

Jobs et Pipelines avec JENKINS

v1.0.0 | 02/09/2025 | Auteur : Bauer Baptiste

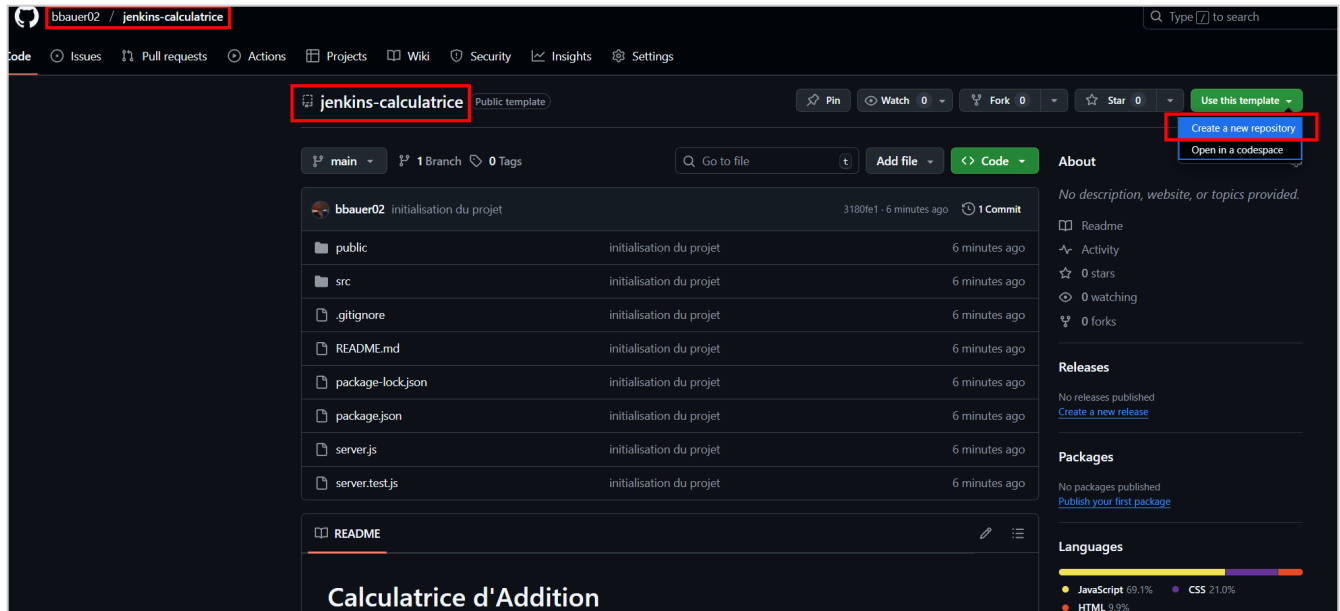
Chapitre | Durée de réalisation : 4 heures

Table des matières

1. Etape 1 : Création du dépôt d'application GIT.	1
2. Etape 2 : Création de l'image Docker de l'application calculatrice-jenkins	5
3. Etape 3 : Utilisation de Jenkins pour exécuter une version de notre application.	7
3.1. Configuration de Docker dans le conteneur Jenkins	7
3.2. Création d'un Webhook GitHub	9
3.3. Création d'un GitHub App	12
3.4. Installer l'application Github dans le dépôt.	17
3.5. Création d'un Credentials dans Jenkins	17
3.6. Création du Job (projet) Jenkins	19

1. Etape 1 : Création du dépôt d'application GIT.

Rendez-vous sur le dépôt de l'application [Calculatrice Jenkins](#) et cliquez sur "Use this template" pour créer votre propre dépôt à partir de ce template.



Il faut ensuite installer et configurer GIT localement dans la machine virtuelle.

```
sudo apt update && sudo apt install git -y
```

```
ubuntu@ip-172-31-38-30:~$ sudo apt install git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
ubuntu@ip-172-31-38-30:~$
```

Il faut ensuite configurer son identité GIT :

- Cette configuration est utilisée pour les commits :

Modifier les commandes en fonction de vos informations

```
git config --global user.name "bbauer02"
git config --global user.email "bbauer02@gmail.com"
```

- Vérification de la configuration :

```
git config --list
```

Ensuite il faut configurer une clé SSH pour pouvoir cloner le dépôt GIT sans avoir à saisir vos identifiants à chaque fois.

```
ssh-keygen -t ed25519 -C "bbauer02@gmail.com"
```

Démarrer l'agent SSH et ajouter la clé :

```
eval "$(ssh-agent -s)"  
ssh-add ~/.ssh/id_ed25519
```

Afficher la clé publique :

```
cat ~/.ssh/id_ed25519.pub
```

Copier la **clé publique** et l'ajouter dans les paramètres SSH de votre compte **GitHub**.



- <https://github.com/settings/keys>

SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

[New SSH key](#)

Authentication keys

 SSH	PC Desktop [Redacted] Added on Nov 8, 2024 Last used within the last 3 months — Read/write	Delete
 SSH	PcPortable [Redacted] Added on Nov 20, 2024 Last used within the last 5 months — Read/write	Delete

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

[New GPG key](#)

There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).

Vigilant mode

☐ **Flag unsigned commits as unverified**

This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.
[Learn about vigilant mode.](#)

Add new SSH Key

Title

Key type

Authentication Key ▾

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

[Add SSH key](#)

Dans un dossier `/home/<utilisateur>/app`, cloner le dépôt GIT de l'application :

```
mkdir -p ~/app/calculatrice-jenkins
```

```
cd ~/app/calculatrice-jenkins
```

Initialiser le dépôt GIT localement et ajouter le dépôt distant :

```
git init
git branch -m main
git config --global init.defaultBranch main
git remote add origin https://github.com/<votre_compte_github>/jenkins-
calculatrice.git
git pull origin main
```



Remarque : Votre nom d'utilisateur *GitHub* est sensible à la casse.

Parfait ! nous avons maintenant notre serveur distant qui contient Jenkins d'installé ainsi que l'application **calculatrice-jenkins** lié à un dépôt **GITHUB**.

