Lab 1 – Introduction à JENKINS

Jenkins est un outil d'automatisation essentiel pour configurer l'intégration continue. C'est l'intégrateur qui vous aide à créer votre flux de travail de développement, de test et de déploiement et à créer des pipelines de tâches. Il ajoute également de la visibilité à toutes les parties prenantes, y compris les équipes de développement, d'assurance qualité et d'exploitation impliquées dans la création, les tests et le déploiement du produit.

Démarche officielle : https://www.jenkins.io/doc/book/installing/linux/

Vous pouvez choisir entre la version Jenkins à support à long terme et la version Jenkins hebdomadaire.

Version de support à long terme

Une version LTS (Long-Term Support) est sélectionnée toutes les 12 semaines parmi les versions régulières comme version stable pour cette période. Elle peut être installée depuis le dépôt yum redhat-stable.

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
    https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade
# Add required dependencies for the jenkins package
sudo yum install fontconfig java-21-openjdk
sudo yum install jenkins
sudo systemctl daemon-reload
```

Démarrer Jenkins

Vous pouvez activer et démarrer le service Jenkins pour qu'il démarre au démarrage du système avec la commande :

sudo systemctl enable --now jenkins

Vous pouvez vérifier l'état du service Jenkins à l'aide de la commande :

sudo systemctl status jenkins

Si tout a été correctement configuré, vous devriez obtenir un résultat similaire à celui-ci :

```
jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled;
preset: disabled)
     Active: active (running) since Sun 2025-11-02 16:59:13 CET; 14s ago
…
     CGroup: /system.slice/jenkins.service
             └─4200 /usr/bin/java -Djava.awt.headless=true -jar
/usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --
httpPort=8080
```

```
Nov 02 16:59:07 master01.labs.local jenkins[4200]: [LF]> This may also be
found at: /var/lib/jenkins/secrets/initialAdminPassword...
```

NB :

Si un pare-feu est installé, vous devez ajouter Jenkins aux exceptions. Modifiez YOURPORTle script ci-dessous pour indiquer le port souhaité. Le port 800 8080est le plus courant.

```
YOURPORT=8080
PERM="--permanent"
SERV="$PERM --service=jenkins"

firewall-cmd $PERM --new-service=jenkins
firewall-cmd $SERV --set-short="Jenkins ports"
firewall-cmd $SERV --set-description="Jenkins port exceptions"
firewall-cmd $SERV --add-port=$YOURPORT/tcp
firewall-cmd $PERM --add-service=jenkins
firewall-cmd --zone=public --add-service=http --permanent
firewall-cmd --reload
```

Assistant de configuration post-installation

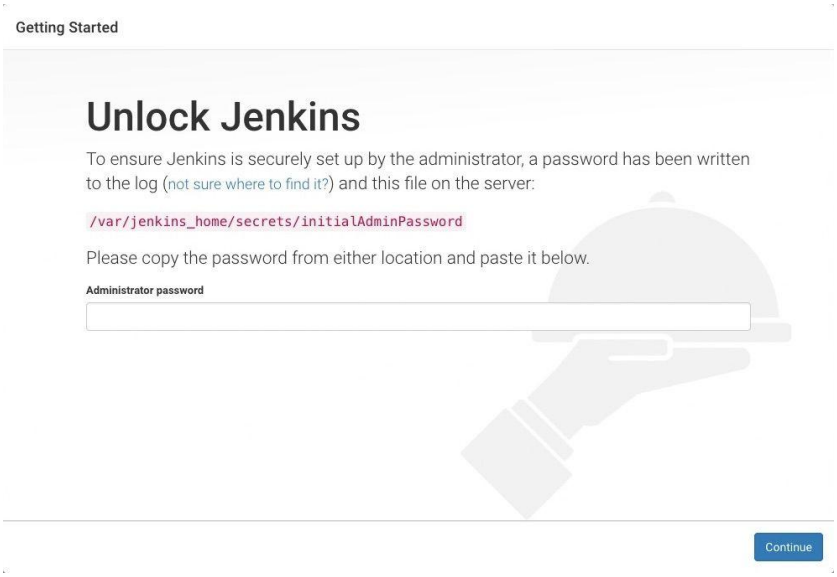
Après avoir téléchargé, installé et exécuté Jenkins en utilisant l'une des procédures ci-dessus (à l'exception de l'installation avec Jenkins Operator), l'assistant de configuration post-installation démarre.

Cet assistant d'installation vous guide à travers quelques étapes rapides et ponctuelles pour déverrouiller Jenkins, le personnaliser avec des plugins et créer le premier utilisateur administrateur grâce auquel vous pourrez continuer à accéder à Jenkins.

Déverrouillage de Jenkins

Lors de votre première connexion à un nouveau contrôleur Jenkins, il vous est demandé de le déverrouiller à l'aide d'un mot de passe généré automatiquement.

1. Accédez à http://master01.lab.local:8080 et attendez que la page **Déverrouiller Jenkins** apparaisse.



2. La commande sudo cat /var/lib/jenkins/secrets/initialAdminPassword affichera le mot de passe dans la console

```
user@master01:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
6fe136853abb419eb0a1ad19abb52602
user@master01:~$
```

3. Sur la page **Déverrouiller Jenkins** , collez ce mot de passe dans le champ **Mot de passe administrateur et cliquez sur Continuer** .
   Remarque :
   - Le journal de la console Jenkins indique l'emplacement (dans le répertoire d'installation de Jenkins) où ce mot de passe est également disponible. Ce mot de passe doit être saisi dans l'assistant d'installation lors des nouvelles installations de Jenkins avant de pouvoir accéder à l'interface utilisateur principale. Ce mot de passe sert également de mot de passe par défaut pour le compte administrateur (nom d'utilisateur : « admin ») si vous omettez l'étape suivante de création d'utilisateur dans l'assistant d'installation.

Personnalisation de Jenkins avec des plugins

Une fois Jenkins déverrouillé, la page **« Personnaliser Jenkins »** s'affiche. Vous pouvez y installer autant de plugins utiles que vous le souhaitez lors de votre configuration initiale.

Cliquez sur l'une des deux options affichées :

- **Installer les plugins suggérés** - pour installer l'ensemble de plugins recommandé, basé sur les cas d'utilisation les plus courants.
