Déploiement d’une solution *CI/CD*

Maxime Lambert

Février 2025

Quatrième TP de virtualisation pour le module R4.08, BUT Informatique deuxième année.

Travail en binôme en salle de machines.

Savoirs de référence étudiés :

- Virtualisation d’applications
- Conteneurs

Pré-requis :

- Aisance sous un système Unix en ligne de commande
- CM du module R4.08 sur les conteneurs Linux
- Notions sur les réseaux TCP/IP
- Architecture client-serveur

Matériel ou logiciels utilisés :

- Accès au serveur Proxmox de l’Université depuis un navigateur web

1 Présentation du TP

L'objectif de cette séance est de vous faire découvrir la démarche devOps[1] en déployant une solution complète d'intégration et de déploiement continu[2].

1.1 Evaluation

Cette séance de TP comporte un QCM de 15 min disponible sur e-campus. Le QCM est à réaliser de manière individuelle. Tous les supports et ressources sont autorisées pendant le QCM sauf les IA génératives.

1.2 Prérequis

→ Si cela n'a pas déjà été fait lors du TD3 et que le fichier existe, exécutez sur votre machine hôte la commande

```
$ mv ~/.docker/config.json ~/.docker/config.json_ignore
```

De cette manière aucun proxy ne sera utilisé par vos conteneurs, mais vous pourrez facilement rétablir la configuration au besoin.

→ Faites le ménage sur votre machine

```
$ docker rm -f $(docker ps -aq)
$ docker system prune -f --all
```

→ Dans le fichier `/etc/hosts`, ajoutez la ligne :

```
$ 127.0.0.1        gitea drone-serveur drone-demon registre sonar sonarqube
```

→ Ouvrez le fichier `/home/user/.profile` et remplacez la dernière ligne `export no_proxy="…"` par la ligne :

```
export no_proxy="localhost,127.0.0.1/8,172.17.0.0/16,prof,gitea,drone-
serveur,drone-demon,sonar,sonarqube,registre"
```

N.B : ne pas faire de retour à ligne, c'est une contrainte de formatage sur ce document. Ce n'est pas une erreur d'avoir dans le fichier `/etc/hosts` *uniquement* `sonarqube` *et dans le fichier* `/home/user/.profile` `sonarqube` *et* `sonar`*.*

→ Enfin, redémarrez votre machine avec la commande `sudo reboot`.

[1]DevOps : Le terme devOps n'est pas une méthode a proprement parlé. Il s'agit plus d'une démarche visant à concilier et rapprocher la partie « développement » de la partie « exploitation » qui historiquement étaient vues comme deux mondes distincts.

[2]On utilise généralement le terme anglais *CI/CD : Continuous Integration / Continuous Delivery*

1.3 Présentation des annexes

Vous trouverez en annexes différents schéma permettant de visualiser l'infrastructure à mettre en place. Je vous recommande vivement d'en prendre connaissance et de vous y référer fréquemment si vous êtes un peu perdu entre les différents conteneurs et la configuration à mettre en place.

Annexe 1 : Schéma de l'architecture

Il s'agit d'un schéma simpliste visant à présenter quels sont les composants à déployer lors de ce TP, où se trouvent-ils et quels liens les unissent.

Annexe 2 : Schéma du réseau

Cette annexe est un diagramme logique de réseau, il est similaire à ce que nous avons fait lors du TD3. Appuyez-vous dessus afin de mettre en place le réseau reliant les conteneurs.

Annexe 3 : Schéma du stockage

À la suppression d'un conteneur, nous ne souhaitons pas perdre les données hébergées sur nos serveurs (code, configuration…). Ce schéma représente toutes les **destinations** (*target*) à utiliser avec l'option --mount.

1.4 Et si quelque chose ne fonctionne pas… ?

Vous pouvez investiguer les erreurs de configurations avec les commandes `docker logs NOM_DE_MON_CONTENEUR` ou même `docker inspect XXX`

2 Déploiement du serveur gestionnaire de versions

2.1 Présentation de Gitea

La part de marché la plus importante, et de loin, pour les logiciels gestionnaires de versions est attribuée à Git qui fonctionne selon un système distribué.

Chaque dépôt local Git peut-être lié à un ou des dépôts distants. La grande majorité des opérations est réalisée en local. Lorsque l'on souhaite partager ses modifications, on effectue une mise à jour des références distantes[3].

Il existe une interface web assez rudimentaire, gitweb[4], tandis que d'autres projets tel que Gitlab ou Github ne se limitent pas au stockage de dépôts. Gitlab par exemple, est une plateforme permettant la gestion du code source, l'analyse de vulnérabilité, l'intégration et la distribution continue ou encore la planification de projet.

Gitea, une alternative open-source simple, légère mais complète a été retenu pour ce TP. D'après leur documentation, le but de ce projet est de fournir de la manière la plus simple, la plus rapide et sans complication un service Git auto-hebergé.