2. Etape 2 : Création de l'image Docker de l'application calculatrice-jenkins

Nous allons maintenant créer l'image Docker de l'application `calculatrice-jenkins` et la pousser sur **Docker Hub**.

Q1) Rédiger un `Dockerfile` pour construire l'image Docker.

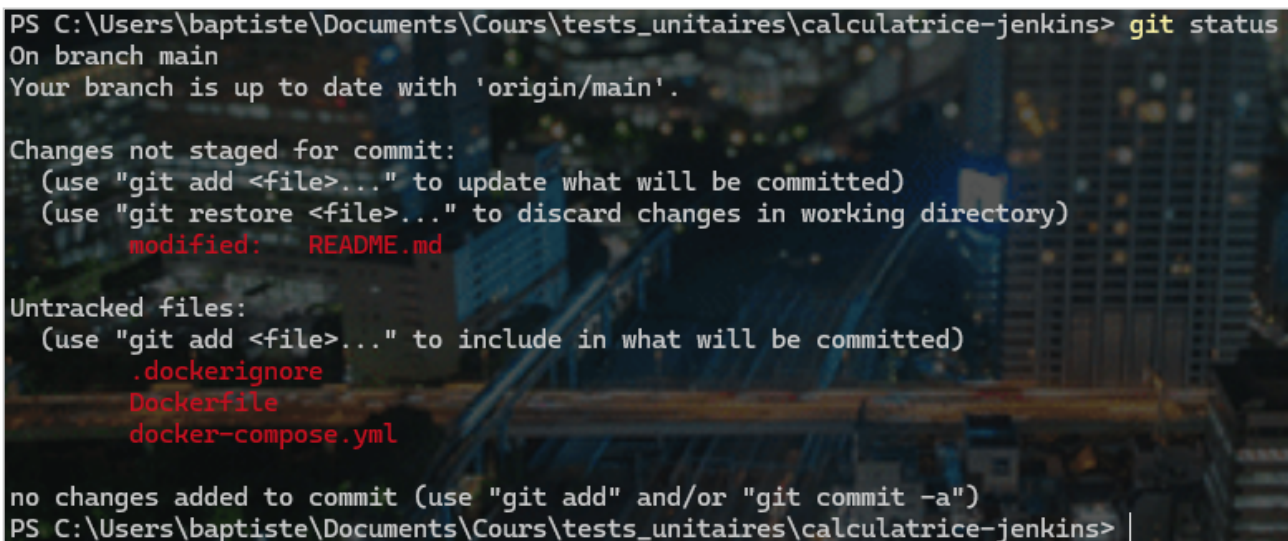
réponse 1 disponible.

Q2) Rédiger un `docker-compose.yml` : grâce à ce fichier chaque nouvelle version de l'image pourra être montée facilement : `docker-compose up -d`

réponse 2 disponible.

Il faut maintenant push les 2 fichiers sur le dépôt GIT distant :

Sur ma machine locale la commande : `git status` me permet de voir les fichiers modifiés.



```
PS C:\Users\baptiste\Documents\Cours\tests_unitaires\calculatrice-jenkins> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .dockerignore
        Dockerfile
        docker-compose.yml

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\baptiste\Documents\Cours\tests_unitaires\calculatrice-jenkins> |
```

Je peux ensuite ajouter les fichiers au commit :

```
git add Dockerfile docker-compose-calculatrice.yaml
ou
git add .
```

Puis je peux faire mon commit :

```
git commit -m "Ajout du Dockerfile et du docker-compose"
git push --set-upstream origin main
```

Maintenant que les fichiers sont dans le dépôt distant, nous allons pouvoir construire l'image Docker depuis notre serveur Jenkins.

bbauer02 Ajout du Dockerfile et du docker-compose f94bfb1 · 30 minutes ago 2 Commits		
public	initialisation du projet	6 hours ago
src	initialisation du projet	6 hours ago
.dockerignore	Ajout du Dockerfile et du docker-compose	30 minutes ago
.gitignore	initialisation du projet	6 hours ago
Dockerfile	Ajout du Dockerfile et du docker-compose	30 minutes ago
README.md	Ajout du Dockerfile et du docker-compose	30 minutes ago
docker-compose.yml	Ajout du Dockerfile et du docker-compose	30 minutes ago
package-lock.json	initialisation du projet	6 hours ago
package.json	initialisation du projet	6 hours ago
server.js	initialisation du projet	6 hours ago
server.test.js	initialisation du projet	6 hours ago



Vérifier que votre dépôt Git est mis à jour sur le site **GitHub**. Vous devriez voir votre nouveau message (Ajout du Dockerfile et du docker-compose) (l'horodatage de validation a été mis à jour).

Pour cela, il faut se connecter en **SSH** sur le serveur Jenkins et faire un **git pull** pour récupérer la dernière version du dépôt GIT.

```
cd /home/jenkins/app/calculatrice-jenkins
```

```
ubuntu@ip-172-31-38-30:~/app/calculatrice-jenkins$ |
```

ensuite :

```
git pull origin main
```


3. Etape 3 : Utilisation de Jenkins pour exécuter une version de notre application

3.1. Configuration de Docker dans le conteneur Jenkins

L'unité fondamentale de Jenkins est le Job (également connu sous le nom de projet). Vous pouvez créer des tâches qui effectuent diverses tâches, notamment les suivantes :

- Récupérer du code à partir d'un référentiel de gestion de code source tel que GitHub, GitLab
- Créer une application à l'aide d'un script ou d'un outil de construction.
- Créer un package d'une application et l'exécuter sur un serveur

Dans cette partie, nous allons créer un Job Jenkins simple qui récupère la dernière version de votre Calculatrice-jenkins à partir de GitHub et exécute un script de construction. Dans Jenkins, nous pourrions ensuite tester notre application et l'ajouter à un pipeline de développement.

Nous devons donc paramétrer le Socket de Docker pour que Jenkins puisse exécuter des commandes Docker.