- **Sélectionnez les plugins à installer** : choisissez l'ensemble de plugins à installer initialement. Lors de votre première visite sur la page de sélection des plugins, les plugins suggérés sont sélectionnés par défaut.

Choisissez **« Installer les plugins suggérés »** . Vous pourrez installer (ou supprimer) des plugins Jenkins supplémentaires ultérieurement via la page **« Gérer Jenkins »** > **« Plugins »** dans Jenkins.

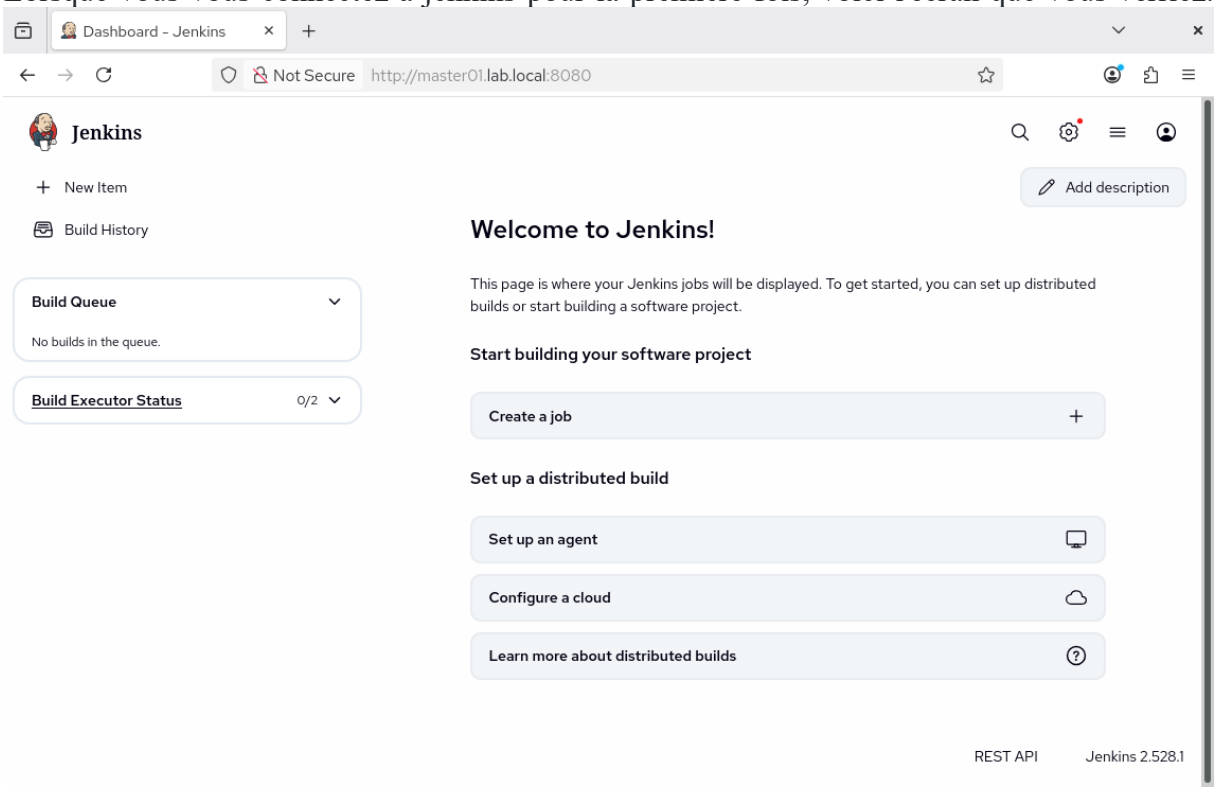
Création du premier utilisateur administrateur

Enfin, après avoir personnalisé Jenkins avec des plugins , Jenkins vous demande de créer votre premier utilisateur administrateur.

1. Lorsque la page **« Créer le premier utilisateur admininstrateur »** apparaît, spécifiez les détails de votre utilisateur administrateur dans les champs correspondants « admin/password » et cliquez sur « **Enregistrer et terminer »** .
2. Lorsque la page **« Jenkins est prêt » s'affiche, cliquez sur « Démarrer l'utilisation de Jenkins »** .
   Si nécessaire, connectez-vous à Jenkins avec les identifiants de l'utilisateur que vous venez de créer et vous êtes prêt à commencer à utiliser Jenkins !

Se familiariser avec la console Jenkins

Lorsque vous vous connectez à jenkins pour la première fois, voici l'écran que vous verriez.



- Sur le côté gauche de l'écran, en haut se trouve le menu pour créer de nouveaux projets, pour gérer jenkins, pour créer des utilisateurs, etc.
- Juste en dessous du menu se trouve la file d'attente de construction. Tous les travaux programmés pour s'exécuter sont ajoutés à la file d'attente et apparaîtront ici.
- Sous la file d'attente de construction se trouve le statut de l'exécuteur de compilation. Cela montre l'état des travaux en cours d'exécution en temps réel.
- En bas à droite de la page se trouvent les informations sur la version de jenkins affichées.

Installation d'un plugin

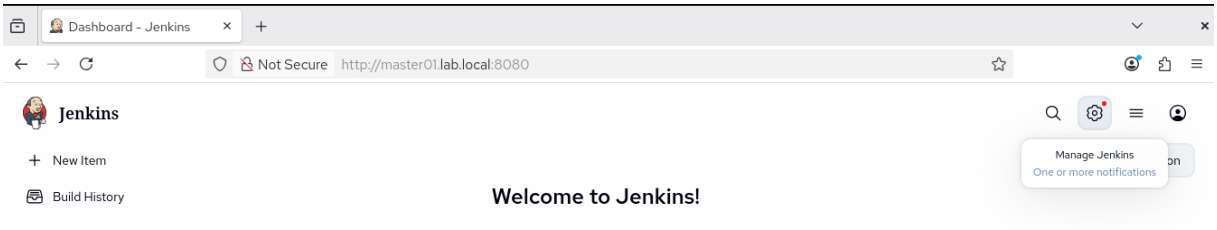
Jenkins propose deux méthodes pour installer des plugins sur le contrôleur :