2.2 Création du réseau docker

→ Merci de vous référer à l'annexe 2 « Schéma du réseau » afin de créer le réseau docker de type bridge qui va relier l'ensemble de nos conteneurs.

2.3 Déploiement de Gitea

2.3.1 Répertoire /data

→ Veuillez créer un répertoire à cet emplacement : **`/home/user/gitea/data`**

Ce répertoire sera utilisé comme la source de notre montage lié entre le système de fichiers de la machine hôte et le conteneur.

[3]Grâce à la fameuse commande `git push`
[4]https://git-scm.com/book/en/v2/Git-on-the-Server-GitWeb

2.3.2 *Instanciation du conteneur gitea*

→ Veuillez démarrer le conteneur qui hébergera notre serveur gestionnaire de versions avec la commande Docker `run` en renseignant les informations suivantes :

- **image** : `prof:5000/gitea:latest`
- **nom du conteneur** : `gitea`
- **points de montage** :
  - **type** : `bind`
  - **source**: `/home/user/gitea/data`
  - **destination** : c. f. Annexe 3 « Schéma du stockage »
- **réseau** : utiliser le nom du réseau créé dans la section 2.2
- **activer le redémarrage systématique** avec l'option `--restart=always`
- **pour ne pas bloquer votre terminal**, mettre l'option `-d`
- **réaliser un mappage** du port TCP 3000 du conteneur sur le port TCP 3000 de l'hôte

Si tout se passe bien, votre conteneur démarre. Attention à bien utiliser le nom « `gitea` ». Ce sera également le nom d'hôte de votre conteneur. En cas d'erreur cela aura une incidence sur la suite du TP.

→ Vérifiez son statut en listant les conteneurs actifs avec `docker container ls`

2.3.3 *Configuration initiale de Gitea*

→ Sur votre machine hôte, ouvrez votre navigateur à l'adresse `http://gitea:3000`.

Vous devriez obtenir un résultat identique à la figure 1.

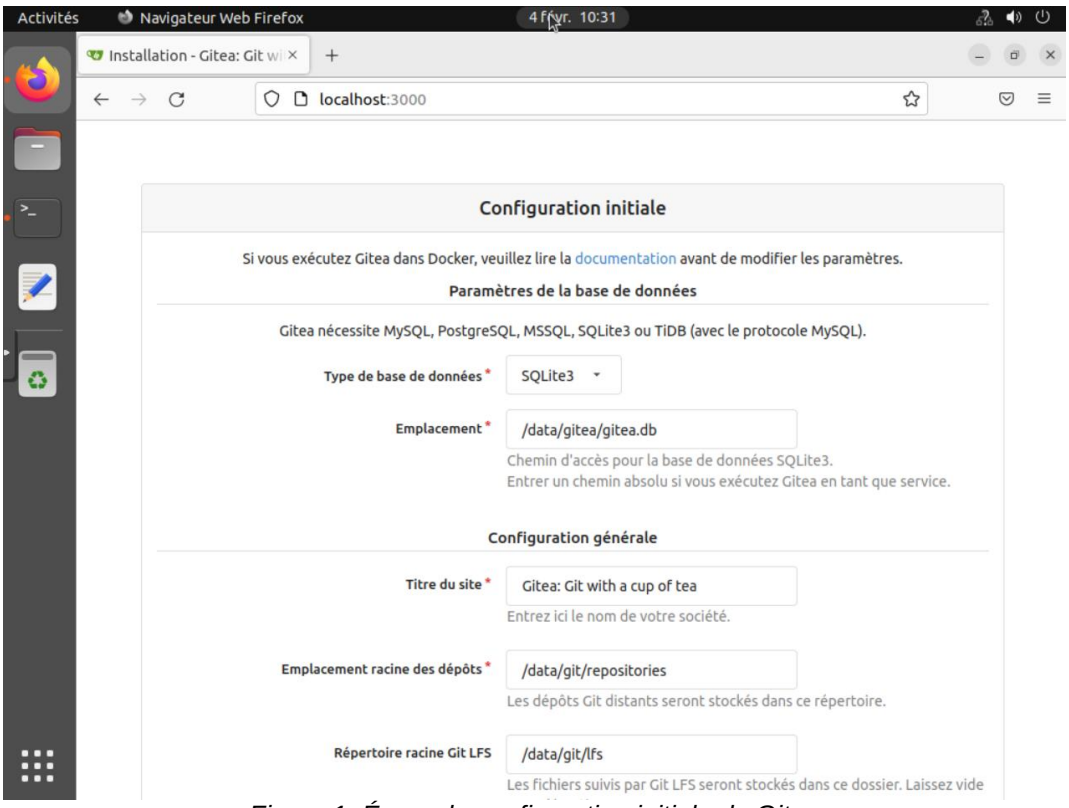


Figure 1: Écran de configuration initiale de Gitea

→ Modifiez les paramètres suivants pour la configuration initiale :

- **Titre du site** : `Gitea-groupe-XX` où `XX` correspond à votre numéro de groupe
- **Domaine du serveur** : `gitea`
- **Url de base de Gitea** : `http://gitea:3000/`

→ Vérifiez que tous les chemins sur la page commencent par `/data`. Veuillez noter que c'est à cet emplacement dans le conteneur que seront sauvegardés la base de données, les journaux, dépôts Git et autres configurations. Vérifiez votre commande Docker ayant servi à créer votre conteneur, en particulier la clé « `target` » ou « `destination` » de l'option `mount`. On ne souhaite pas perdre nos projets en cas de suppression du conteneur !