- Vérifions si le socket est bien monté

```
docker inspect jenkins --format '{{json .HostConfig.Binds}}' | jq .
```

- Récupérer le GID du groupe docker (sur l'hôte)

```
getent group docker | cut -d: -f3
```

Exemple : 988 pour ma machine

- Vérification/ajustement des droits du socket sur l'hôte (important) :

```
ls -l /var/run/docker.sock
# doit idéalement être: srw-rw---- 1 root docker ... /var/run/docker.sock
sudo chgrp docker /var/run/docker.sock
sudo chmod 660 /var/run/docker.sock
```

- Créer/ajuster le groupe docker dans le conteneur, remplacez XXX par le GID du groupe docker

```
# Sur l'hôte : récupérer le GID (ex: 988)
HOST_DOCKER_GID=$(getent group docker | cut -d: -f3); echo "$HOST_DOCKER_GID"

# Dans le conteneur : aligner le GID du groupe 'docker' sur celui de l'hôte
docker exec -u root jenkins groupadd -f -g "$HOST_DOCKER_GID" docker || true
docker exec -u root jenkins groupmod -g "$HOST_DOCKER_GID" docker 2>/dev/null || _
```

true

```
# Ajouter l'utilisateur jenkins au groupe docker
docker exec -u root jenkins usermod -aG docker jenkins
```

- Installer le client Docker dans le conteneur Jenkins

Par défaut, l'image `jenkins/jenkins:lts-jdkXX` n'a pas la commande docker. Il faut l'installer.

❑ Docker socket = API REST Unix

- Le fichier `/var/run/docker.sock` est un socket Unix exposé par le daemon Docker (`dockerd`).
- Ce socket est en fait une **API REST locale** : par exemple : `curl --unix-socket /var/run/docker.sock http://localhost/containers/json` permet de lister les conteneurs sans docker d'installé.

Donc en théorie : pas besoin du client docker si nous souhaitons appeler l'API directement.



❑ Mais alors pourquoi installer docker (le client) ?

- Le binaire docker n'est qu'un client CLI qui parle à ce socket.
- Jenkins ou les jobs peuvent appeler l'API REST directement (plugins spécifiques, scripts en curl, librairies).

Mais dans la pratique :

Beaucoup de pipelines Jenkins utilisent des étapes comme `sh 'docker build ...'`.

Sans le binaire docker, ces commandes échoueront (*docker: command not found*).

```
docker exec -u root jenkins bash -lc 'apt-get update && apt-get install -y
docker.io && rm -rf /var/lib/apt/lists/*'
```

- Redémarrer le conteneur Jenkins

```
docker restart jenkins
```

- Vérifier que le client docker fonctionne

```
# Vérifier l'appartenance groupe & GID
docker exec jenkins id jenkins          # doit montrer ... 988(docker) chez toi
docker exec jenkins getent group docker
```

```
# Vérifier les droits vus DANS le conteneur
docker exec jenkins ls -l /var/run/docker.sock
# doit être root docker et rw pour le groupe
```

Nous devons voir docker dans la liste des groupes.

```
ubuntu@ip-172-31-38-30:~/app/calculatrice-jenkins$ docker exec jenkins id jenkins
uid=1000(jenkins) gid=1000(jenkins) groups=1000(jenkins),998(docker)
```

Testons maintenant la commande docker dans le conteneur Jenkins :

```
docker exec jenkins docker version
```



□ Pourquoi toutes ces manipulations ?

- Le **noyau** se base sur les **numéros (GID)**, pas les noms.
- Tant que le groupe docker dans le conteneur n'a pas **le même GID que le socket monté** (côté hôte), nous aurons l'erreur : **permission denied**.
- S'assurer aussi que le socket est bien **root:docker** et **0660** côté hôte.

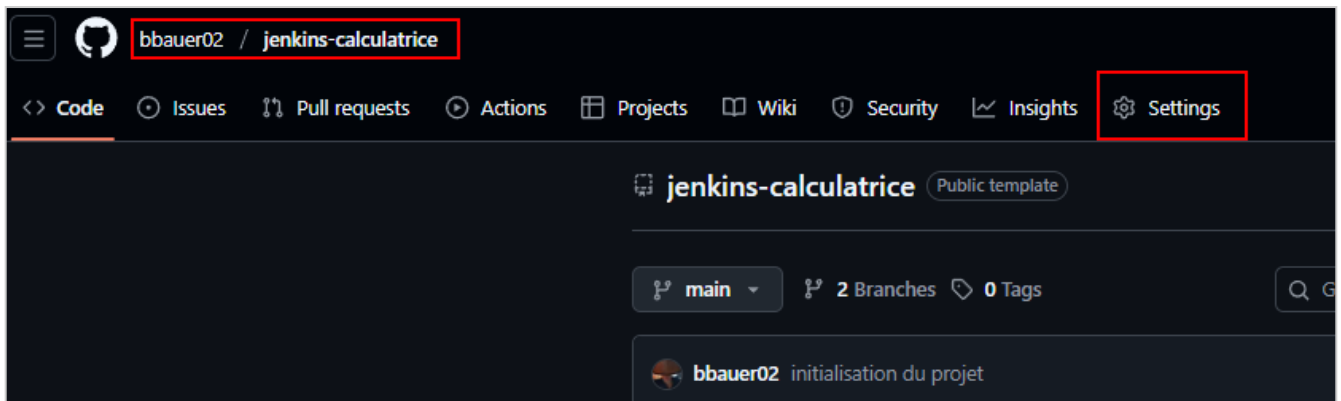
3.2. Création d'un Webhook GitHub

Un webhook est un mécanisme qui permet à GitHub de prévenir automatiquement une application externe (ici Jenkins) lorsqu'un événement survient dans un dépôt (ex. un push, une pull request, une release...).