1. Utilisation du « Gestionnaire de plugins » dans l'interface utilisateur Web.
2. Utilisation de la commande CLI Jenkins install-plugin .

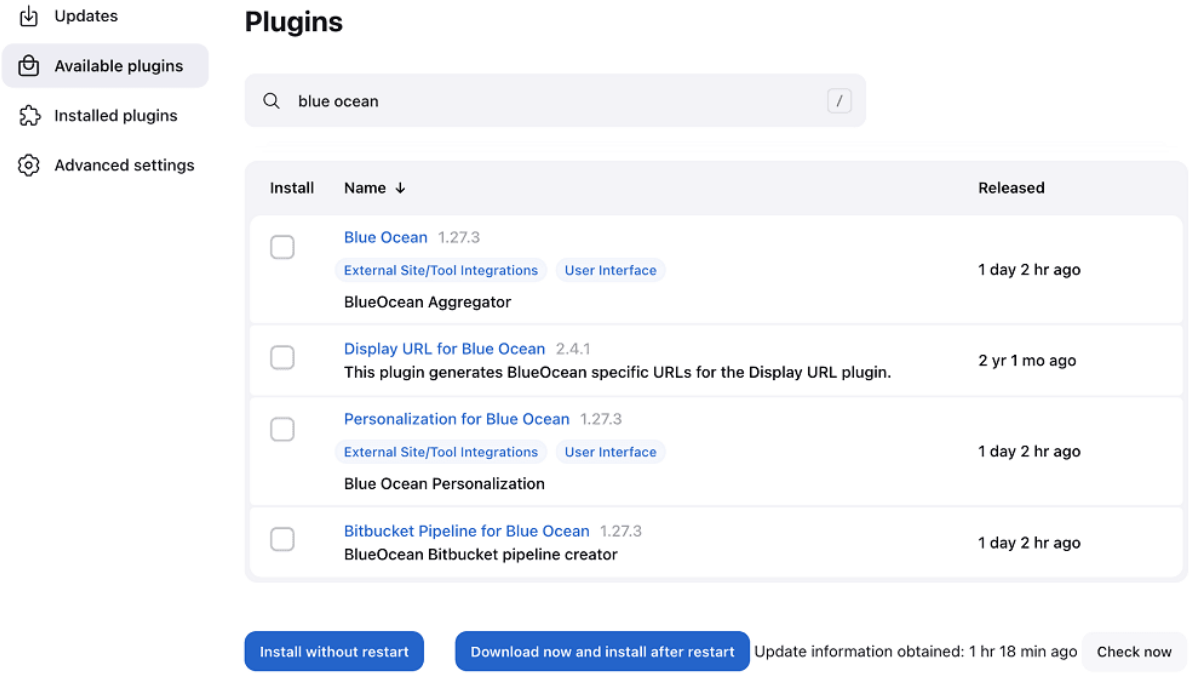
Les deux approches nécessitent que le contrôleur Jenkins soit capable de télécharger des métadonnées à partir d'un centre de mise à jour, qu'il s'agisse du centre de mise à jour principal exploité par le projet Jenkins [1] ou d'un centre de mise à jour personnalisé.

Depuis l'interface utilisateur Web

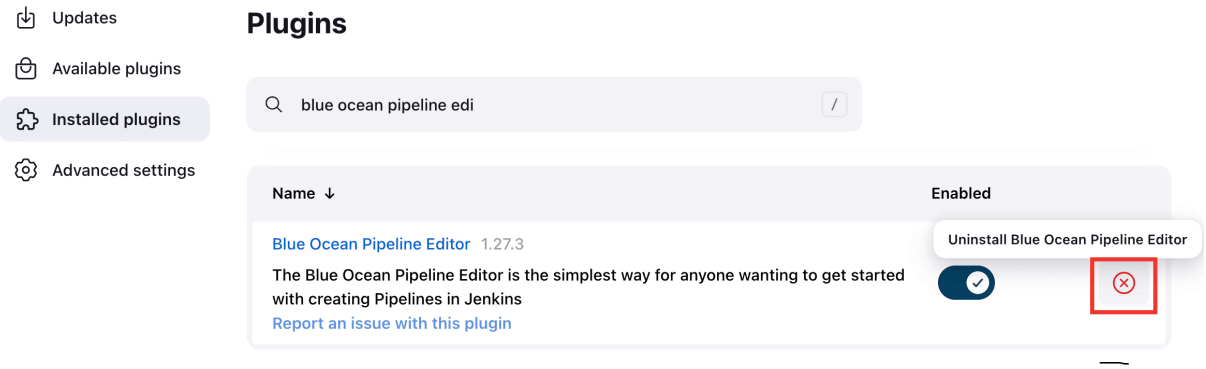
La méthode la plus simple et la plus courante pour installer des plugins consiste à utiliser la vue **Gérer Jenkins** > **Plugins** , accessible aux administrateurs d'un environnement Jenkins.



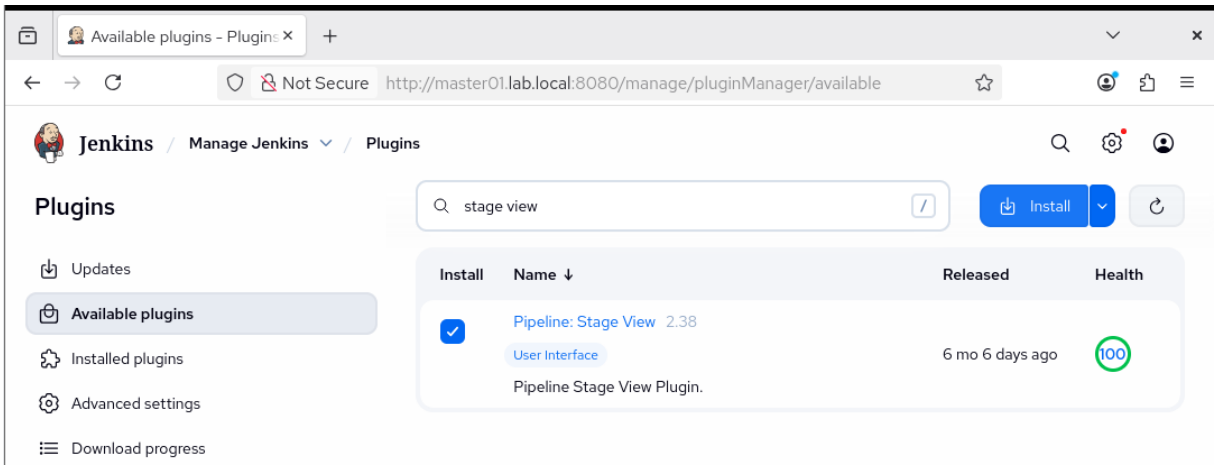
Sous l' onglet **Disponible** , les plugins disponibles au téléchargement depuis le Centre de mise à jour configuré peuvent être recherchés et pris en compte :



La plupart des plugins peuvent être installés/désinstallés et utilisés immédiatement en cochant la case située à côté du plugin et en cliquant sur **« Installer sans redémarrage »**.