→ Une fois les vérifications et modifications effectuées, cliquez sur « Installer Gitea »

→ Le service Gitea redémarre, attendez quelques secondes avant de rafraîchir la page de votre navigateur. Un formulaire de Connexion vous sera proprosé.

Vous n'êtes pas certain de la configuration de votre point de montage ? Voici un moyen simple de le vérifier. Supprimez votre conteneur gitea (`docker container rm -f gitea`) et ré-instanciez le avec la commande exécutée dans la section 2.3.2. Lorsque vous allez rafraîchir votre navigateur, la page de configuration initiale ne doit pas vous être proposée.

2.3.4 *Création d'un utilisateur administrateur*

Le premier utilisateur créé sera administrateur sur Gitea. Veuillez noter qu'il aurait aussi été possible de créer cet utilisateur en exécutant une commande à l'intérieur du conteneur grâce à la commande `docker exec`.

→ Cliquez sur « Pas de Compte ? Inscrivez-vous maintenant ».

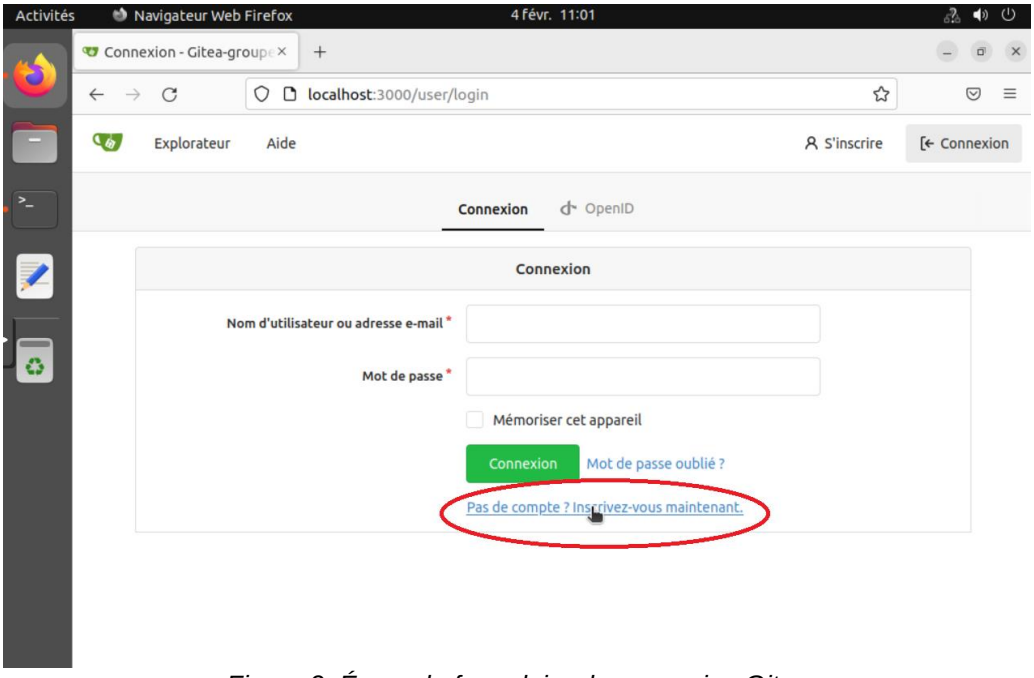


Figure 2: Écran du formulaire de connexion Gitea

→ Créer un compte avec les valeurs suivantes :

- **Nom d'utilisateur** : `etudiant`
- **Adresse e-mail** : votre adresse mail `@etu-unicaen.fr`
- **Mot de passe** : `123456`

2.3.5 Création d'un dépôt local Git pour notre application

Nous allons créer un dépôt qui va héberger le code source d'une application fournie par vos enseignants.

→ A l'aide de la figure 3, créez un nouveau dépôt que vous nommerez : `tp4`

Attention, merci de respecter scrupuleusement le nom du dépôt.