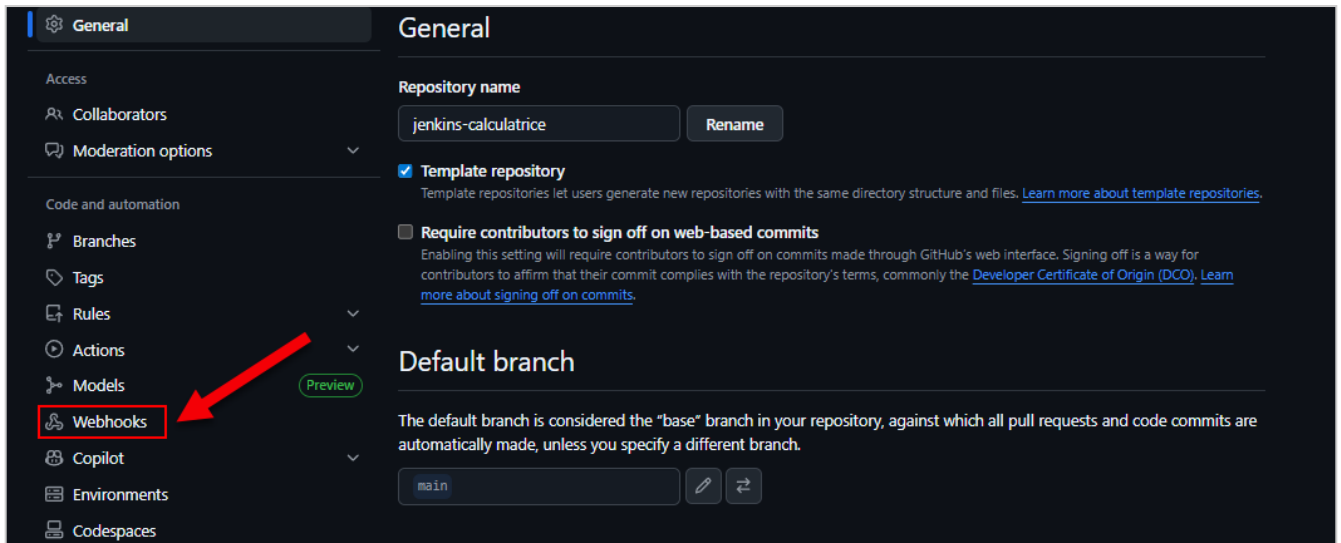
□ Plutôt que Jenkins vérifie toutes les minutes si ton code a changé (polling), GitHub envoie directement une notification HTTP (un message JSON) à Jenkins.

- **Automatisation** : déclencher un job Jenkins dès qu'il y a un commit sur GitHub.
- **Réduction de la charge** : évite à Jenkins d'interroger GitHub en boucle (poll SCM).
- **Rapidité** : la compilation/lancement de tests démarre immédiatement après un git push.
- **Cohérence DevOps** : intègre GitHub (gestion du code) et Jenkins (CI/CD) de façon fluide.

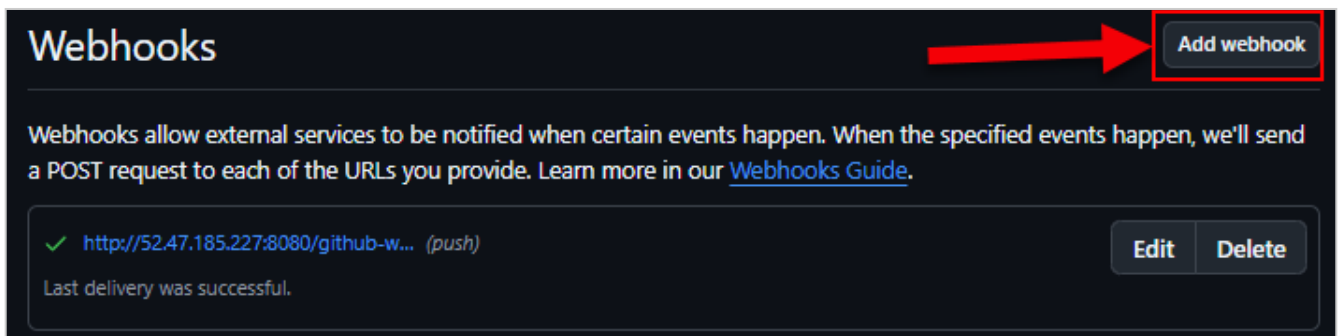
Pour créer un webhook dans github, il faut aller dans les paramètres du dépôt (Settings)



Puis cliquez sur Webhooks :



Puis cliquez sur **Add Webhook**



Remplissez le formulaire comme suit :

The screenshot shows the 'Webhooks / Manage webhook' interface in Jenkins. It has two tabs: 'Settings' and 'Recent Deliveries'. The 'Settings' tab is active. The page contains the following elements:

- Introductory text:** 'We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).'
- Payload URL *:** A text input field containing 'http://52.47.185.227:8080/github-webhook/'. A red arrow points to this field with the annotation 'Adresse IP de votre server avec Jenkins et le port utilisé par Jenkins'.
- Content type *:** A dropdown menu showing 'application/json'. It is highlighted with a red box.
- Secret section:** A blue box with an information icon and text: 'There is currently a secret configured for this webhook. If you've lost or forgotten this secret, you can change it, but be aware that any integration using this secret will need to be updated.' Below this is a 'Change secret' button. A red arrow points to this section with the annotation 'Saisissez une phrase secrète'.
- SSL verification:** A section with a lock icon and text: 'By default, we verify SSL certificates when delivering payloads.' It has two radio buttons: 'Enable SSL verification' (selected) and 'Disable (not recommended)'.
- Which events would you like to trigger this webhook?:** A section with three radio buttons: 'Just the push event.' (selected and highlighted with a red box), 'Send me everything.', and 'Let me select individual events.'
- Active:** A checked checkbox labeled 'Active' with the text 'We will deliver event details when this hook is triggered.'
- Buttons:** 'Update webhook' (green) and 'Delete webhook' (red).

Une fois le webhook créé, nous pourrions brancher ce webhook dans le job Jenkins que nous allons créer. Github enverra une notification à Jenkins à chaque fois qu'un push sera effectué sur le dépôt vers l'URL enregistrée dans le webhook.

Pour ma part, il s'agit de : <http://52.47.185.227:8080/github-webhook/>

Créons maintenant une GitHub App pour Jenkins.