Installez de même le plugin suivant :



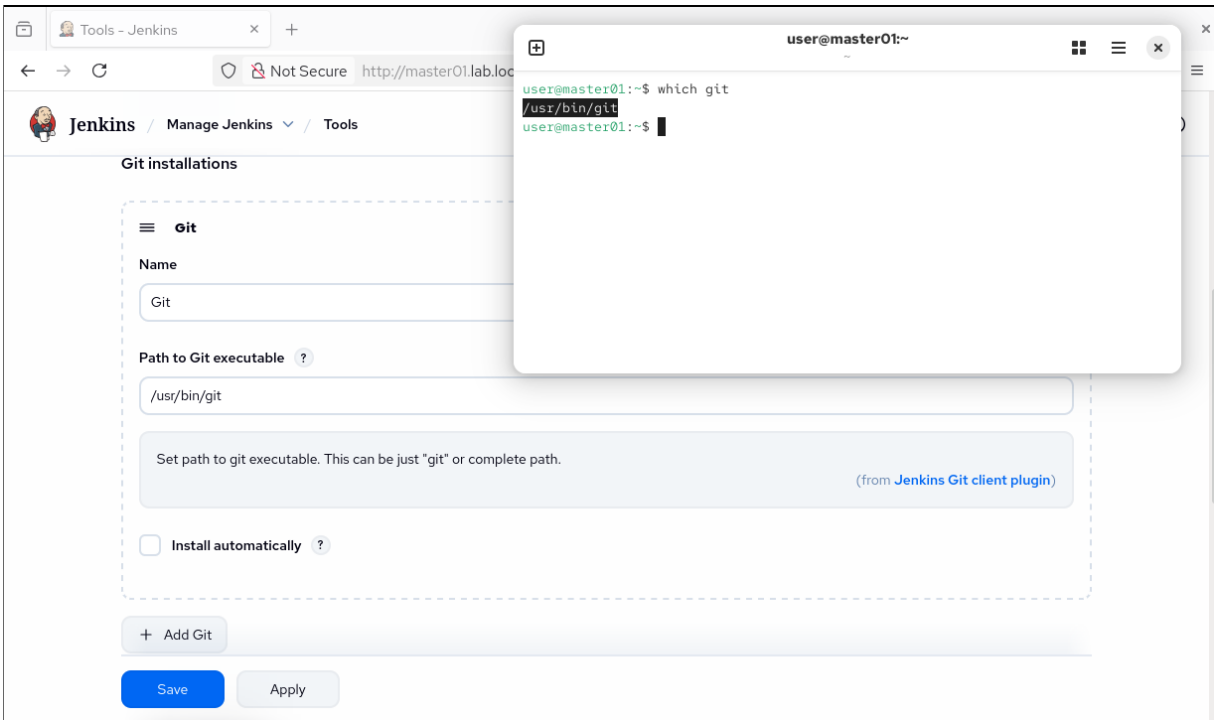
La section « Manage Jenkins > Tools »

Permet de configurer et de gérer les outils externes nécessaires à l'exécution des tâches de construction, de test et de déploiement, tels que les installations de Java (JDK), Git, Maven, etc.. Elle permet de spécifier les chemins d'accès à ces outils ou de laisser Jenkins les télécharger et les installer automatiquement, garantissant que Jenkins peut les utiliser pour exécuter les pipelines CI/CD.

Fonctions clés de "Manage Jenkins > Tools" :

- **Spécifier les chemins d'accès** : Vous pouvez indiquer manuellement le chemin d'installation de chaque outil, par exemple le répertoire d'installation de Java (JDK) ou le binaire de Git.
- **Installation automatique** : Il est également possible de configurer Jenkins pour qu'il télécharge et installe automatiquement les outils dont il a besoin (comme le JDK), simplifiant ainsi le processus d'installation.
- **Configuration des outils de build** : Cette section est utilisée pour configurer des outils comme Maven ou Gradle. Vous pouvez y définir leurs propriétés et leur emplacement.
- **Gestion des configurations globales** : Au-delà des outils, ce menu est souvent lié à la configuration système globale, permettant de définir des variables d'environnement, de configurer des notifications par e-mail, etc

Configurons l'outils Git :



Configurons JDK

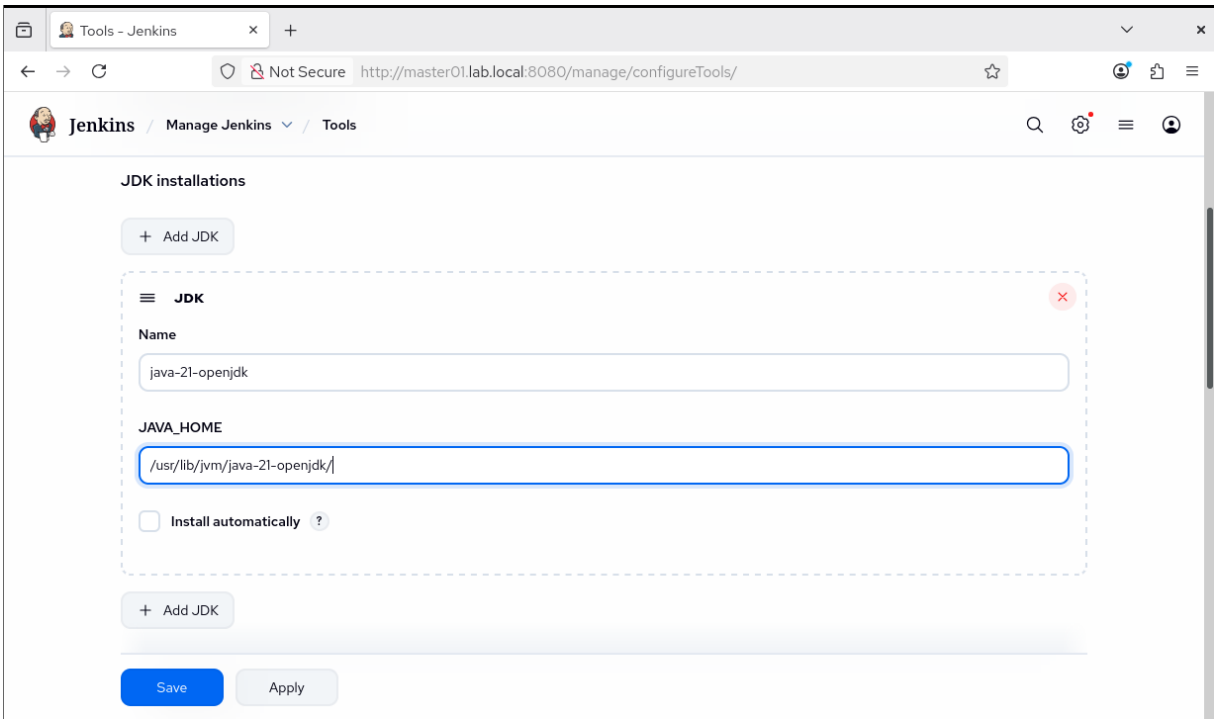
```
user@master01:~$ sudo dnf install java-21-openjdk-devel
[sudo] password for user:
Last metadata expiration check: 2:28:50 ago on Sun 02 Nov 2025 05:46:43 PM CET.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                          Repository  Size
================================================================================
Installing:
 java-21-openjdk-devel   x86_64        1:21.0.9.0.10-1.el10.alma.1      appstream   5.0 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 5.0 M
Installed size: 11 M
Is this ok [y/N]: y
```

```
$ update-alternatives --display javac
```

```
user@master01:~$ update-alternatives --display javac
javac - status is auto.