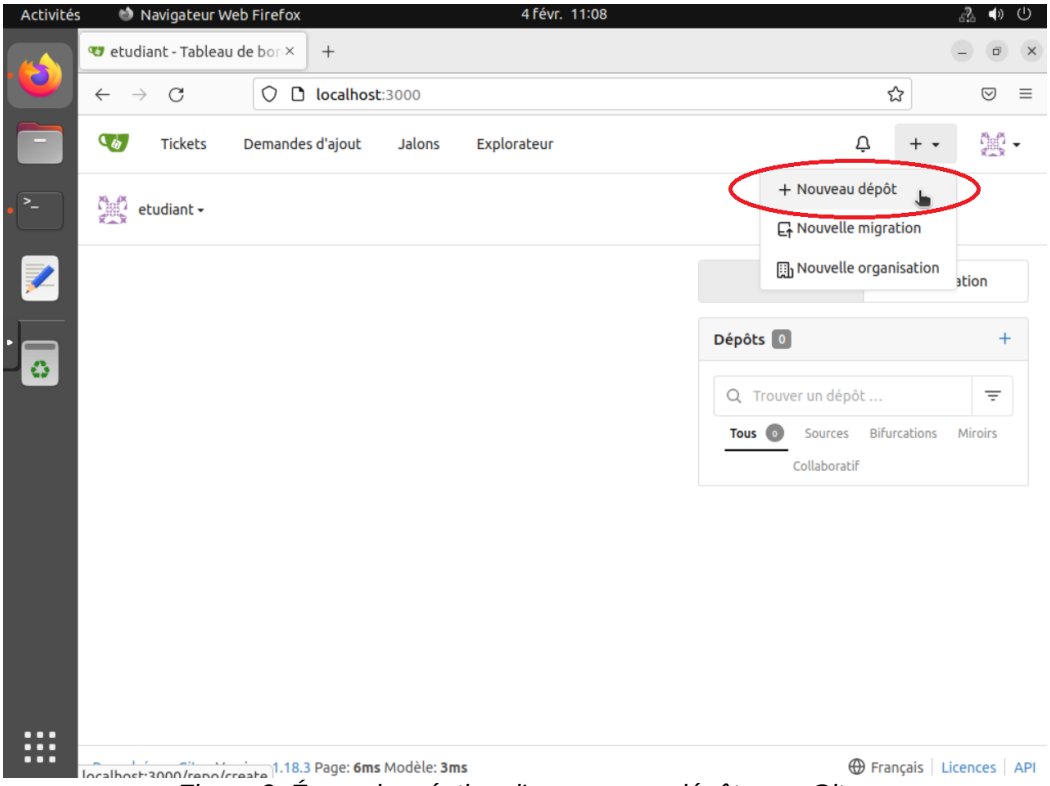


Figure 3: Écran de création d'un nouveau dépôt sous Gitea

→ Sur votre machine hôte, ouvrez un terminal et placez votre répertoire courant dans `/home/user`.

→ Téléchargez une copie du dépôt distant (dans le conteneur) sur votre machine hôte. Votre VM peut être vue comme une machine de développement. Idéalement, il faudrait séparer notre dépôt Git de la machine de développement.

```
$ git clone http://gitea:3000/etudiant/tp4.git
```

Vous venez de cloner votre dépôt distant sur votre machine hôte. À ce stage du TP, grâce à la redirection de port, vous êtes aussi en mesure de cloner les dépôts des autres étudiants hébergés sur leur VM.

2.3.6 *Initialisation de notre application*

Vos enseignants mettent à votre disposition les sources d'une application web angular. Vous allez la télécharger depuis le serveur web « prof » et la placer dans votre dépôt local Git.

→ Dans un terminal, placez-vous dans le répertoire `/home/user`.

→ Téléchargez l'archive du projet à l'aide de la commande :

```
$ curl -LJO prof/tp4.zip
```

→ Décompressez l'archive à l'aide de la commande :

```
$ unzip tp4.zip
```

→ Déplacez votre répertoire courant dans le répertoire `/home/user/tp4`

→ Vous allez maintenant enregistrer les modifications de votre dépôt local et les partager sur le dépôt distant Gitea :

```
$ git add *
$ git commit -m "init"
$ git push origin main
```

→ Si vous recevez un message d'erreur « Identité d'auteur inconnue », merci de suivre les instructions dans votre terminal.

3 Intégration continue

Nous avons partagé le fait que gitlab pouvait être utilisé comme solution d'intégration continue. Il existe de multiples logiciels répondant à ce besoin : Jenkins, CircleCI, TravisCI, TeamCity…

Pour ce TP, nous utiliserons drone. Cette plateforme d'intégration continue est déployable en s'appuyant sur des conteneurs. De plus, c'est une solution open source et la configuration des pipelines[5] via une syntaxe au format yaml est relativement accessible.

Nous allons installer drone avec deux composants :

- un serveur qui expose une interface web d'administration, et gère les différents pipelines

- un démon qui interroge le serveur pour obtenir des informations sur les pipelines à exécuter. Chaque étape d'un pipeline sera exécutée dans un conteneur éphémère

3.1 Autorisation d'accès à Gitea

Notre serveur drone va avoir besoin d'une autorisation afin d'accéder au serveur Gitea hébergeant nos registres Git.