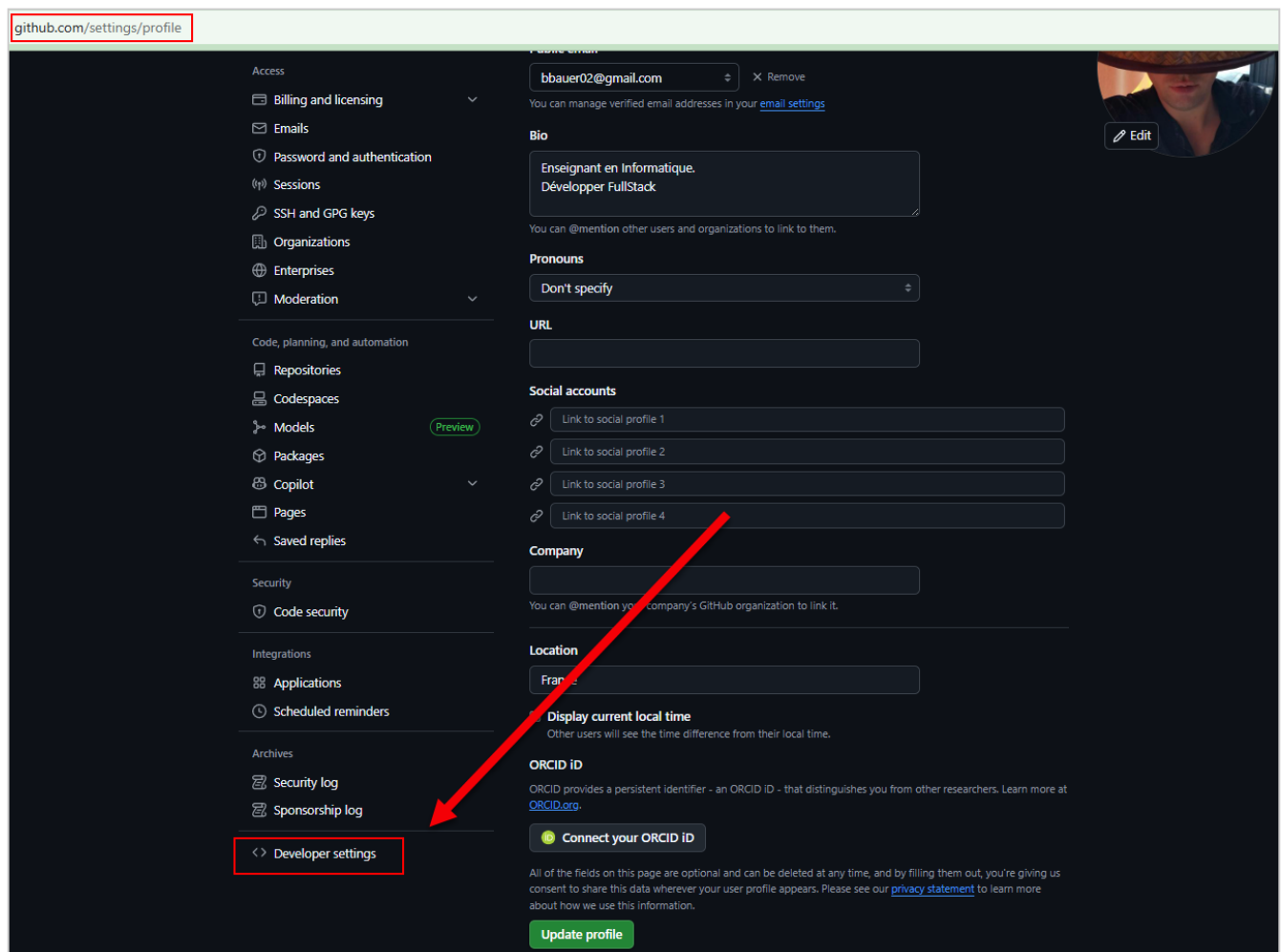
3.3. Création d'un GitHub App

Une **GitHub App** est une application déclarée dans GitHub, qui peut interagir avec les dépôts d'un utilisateur ou d'une organisation.

Elle utilise une authentification par App ID et clé privée, au lieu d'un mot de passe ou d'un token personnel.

Pourquoi créer une GitHub App pour Jenkins ?

- **Sécurité renforcée** : au lieu de donner ton compte GitHub à Jenkins, tu crées une App qui a ses propres identifiants.
- **Permissions fines** : tu choisis exactement ce que l'App peut faire (ex. lire le code, gérer les PR, etc.).
- **Scalabilité** : une App peut être installée sur plusieurs dépôts ou organisations.
- **Bonne pratique DevOps** : séparation claire entre ton compte GitHub personnel et l'accès machine de Jenkins.
- Allez dans les settings de GitHub (clique sur votre photo de profil en haut à gauche) → **Developer settings** → **GitHub Apps** → **New GitHub App**



- Cliquez sur **New GitHub App** et remplissez le formulaire comme suit :

Register new GitHub App

GitHub App name *

calculatrice-jenkins

The name of your GitHub App.

Write Preview Markdown supported

This is displayed to users of your GitHub App

Homepage URL *

http://52.47.185.227:8080/

The full URL to your GitHub App's website.

Identifying and authorizing users

The full URL to redirect to after a user authorizes an installation.

Read our [Callback URL documentation](#) for more information.

Add Callback URL

Callback URL

e.g. https://example.com/auth **Delete**

☒ **Expire user authorization tokens**

This will provide a refresh_token which can be used to request an updated access token when this access token expires.

☐ **Request user authorization (OAuth) during installation**

Requests that the installing user grants access to their identity during installation of your App

Read our [Identifying and authorizing users for GitHub Apps documentation](#) for more information.

☐ **Enable Device Flow**

Allow this GitHub App to authorize users via the Device Flow.

Read the [Device Flow documentation](#) for more information.

Post installation

Setup URL (optional)

Users will be redirected to this URL after installing your GitHub App to complete additional setup.