 link currently points to /usr/lib/jvm/java-21-openjdk/bin/javac
/usr/lib/jvm/java-21-openjdk/bin/javac - priority 21000910
 follower java_sdk: /usr/lib/jvm/java-21-openjdk
```



Création du premier projet

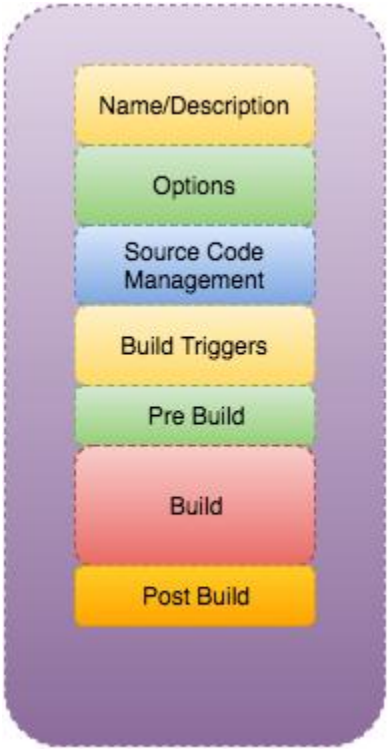
Dans cette session, nous allons créer et lancer notre premier projet avec jenkins. Nous utiliserons un projet de style libre pour cet exemple.

Types de Jobs

Avec Jenkins, vous pouvez créer les types de projets ou d'emplois suivants.

- Free Style
- Pipeline
- Maven
- Configuration multiple

Anatomie d'un Jenkins Job



Un travail jenkins de style typique comprend les sections suivantes.

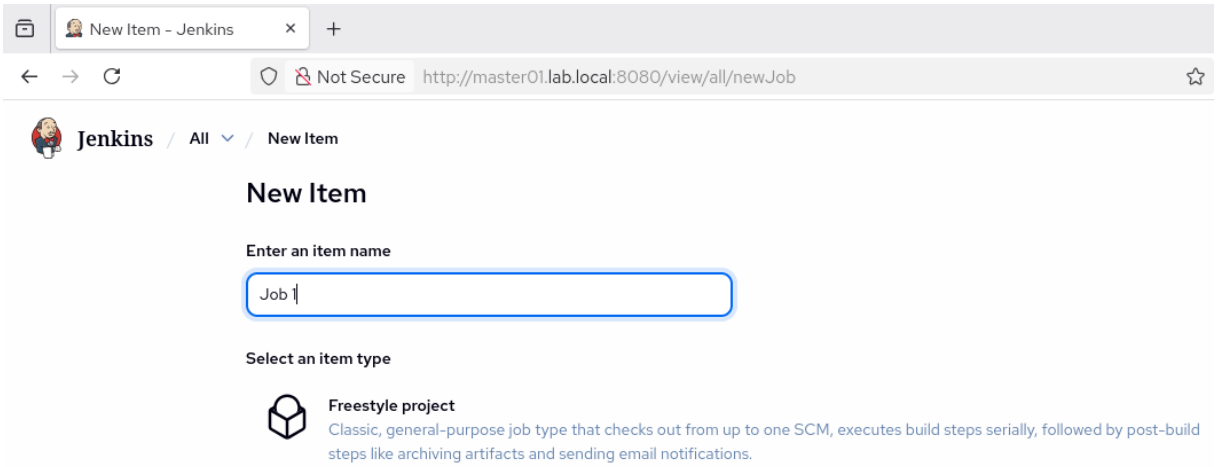
- Nom / description
- Option avancée
- Gestion du code source
- Créer des déclencheurs
- Pré-construction
- Construction
- Post construction

Créer un Job simple

Nous allons tester un job Java, pour cela nous avons déjà configuré le plugin JDK de jenkins

On peut maintenant créer un Job simple en utilisant jenkins pour exécuter un programme hello world.

1. À partir de la page principale de Jenkins, cliquez sur **Nouvel élément**.
2. Donnez un nom au projet dans Nom de l'élément, c'est-à-dire "job1". Sélectionnez Freestyle Project.



L'écran suivant ouvre la page de configuration du travail. Sur la page de configuration du travail,

3. Ajoutez une description de job par exemple, "Notre premier Job Jenkins".