Nous nous appuierons sur le protocole d'autorisation OAuth 2.0[6]. Ce protocole libre permet à un consommateur (dans notre cas le serveur drone) d'être autorisé à utiliser l'API sécurisée d'un autre service web (ici Gitea).

Depuis l'interface de Gitea, nous allons enregistrer une nouvelle application OAuth2.

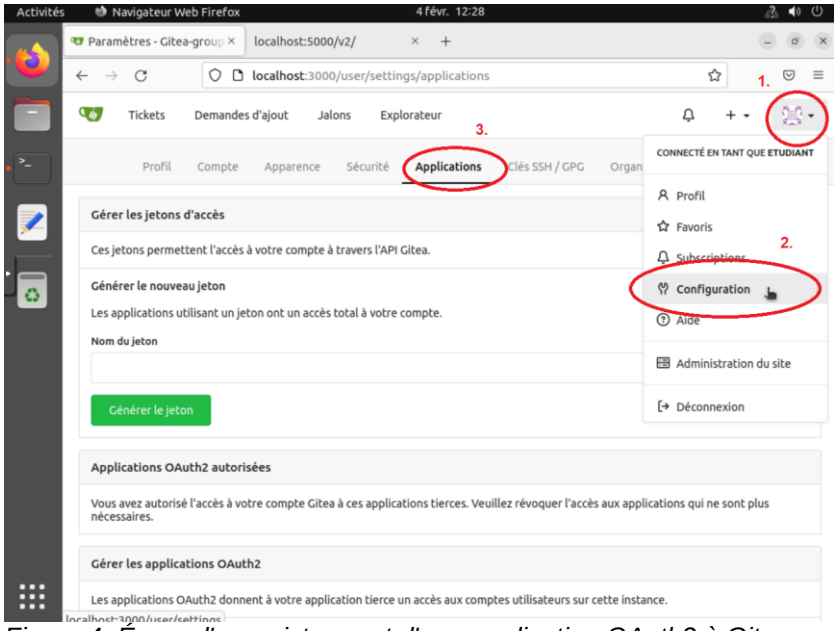


Figure 4: Écran d'enregistrement d'une application OAuth2 à Gitea

[5]Ensemble d'étapes automatisées visant à construire, analyser, tester, déployer un logiciel. Il est possible de se limiter qu'à quelques étapes.

[6]Lien vers les spécifications du protocole OAuth2 : https://oauth.net/2/

→ Sur votre machine hôte, consultez l'interface web d'administration de Gitea. Une fois authentifié, cliquez sur l'icône de votre utilisateur et accédez à la page : « Configuration / Applications ».

→ Ajoutez une nouvelle application OAuth2 avec les paramètres suivants :

- **Nom de l'application** : `drone`
- **URL de redirection** : `http://drone-serveur/login`
- **Cochez** « Confidential Client »

→ Cliquez sur « Créer une application »

→ Vous obtenez un résultat similaire à la figure 5. **Notez l'identifiant et le secret du client dès maintenant** (un simple copier-coller dans l'éditeur de texte de la VM suffira), vous en aurez besoin dans la section 3.3.1.