☐ **Redirect on update**
Redirect users to the 'Setup URL' after installations are updated (E.g. repositories added/removed).

Webhook

☒ **Active**
We will deliver event details when this hook is triggered.

Webhook URL *

`http://52.47.185.227:8080/github-webhook/`

Events will POST to this URL. Read our [webhook documentation](#) for more information.

Secret

`votre_phrase_secrete_a_memoriser`

Read our [webhook secret documentation](#) for more information.

Le webhook que nous venons de créer plus haut

Phrase secrète à saisir

Ensuite dans les permissions, il faut configurer les permissions comme suit :

Laisser les autres permissions par défaut.

Contents **Selected** **Access: Read-only**
Repository contents, commits, branches, downloads, releases, and merges. [Learn more.](#)

Metadata **Mandatory** **Access: Read-only**
Search repositories, list collaborators, and access repository metadata. [Learn more.](#)

Pull requests **Selected** **Access: Read-only**
Pull requests and related comments, assignees, labels, milestones, and merges. [Learn more.](#)

Puis, nous allons souscrire à des événements :

Subscribe to events

Based on the permissions you've selected, what events would you like to subscribe to?

- | | |
|---|--|
| <input type="checkbox"/> Installation target ⓘ
A GitHub App installation target is renamed. | <input type="checkbox"/> Meta ⓘ
When this App is deleted and the associated hook is removed. |
| <input type="checkbox"/> Security advisory ⓘ
Security advisory published, updated, or withdrawn. | <input type="checkbox"/> Commit comment ⓘ
Commit or diff commented on. |
| <input type="checkbox"/> Create ⓘ
Branch or tag created. | <input type="checkbox"/> Delete ⓘ
Branch or tag deleted. |
| <input type="checkbox"/> Fork ⓘ
Repository forked. | <input type="checkbox"/> Gollum ⓘ
Wiki page updated. |
| <input type="checkbox"/> Label ⓘ
Label created, edited or deleted. | <input type="checkbox"/> Milestone ⓘ
Milestone created, closed, opened, edited, or deleted. |
| <input type="checkbox"/> Merge queue entry ⓘ
Merge Queue entry added | <input type="checkbox"/> Public ⓘ
Repository changes from private to public. |
| <input checked="" type="checkbox"/> Pull request ⓘ
Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestoned, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked. | <input type="checkbox"/> Pull request review ⓘ
Pull request review submitted, edited, or dismissed. |
| | <input type="checkbox"/> Pull request review comment ⓘ
Pull request diff comment created, edited, or deleted. |
| | <input type="checkbox"/> Pull request review thread ⓘ
A pull request review thread was resolved or unresolved. |
| <input checked="" type="checkbox"/> Push ⓘ
Git push to a repository. | <input type="checkbox"/> Release ⓘ
Release created, edited, published, unpublished, or deleted. |
| <input type="checkbox"/> Repository ⓘ
Repository created, deleted, archived, unarchived, publicized, privatized, edited, renamed, or transferred. | <input type="checkbox"/> Repository dispatch ⓘ
When a message is dispatched from a repository. |
| <input type="checkbox"/> Watch ⓘ
User stars a repository. | <input type="checkbox"/> Star ⓘ
A star is created or deleted from a repository. |
| <input type="checkbox"/> Workflow job ⓘ
Workflow job queued, waiting, in progress, or completed on a repository. | <input type="checkbox"/> Workflow dispatch ⓘ
A manual workflow run is requested. |
| | <input type="checkbox"/> Workflow run ⓘ
Workflow run requested or completed on a repository. |

Avant de finaliser, cochez la case "Only allow this GitHub App to be installed on private repositories" si votre dépôt est privé.

Where can this GitHub App be installed?

☒ **Only on this account**
Only allow this GitHub App to be installed on the @bbauer02 account.

☐ **Any account**
Allow this GitHub App to be installed by any user or organization.

[Create GitHub App](#) [Cancel](#)

Registration successful. You must [generate a private key](#) in order to install your GitHub App.

Settings / Developer settings / GitHub Apps / calculatrice-jenkins2

About

Owned by: @bbauer02

App ID: 1888807

Using your App ID to get installation tokens? You can now [use your Client ID instead](#).

Client ID: lv23lizjD2CxyYm7TAj

[Revoke all user tokens](#)

GitHub Apps can use OAuth credentials to identify users. Learn more about identifying users by reading our [integration developer documentation](#).

Client secrets [Generate a new client secret](#)

You need a client secret to authenticate as the application to the API.

Basic information

GitHub App name *

calculatrice-jenkins2

The name of your GitHub App.

Cliquez et téléchargez la clé .PEM

Après avoir créé l'application, vous serez redirigé vers la page de l'application. Vous devez générer une clé privée pour l'application.

La clé privé sera utilisée par Jenkins pour s'authentifier auprès de GitHub. Toutefois, le format de la clé est au format : **PKCS#1** (**BEGIN RSA PRIVATE KEY**) alors que le plugin Jenkin exige le format **PKCS#8** (**BEGIN PRIVATE KEY**).

Voici une solution rapide pour convertir la clé au format **PKCS#8** :

- Copier le contenu de la clé privée générée par GitHub dans un fichier **private_key_jenkins.pem** sur votre machine Ubuntu.

Exemple de contenu de la clé .PEM à copier

```
-----BEGIN RSA PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-----END RSA PRIVATE KEY-----
```

Allez dans le dossier `.ssh` de votre machine jenkins (ici `/home/jenkins/.ssh`) et créez le fichier `private_key_jenkins.pem` avec le contenu copié précédemment.

Puis convertissez la clé au format PKCS#8 avec la commande suivante :

```
sudo openssl pkcs8 -topk8 -inform PEM -outform PEM -in
VOTRE_FICHER_FORMAT_PKCS1.pem -out FICHER_FORMAT_PKCS8.pem -nocrypt
```

Affichez le contenu du fichier `FICHER_FORMAT_PKCS8.pem` et copiez-le, nous en aurons besoin pour Jenkins.