4. Ignorez la « gestion du code source » et « Ce qui déclenche le build », puis faites défiler jusqu'à **Build steps**.

5. Dans "Ajouter une étape de construction", sélectionnez **Exécuter un script Shell** et fournissez l'exemple de code suivant,

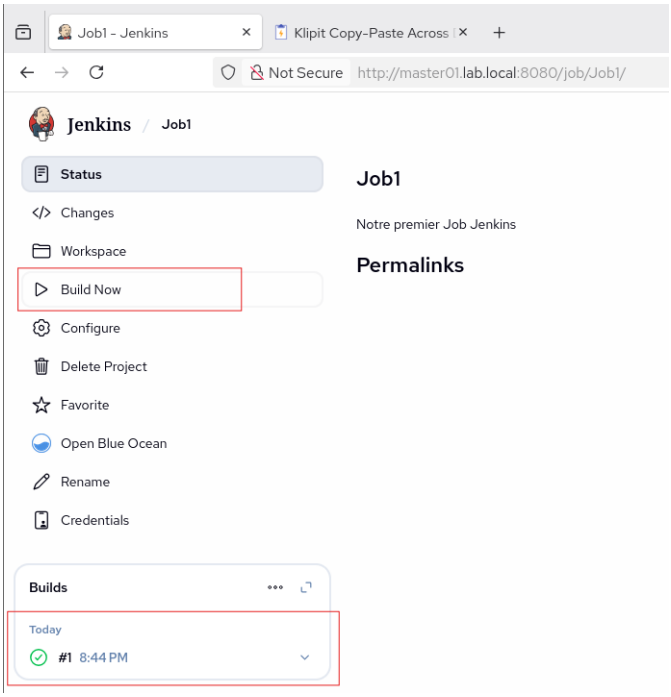
```
#!/bin/bash
cat > java_test.java << 'EOF'
class java_test {
    public static void main(String[] args) {
        System.out.println("Hello Java World!");
    }
}
EOF

javac java_test.java
java java_test
```

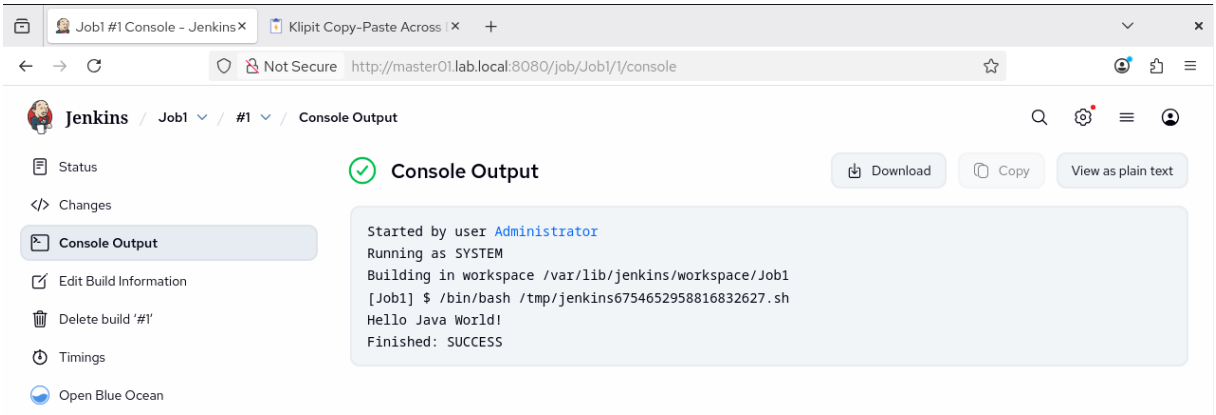
Apply & Save pour accéder à la page du projet.

Construire un Job pour la première fois

Cliquez sur **Build Now** pour lancer un build. Une fois la construction démarrée, vous verrez le statut dans la section **Historique de build**.

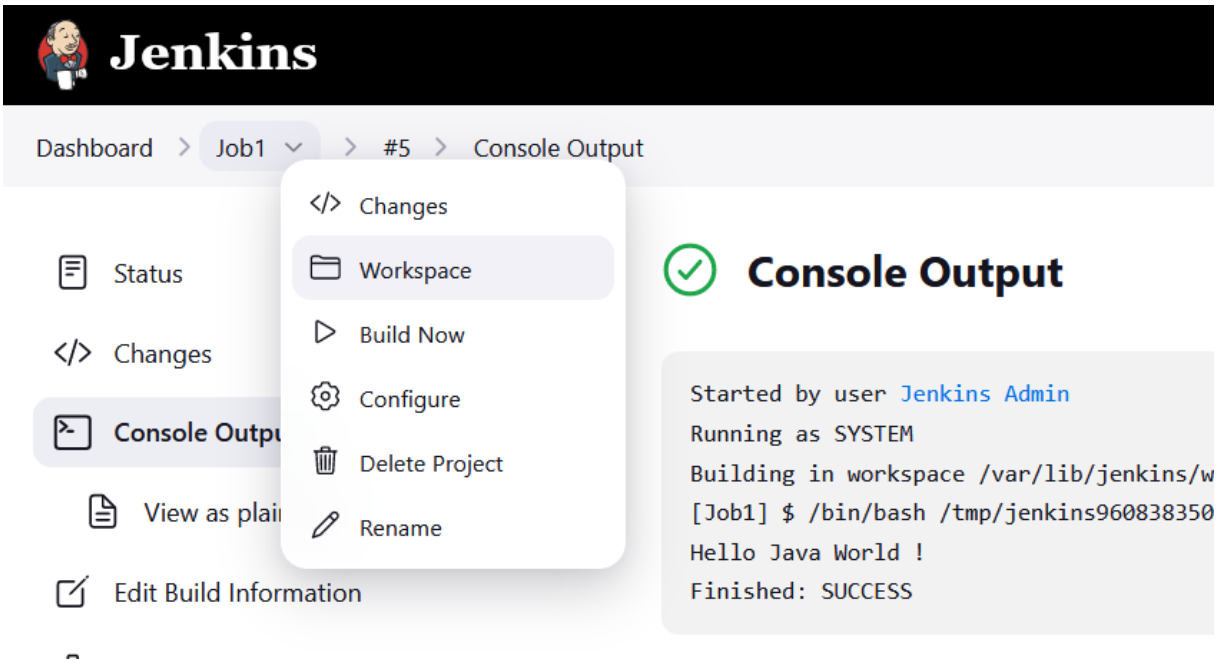


Une fois la construction terminée, cliquez sur le numéro de version qui commence par #. En cliquant sur le numéro de build, par exemple **#1** , vous accédez à la page qui affiche les statistiques de build. L'option intéressante sur cette page peut être l'**état de la console,** qui affiche la sortie au moment de l'exécution.