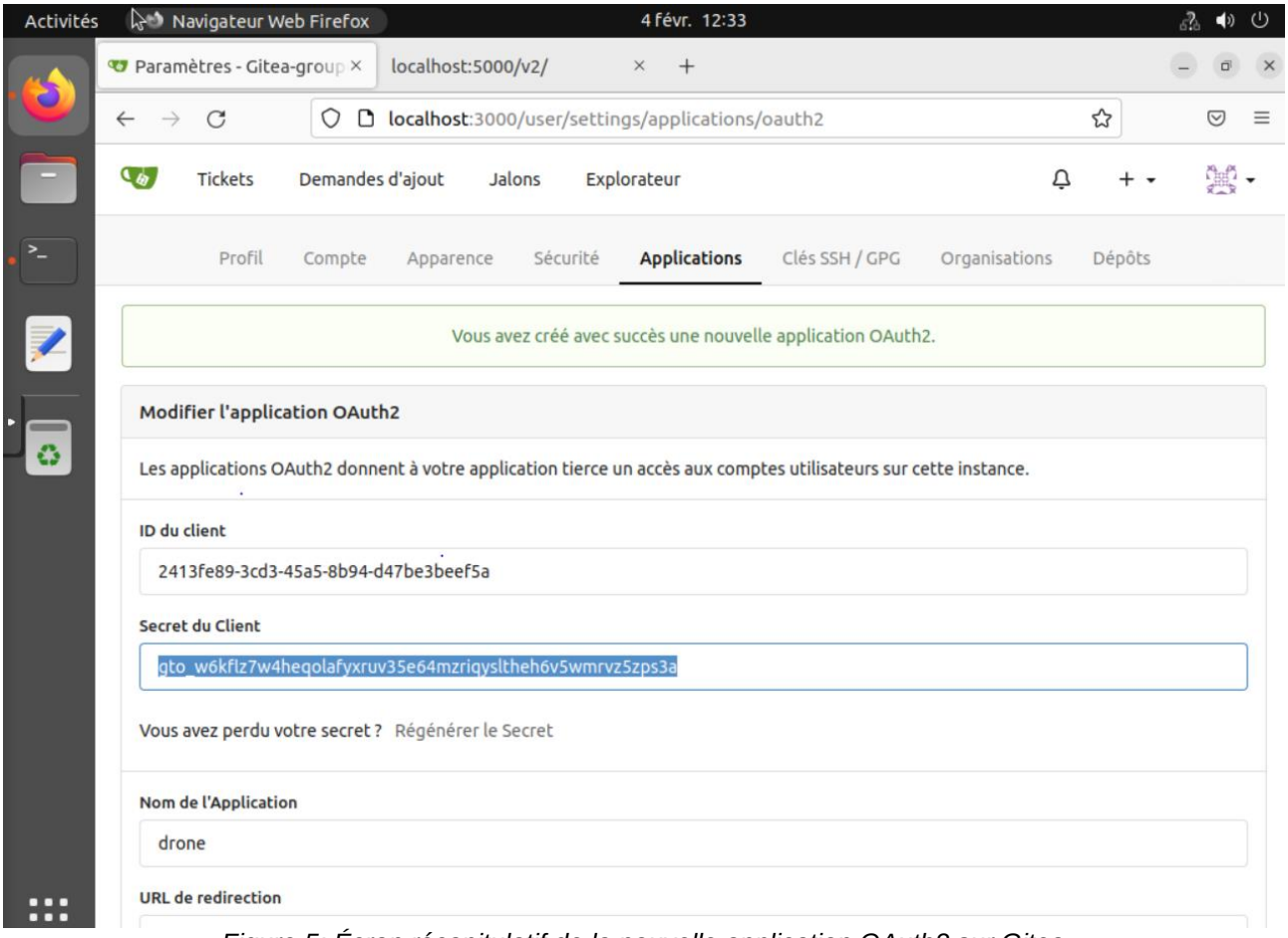


Figure 5: Écran récapitulatif de la nouvelle application OAuth2 sur Gitea

3.2 Authentification des communications entre le serveur et le démon drone

Pour authentifier les communications entre les *workers*[7] et le serveur drone, nous allons générer un secret partagé.

→ Exécutez la commande :

```
$ openssl rand -hex 16
```

→ Notez la valeur obtenue, nous en aurons besoin dans la section 3.3.1.

[7]Un *worker* est un conteneur éphémère qui exécute une étape (*step*) d'un pipeline. Il est instancié par le démon drone (*runner*).

3.3 Déploiement du serveur Drone

3.3.1 Préparation des variables d'environnement

Le serveur d'intégration continue est distribué sous la forme d'une image docker. Sa configuration est effectuée au moyen de variable d'environnements.

Pour éviter d'avoir une commande trop longue, et pour faciliter la configuration, nous allons définir un fichier qui contiendra toutes les variables d'environnement à utiliser pour notre image.

→ Créez le répertoire `/home/user/drone` et le définir en tant que répertoire courant

→ À l'intérieur de ce répertoire, créez un fichier `env-serveur` et y placez le contenu suivant :

```
DRONE_GITEA_CLIENT_ID=A_REMPLACER_PAR_ID_CLIENT_OAUTH_2_GITEA
DRONE_GITEA_CLIENT_SECRET=A_REMPLACER_PAR_SECRET_CLIENT_OAUTH_2_GITEA
DRONE_GITEA_SERVER=http://gitea:3000
DRONE_SERVER_PROTO=http
DRONE_RPC_SECRET=A_REMPLACER_PAR_LE_SECRET_DE_L_ETAPE_3.2
DRONE_SERVER_HOST=drone-serveur
```

3.3.2 Répertoire /data

→ Veuillez créer un répertoire à cet emplacement : `/home/user/drone/data`

Ce répertoire sera utilisé comme la source de notre montage lié entre le système de fichiers de la machine hôte et le conteneur.

