Credentials
- Script Path: Jenkinsfile.
- Save.

- Cliquer sur Build now pour déclencher le pipeline manuellement:



- Après terminaison cliquer sur le dernier Build et voir le résultat (Console log)



#1 (Nov 12, 2025, 3:24:46 AM)

Build Artifacts

coverage.xml	4,97 KiB	view
junit.xml	4,36 KiB	view

Started 2 min 24 sec ago
Took 28 sec

Started by user Ali El habchi

This run spent:

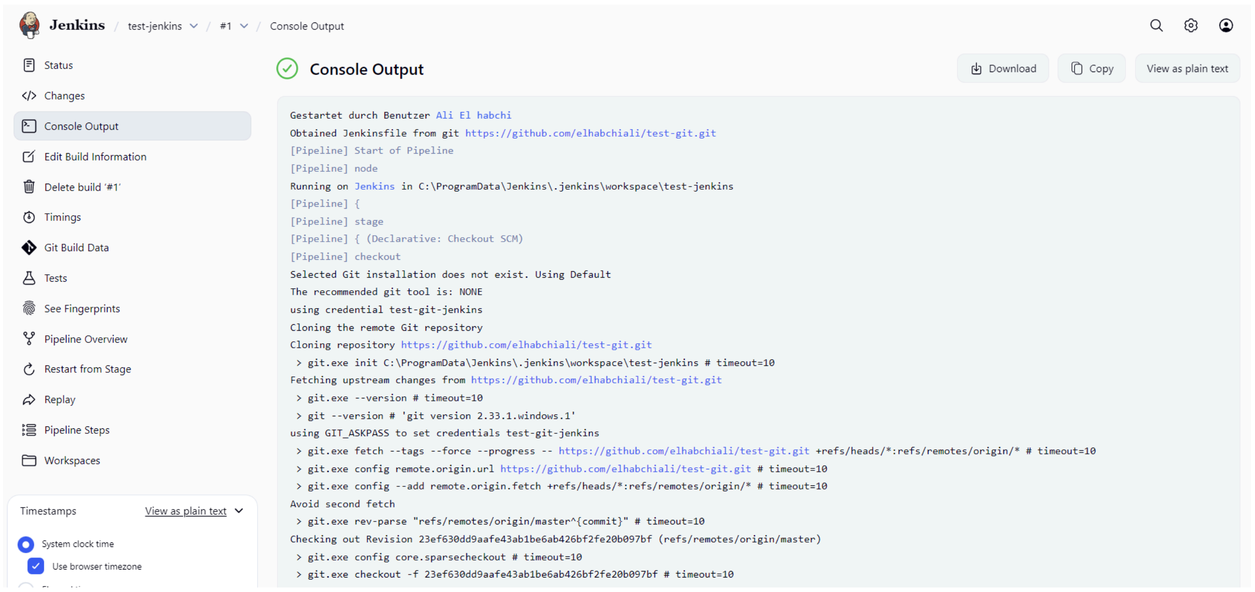
- 6 ms waiting;
- 28 sec build duration;
- 28 sec total from scheduled to completion.

Revision: 23ef630dd9aafe43ab1be6ab426bf2fe20b097bf
Repository: https://github.com/elhabchiali/test-git.git

- refs/remotes/origin/master

Tests (no failures)

No changes.



Console Output

```
Gestartet durch Benutzer Ali El habchi
Obtained Jenkinsfile from git https://github.com/elhabchiali/test-git.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\test-jenkins
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential test-git-jenkins
Cloning the remote Git repository
Cloning repository https://github.com/elhabchiali/test-git.git
 > git.exe init C:\ProgramData\Jenkins\.jenkins\workspace\test-jenkins # timeout=10
Fetching upstream changes from https://github.com/elhabchiali/test-git.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.33.1.windows.1'
using GIT_ASKPASS to set credentials test-git-jenkins
 > git.exe fetch --tags --force --progress -- https://github.com/elhabchiali/test-git.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe config remote.origin.url https://github.com/elhabchiali/test-git.git # timeout=10
 > git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 23ef630dd9aafe43ab1be6ab426bf2fe20b097bf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 23ef630dd9aafe43ab1be6ab426bf2fe20b097bf # timeout=10
```

- dans Build Triggers:
- Cocher “Poll SCM”.
- Indiquer une expression de type cron, par exemple:

  H/2 * * * *

- Toutes les 2 minutes, Jenkins va vérifier le dépôt Git.
- Faire une modification sur git pour déclencher un autre build.
- Après dans le projet git ajouter du code python et des fichiers test, vous pouvez utiliser le projet du tp2.
- Sur le pipeline Jenkinsfile ajouter un stage pour l’installation du projet en utilisant poetry et un autre pour lancement de test.