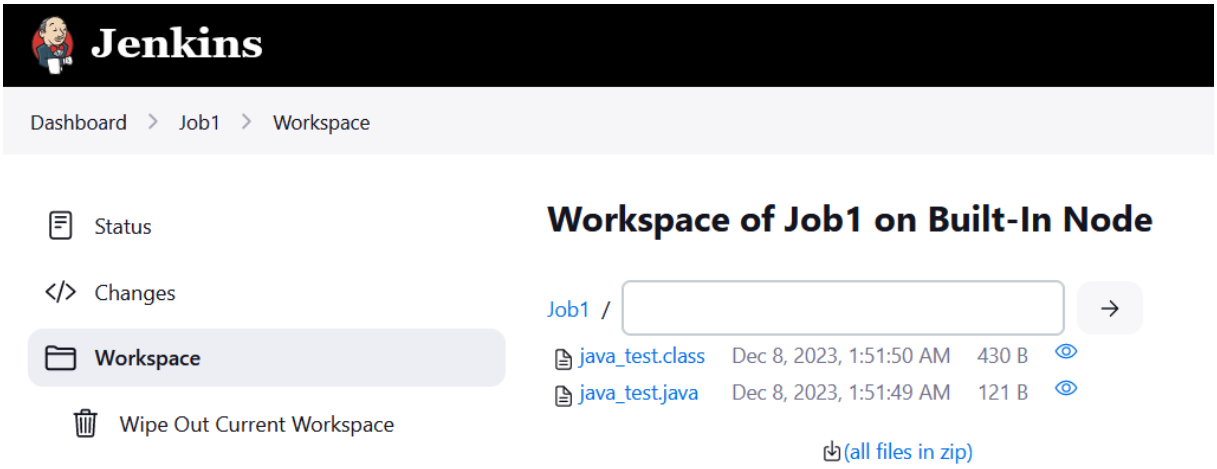


Notez le répertoire « Building in workspace »

Revenez au Workspace du Job1



Notez les fichiers présents dans le répertoire workspace



Ajout de déclencheurs aux jobs

Créer des déclencheurs

Les déclencheurs de construction décident quand une tâche jenkins est exécutée. Que cela se produise sur la base d'un événement externe, par exemple un push to git repository, une exécution planifiée, ou un travail est exécuté après qu'un autre travail est terminé, il existe de nombreuses options pour déclencher des compilations.

Depuis la page du projet, cliquez sur **Configurer**.

Types de déclencheurs

1. Déclencher les builds à distance (Par exemple, à partir de scripts)
2. Construire après le build sur d'autres projets
3. Construire périodiquement
4. GitHub hook trigger for GITScm polling
5. Scrutation de l'outil de gestion de version

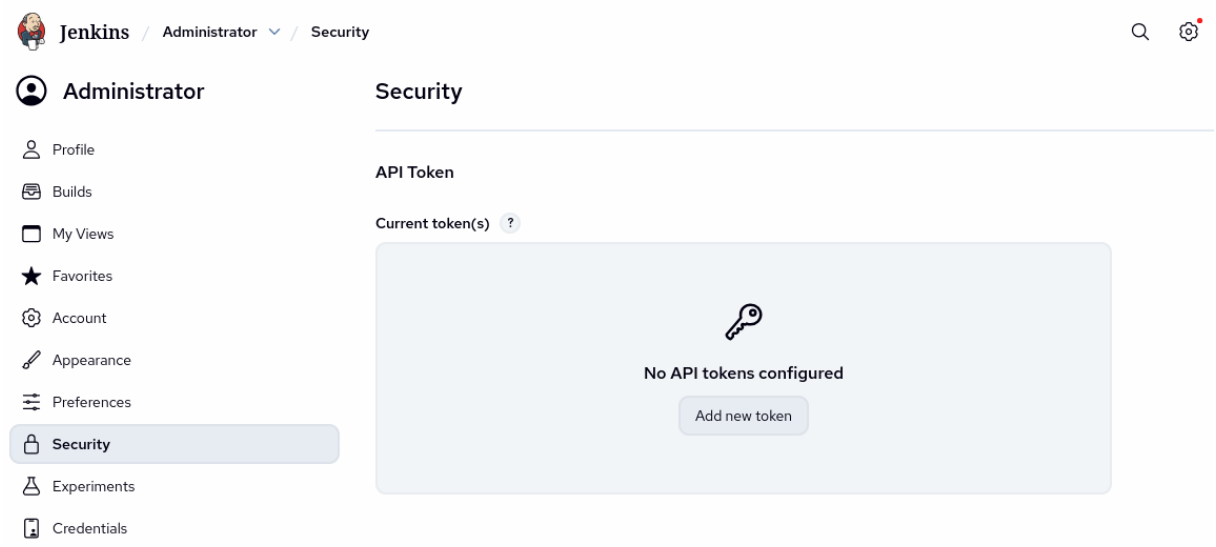
Déclencher les builds à distance

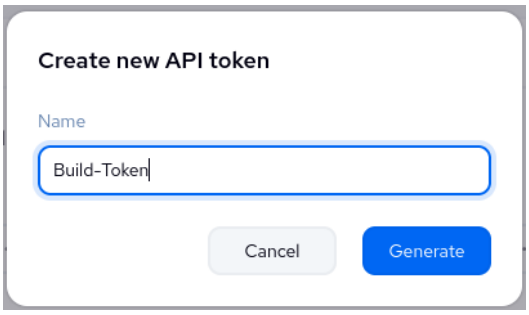
Les jobs peuvent être déclenchés à distance en dehors de jenkins. Ceci est très utile lorsque vous souhaitez que les travaux soient déclenchés en fonction d'un événement ou d'une partie de la logique que vous avez écrite dans le cadre de votre script. C'est également de cette manière que vous déclencheriez le travail en fonction des activités effectuées sur les référentiels. par exemple, ajouter un nouveau commit à git hub.

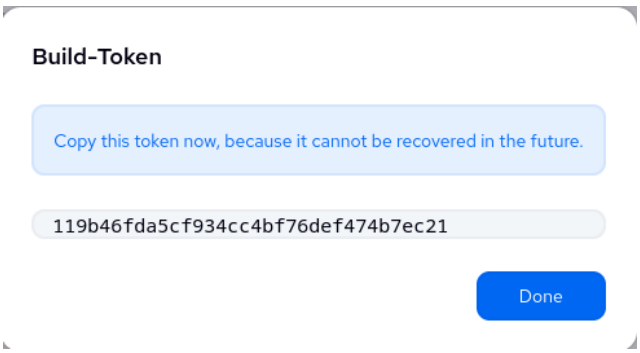
Exemple:

- Cliquez sur *job1* puis sélectionnez Confgure
- Dans **Build Trigger** , cochez **Trigger Builds remotely**.