3.3.3 Instanciation du conteneur drone-serveur

→ Veuillez démarrer le conteneur qui hébergera notre serveur d'intégration continue avec la commande Docker `run` en renseignant les informations suivantes :

- **image** : `prof:5000/drone:latest`
- **nom du conteneur** : `drone-serveur`
- **points de montage** :
  - **type** : `bind`
  - **source**: `/home/user/drone/data`
  - **destination** : c. f. Annexe 3 « Schéma du stockage »
- **réseau** : utiliser le nom du réseau créé dans la section 2.2
- **activer le redémarrage systématique** avec l'option `--restart=always`
- **pour ne pas bloquer votre terminal**, mettre l'option `-d`
- **utiliser le fichier de variables créé en 3.3.1**, avec l'option `--env-file`
- **réaliser un mappage** des ports TCP 80 et 443 du conteneur respectivement sur les mêmes ports de l'hôte

→ Ouvrez votre navigateur à l'adresse `http://drone-serveur`

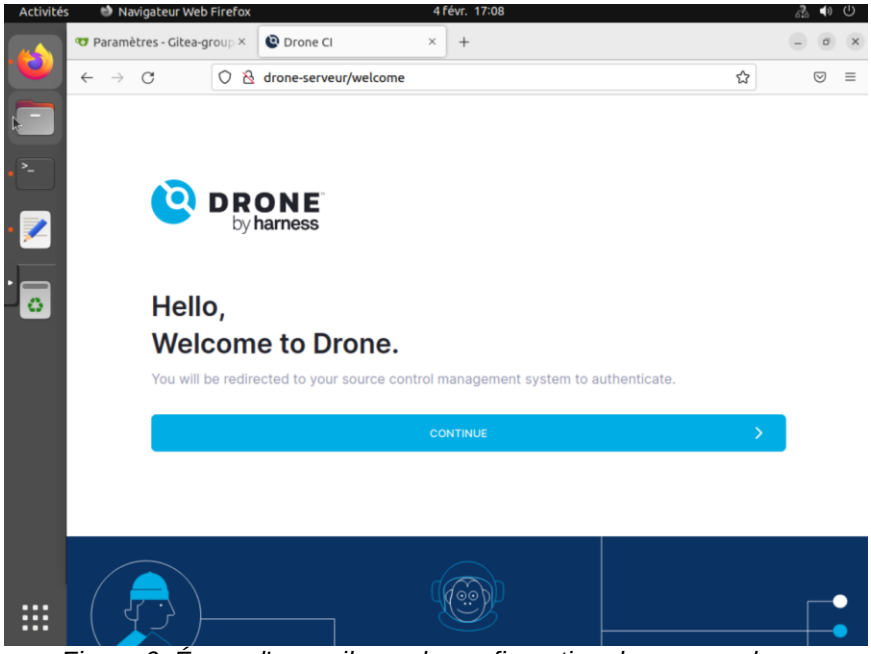


Figure 6: Écran d'accueil pour la configuration du serveur drone

→ Cliquez sur « continue » et autoriser Drone à accéder à Gitea.

Vous serez ensuite redirigé sur drone et invitez à créer un compte administrateur.

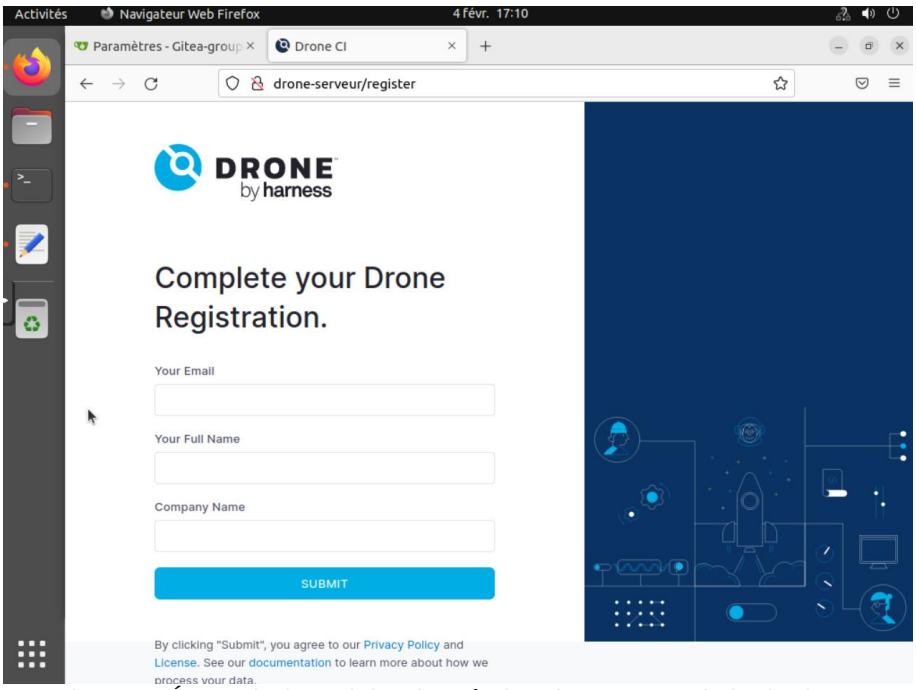


Figure 7: Écran du formulaire de création de compte admin de drone

→ Renseignez les champs de cette manière avant de cliquer sur « Submit » :

- **Your Email** : votre mail `@etu-unicaen.fr`
- **Your Full Name** : `etudiant`
- **Compagny Name** : `IUT`

Si tout se passe bien, vous devriez avoir votre dépôt `tp4` qui s'affiche.

