□□ Fonctionnement avec SSH

Maintenant si on reprends notre projet précédent, et que l'on veut non pas le déployer avec GitLab Pages, mais sur un serveur personnel.

Configurer SSH

Pour cela on peut accéder à ce serveur via SSH. Donc à la place de donner un mot de passe à GitLab, on va générer une clé SSH (comme vu dans le chapitre sur Linux)

```
ssh-keygen -f ~/.ssh/id_gitlabci
```

Ensuite on peut envoyer cette nouvelle clé (publique) sur le serveur :

```
ssh-copy-id -i ~/.ssh/id_gitlabci
```

Enfin on peut se connecter pour voir si tout fonctionne

```
ssh -i ~/.ssh/id_gitlabci <utilisateur>@<hôte>
```

En se connectant on a aussi créé le fichier `known_hosts` que l'on va utiliser plus tard.

Configurer les variables de GitLab-CI

Tout d'abord il faut que l'on récupères quelques informations de notre configuration SSH (la clé privée, et le fichier known_hosts)

```
cat ~/.ssh/id_gitlabci
cat ~/.ssh/known_hosts
```

Maintenant on peut aller sur notre projet GitLab, dans les paramètres CI/CD, puis dans la section "Variables".

Ici on va ajouter 2 variables :

Nom de la variable	Contenu	Protégée ?
`SSH_PRIVATE_KEY`	La clé privée (`cat ~/.ssh/id_gitlabci`)	Oui
`SSH_KNOWN_HOSTS`	Le fichier known_hosts (`cat ~/.ssh/known_hosts`)	Oui

Le fichier .gitlab-ci.yml

Maintenant on peut se repencher sur le fichier que l'on a créé un peu plus tot. Dans celui ci on va ajouter ceci dans la section `before_script` :

```
# "pages" est le nom du "job"
prod:
    # Stage indique quel type d'action qui est effectuée (par exemple: test, build, deploy)
    stage: deploy

    # L'image est la base du système dans lequel les commandes d'installation du projet vont être lançées. Dans ce cas ci, debian
    # Nous verrons plus en détail cela dans le chapitre sur Docker
    image: debian

    # Plus besoin d'artefacts car on en a pas besoin et que l'on utilise pas GitLab Pages

    # Le "before_script" spécifie les commandes d'installation de l'environement, par exemple ici nous allons utiliser rsync pour déployer notre projet. Donc j'installe rsync
    # Il faut toujours faire en sorte que les commandes ne nécessite pas d'intéraction (exemple, en ajoutant -y à la commande APT)
    before_script:
        - apt-get update
        - apt-get install -y rsync

        # On va installer ssh-eval et le lancer dans l'environement de build
        - apt-get install openssh-client -y
        eval $(ssh-agent -s)
        # On va ensuite ajouter notre clé privée
        - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
        # Et on va mettre les bonnes permissions pour le dossier .ssh
        - mkdir -p ~/.ssh
        - chmod 700 ~/.ssh
```

```
    # Enfin on va ajouter le fichier known_hosts pour ne pas créer de problème lors de la vérification automatique des clés
    - echo "$SSH_KNOWN_HOSTS" >> ~/.ssh/known_hosts
    - chmod 644 ~/.ssh/known_hosts


  # Le "script" est la partie principale. Elle indique les commandes à lancer pour déployer notre projet.
  # Ici on va envoyer tous les fichiers vers un répertoire d'un serveur distant grâce à rsync et SSH
  script:
    - rsync -rv * <utilisateur>@<hôte>:/<chemin vers le dossier cible>

  # On précise que seul les pushs vers la branche main peuvent appeller le job "prod"
  only:
    - main
```

Maintenant on peut envoyer le tout au serveur et déclencher la pipeline que l'on vient de modifier.

```
git add .gitlab-ci.yml
git commit -m "Changement de Pages à un serveur distant"
git push origin main
```

Et normalement, tout devrait fonctionner !

Revision #1
Created 26 April 2023 19:09:28 by SnowCode
Updated 26 April 2023 19:09:28 by SnowCode