- Définir un jeton
- Sauvegardez le travail.
- Pour déclencher le travail, vous avez besoin de deux choses
    - utilisateur
    - jeton API de l'utilisateur
- Nous utiliserons le jeton d'API de l'utilisateur admin (premier utilisateur que nous avons créé) pour déclencher cette tâche. Vous pouvez le trouver sur **Manage Jenkins -> Users -> admin -> Security**







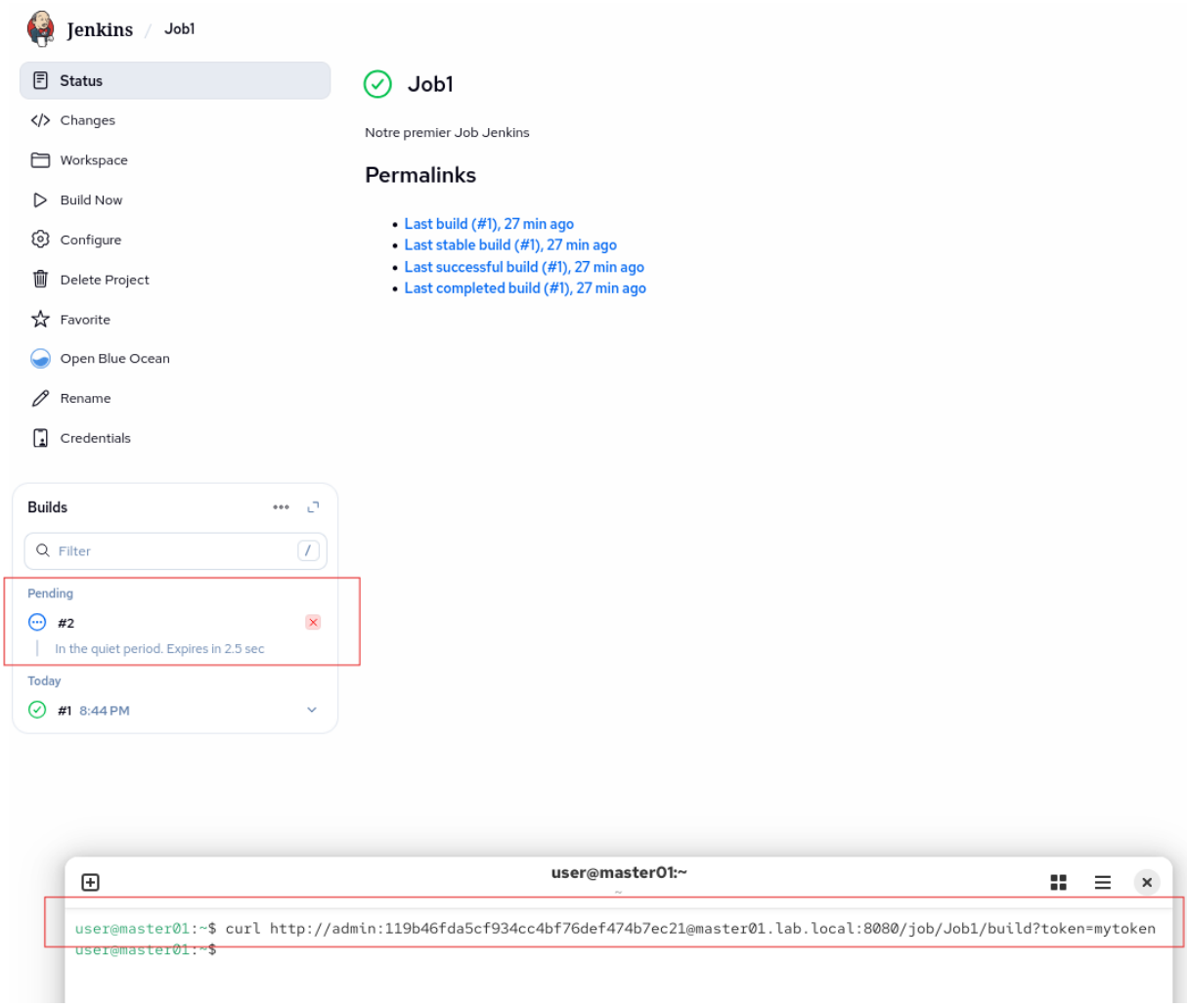
Copiez ce token dans le bloc notes

- Accédez au déclencheur à partir du navigateur ou utilisez curl

user:<API_TOKEN>@<Jenkins_URL>/job/job1/build?token=<JOB_TOKEN>

Exemple:
http://admin:552dab89b070c0fcc3fad281c51318ad@10.40.1.14:8080/job/job1/build?token=mytoken

- Cela déclenchera la construction.



Construire après le build sur d'autres projets (pipeline)

L'une des caractéristiques importantes de Jenkins est sa capacité à créer un pipeline de Jobs, alors qu'en fonction du résultat d'un travail, un autre peut être déclenché. Par exemple seulement si vous êtes capable de compiler le code, vous voudrez peut-être procéder au test, sinon il est inutile de le faire. L'utilisation de Build après que d'autres projets sont déclenchés, cela peut être facilement réalisé. Nous allons créer un pipeline de travaux en utilisant cette fonctionnalité dans le prochain Lab.

Construire périodiquement

Semblable à la création de cronjobs ou de travaux planifiés, il est possible de définir un calendrier d'exécution avec jenkins.

Scrutation de l'outil de gestion de version

Cette option permet à jenkins de vérifier régulièrement le système de gestion du code source, par exemple un référentiel git distant pour vérifier s'il y a des mises à jour, et de lancer un travail basé sur celui-ci. Idéalement, les hooks / webhooks de commit avec git devraient déclencher les builds, mais cela n'est pas toujours possible. Par exemple, si vous hébergez des jenkins dans un réseau privé, qui n'est pas accessible par le référentiel git hébergé sur le cloud, le déclenchement d'un webhook ne sera pas possible. Dans de

tels cas, la meilleure option suivante est d'interroger le référentiel git à intervalles réguliers et de déclencher les builds.

GitHub hook trigger for GITScm polling

Si Jenkins reçoit le hook PUSH du référentiel GitHub défini dans la section Git SCM, il déclenchera la logique d'interrogation Git SCM. La logique d'interrogation appartient donc en fait à Git SCM.