Commande :

```
cat FICHER_FORMAT_PKCS8.pem
```

Vous pouvez supprimer les fichiers `FICHER_FORMAT_PKCS8.pem` et `VOTRE_FICHER_FORMAT_PKCS1.pem` ensuite (Si vous avez bien copié le contenu de la clé privée au format PKCS#8 sur votre machine Windows).

3.4. Installer l'application Github dans le dépôt

Il faut maintenant installer l'application GitHub que nous venons de créer dans le dépôt `calculatrice-jenkins`.

- Allez dans la page de votre GitHub App et cliquez sur **Install App** dans le menu de gauche.
- Cliquez sur **Install** à côté de votre compte utilisateur (ou organisation si vous avez créé l'application dans une organisation).
- Sélectionnez le dépôt `calculatrice-jenkins` et cliquez sur **Install**.

3.5. Création d'un Credentials dans Jenkins

Il faut maintenant configurer Jenkins pour qu'il puisse utiliser cette GitHub App afin d'accéder au dépôt et capter les événements du webhook.

- Cliquez sur l'**engrenage** en haut à gauche pour accéder à la configuration de **Jenkins** puis sur **Credentials**

Jenkins / Administrer Jenkins

Administrer Jenkins

Une construction sur le nœud principal peut engendrer des problèmes de sécurité. Vous devriez configurer des builds distribués. Consulter [la documentation](#).

[Set up agent](#) [Set up cloud](#) [Dismiss](#)

Java 17 end of life in Jenkins

You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details.

[Plus d'informations](#) [Ignore](#)

Configuration du système

- System**
Configurer les paramètres généraux et les chemins de fichiers.
- Tools**
Configurer les outils, leur localisation et les installeurs automatiques.
- Plugins**
Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins.
- Nodes**
Ajouter, supprimer, contrôler et monitorer les divers nœuds que Jenkins utilise pour exécuter les jobs.
- Clouds**
Ajouter, supprimer et configurer les instances de cloud afin de provisionner les agents à la demande.
- Apparence générale**
Configurer l'apparence générale de Jenkins.

Sécurité

- Security**
Sécuriser Jenkins; définir qui est autorisé à accéder au système.
- Credentials**
Configurer credentials
- Credential Providers**
Configurer the credential providers and types
- Users**
Créer/supprimer/modifier les utilisateurs qui peuvent se logger sur ce serveur Jenkins

Cliquez sur **System** :

System

Store **System** Domains (global)

Puis sur **Identifiants Globaux (illimité)**

Identifiants globaux (illimité)

Domaine **Identifiants globaux (illimité)** Description
Credentials that should be available irrespective of domain specification to requirements matching.

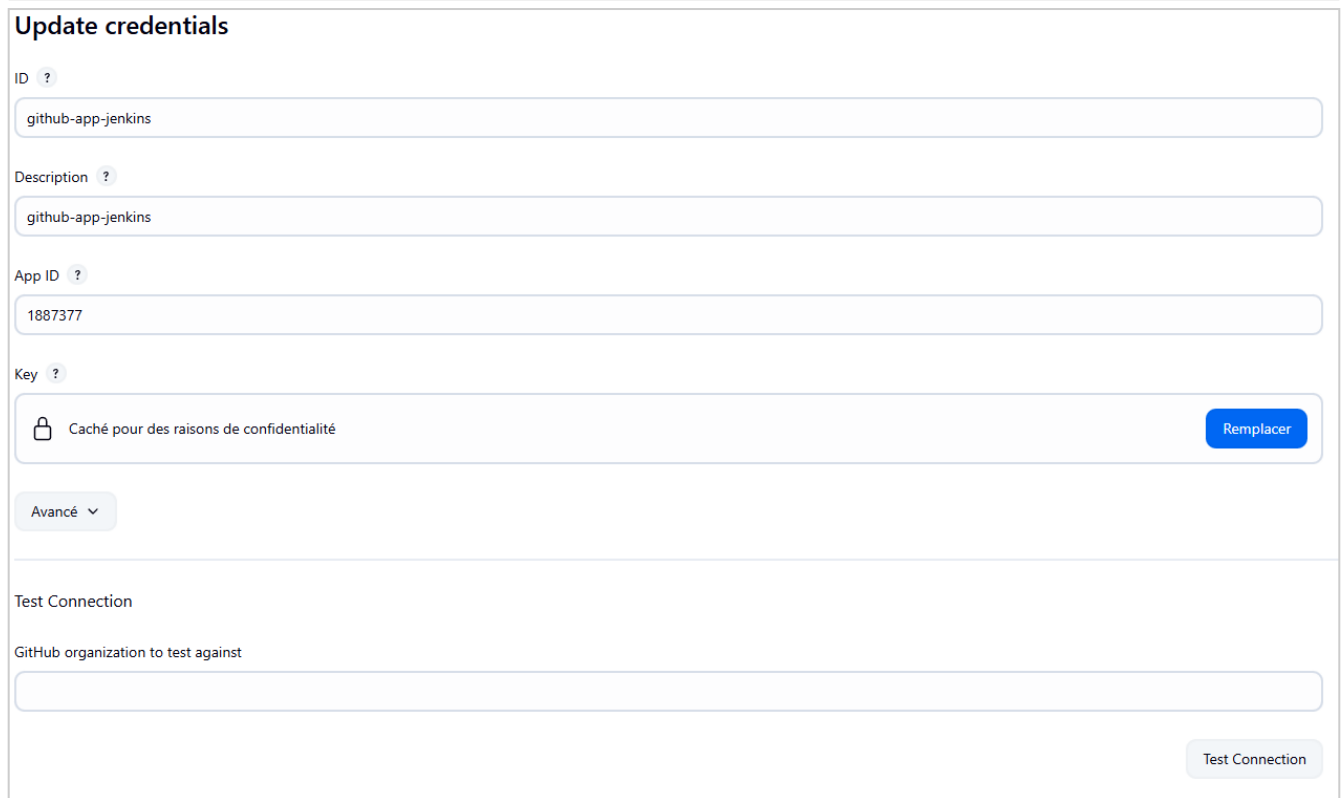
Add Credentials :

Identifiants globaux (illimité) [+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Nom	Type	Description
github-app-jenkins	1887377/***** (github-app-jenkins)	GitHub App	github-app-jenkins

Sélectionnez **GitHub App** dans le menu déroulant **Kind** ou **Type** en français.



Update credentials

ID ?
github-app-jenkins

Description ?
github-app-jenkins

App ID ?
1887377

Key ?
Caché pour des raisons de confidentialité Remplacer

Avancé ▾

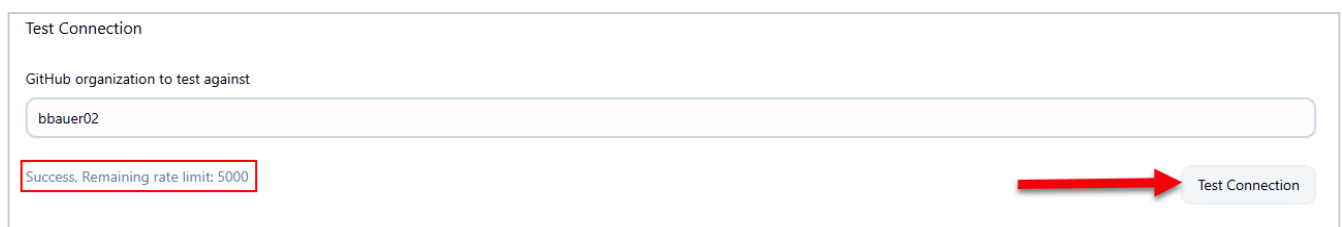
Test Connection

GitHub organization to test against

Test Connection

- **ID** : Jenkins-GitHub-App (ou un autre nom de votre choix)
- **Description** : GitHub App pour Jenkins
- **App ID** : Renseignez l'App ID de votre GitHub App (disponible dans la page de votre GitHub App)
- **Private Key** : Collez le contenu de votre clé privée au format PKCS#8 (générée précédemment)

Vous pouvez tester la connexion dans la partie "Test Connection". Saisissez le nom de votre compte Github (bbauer02 dans mon cas) et cliquez sur **Test Connection**.



Test Connection

GitHub organization to test against
bbauer02

Success, Remaining rate limit: 5000

Test Connection

3.6. Création du Job (projet) Jenkins

Un **Job Jenkins** (aussi appelé Projet) est l'unité de travail fondamentale dans Jenkins. C'est une tâche automatisée que Jenkins va exécuter : compilation, tests, déploiement, analyse de code, etc.

□ On peut voir un Job comme une recette que Jenkins suit chaque fois qu'il doit lancer une action.

Cliquer sur le lien **Créer un job** directement en dessous de la page **Bienvenue sur Jenkins!**. Vous pouvez également cliquer sur **Nouveau Item** dans le menu de gauche.

□ Pourquoi utiliser des Jobs ?

- **Automatiser** les tâches répétitives du cycle de vie logiciel.

- **Standardiser** les étapes (même script pour toute l'équipe).
- **Gagner du temps** (plus besoin d'exécuter manuellement).
- **Réduire les erreurs humaines**.
- **Intégrer le DevOps** : Jenkins sert de lien entre le code source, les tests, le déploiement.

Il existe plusieurs types de Jobs Jenkins :

- **Freestyle Project** : le type de job le plus simple et flexible. Permet d'exécuter des scripts shell, des commandes Windows, etc. Pour le configurer, une interface graphique est disponible.
- **Pipeline** : permet de définir des workflows complexes en utilisant un langage de script (Groovy). Idéal pour les processus CI/CD avancés. La configuration se fait via un fichier **Jenkinsfile** versionné avec le code.
- **Multibranch Pipeline** : similaire au Pipeline, mais permet de gérer automatiquement plusieurs branches d'un dépôt Git. Chaque branche peut avoir son propre **Jenkinsfile**. Jenkins détecte automatiquement les branches d'un repo Git contenant un **Jenkinsfile**. Utile pour tester chaque branche.
- **Folder** : permet de regrouper plusieurs jobs dans un dossier pour une meilleure organisation.

Concrètement, dans le cadre de notre application **calculatrice-jenkins**, nous allons créer un **Freestyle Project** pour récupérer le code source depuis GitHub et exécuter des commandes Docker pour construire et lancer l'application.



Dans le champ **Saisissez un nom** renseigner le nom **BuildCalculatriceJob**.

Nouveau Item

Saisissez un nom

BuildCalculatriceJob

Select an item type



Construire un projet free-style

Job legacy polyvalent qui récupère l'état depuis un outil de gestion de version au plus, exécute les étapes de build en série, suivi d'étapes post-construction telles que l'archivage d'artefacts et l'envoi de notifications par e-mail.



Pipeline

Organise des activités de longue durée qui peuvent s'étendre sur plusieurs agents de construction. Adapté pour la création des pipelines (anciennement connues comme workflows) et/ou pour organiser des activités complexes qui ne s'adaptent pas facilement à des tâches de type libre.



Construire un projet multi-configuration

Adapté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, etc.



Dossier

Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.



Organization Folder

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



Pipeline Multibranches

Crée un ensemble de projets Pipeline en se basant sur les branches détectées dans le dépôt d'un gestionnaire de code source.

OK

Cliquer sur l'onglet **General**, ajouter une description pour votre **Job**. Par exemple, "Mon premier Job Jenkins."

Général

Enabled



Description

Mon premier Job Jenkins

Texte brut [Prévisualisation](#)

☐ Ce build a des paramètres ?

☐ GitHub project

☐ Supprimer les anciens builds ?

☐ Throttle builds ?

☐ Exécuter des builds simultanément si nécessaire ?

Avancé ▾

Cliquer sur l'onglet **Gestion de code Source** et cliquer sur le bouton radio **Git**. Dans le champ URL du référentiel, ajouter votre lien de référentiel GitHub pour l'exemple d'application en prenant soin de saisir votre nom d'utilisateur correctement, car sensible à la casse.