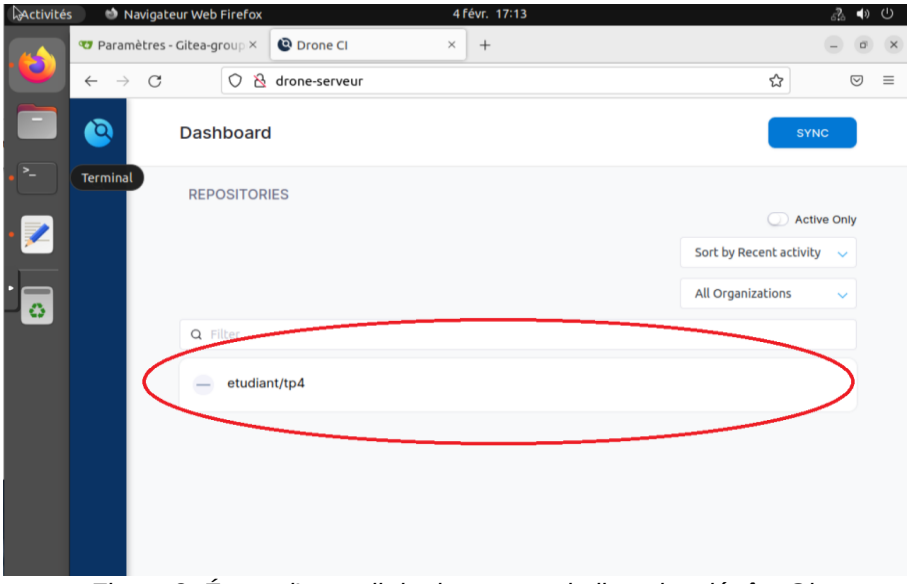


Figure 8: Écran d'accueil de drone avec la liste des dépôts Git

→ Sur l'interface web de votre serveur drone, cliquez sur votre dépôt `tp4`.

→ Dans l'onglet « Settings », cliquez sur « ACTIVATE REPOSITORY ».

Vous pouvez maintenant déclencher **manuellement** des pipelines drone sur votre dépôt tp4. Toutefois, il manque un composant à déployer pour drone et nous n'avons pas encore créé le fichier `drone.yml` permettant de décrire les étapes de notre pipeline, vous n'aurez donc aucun retour si vous appuyez sur ce bouton pour l'instant.

3.4 Déploiement du démon Drone

3.4.1 Préparation des variables d'environnement

→ Dans un terminal sur la machine hôte, créez un fichier : `/home/user/drone/env-demon` et y placer le contenu suivant :

```
DRONE_RUNNER_CAPACITY= 2
DRONE_RUNNER_NAME=my-runner
DRONE_RUNNER_NETWORKS=A_REMPLACER_PAR_NOM_RESEAU_SECTION_2.2
DRONE_RPC_HOST=drone-serveur
DRONE_RPC_PROTO=http
DRONE_RPC_SECRET= A_REMPLACER_PAR_LE_SECRET_DE_L_ETAPE_3.2
```

3.4.2 Instanciation du conteneur drone-demon

→ Veuillez démarrer le conteneur qui orchestrera l'exécution des étapes de notre pipeline d'intégration continue avec la commande Docker `run` en renseignant les informations suivantes :

- **image** : `prof:5000/drone-runner:latest`
- **nom du conteneur** : `drone-demon`
- **réseau** : utiliser le nom du réseau créé dans la section 2.2
- **activer le redémarrage systématique** avec l'option `--restart=always`
- **pour ne pas bloquer votre terminal**, mettre l'option `-d`
- **utiliser l'option** `-v /var/run/docker.sock:/var/run/docker.sock`
- **utiliser le fichier de variables créé en 3.4.1**, avec l'option `--env-file`

L'option `-v /var/run/docker.sock:/var/run/docker.sock` permet de partager le socket Docker de l'hôte avec notre conteneur démon Drone. En effet, ce conteneur est un orchestrateur qui lancera d'autres conteneurs sur la machine hôte afin de traiter les différentes étapes de nos pipelines d'intégration.

4 Exploitation de notre plateforme

4.1 Exécution des tests unitaires en mode CI/CD

Vous allez maintenant écrire un fichier permettant réaliser la compilation de cette application depuis votre serveur d'intégration continue.

Vous pouvez au choix, travailler depuis le dépôt distant tp4 sur Gitea, ou en sur le dépôt local sur votre VM dans le répertoire `/home/user/tp4`. Mais afin d'éviter de gérer des conflits git, s'il vous plaît choisissez une méthode et tenez-vous-y.

Dans le second choix, n'oubliez pas de faire les commandes suivantes pour envoyer vos modifications sur le dépôt distant :

```
$ git add *
$ git commit -m "un commentaire sur les modifications"
$ git push origin main
```

→ A la racine de votre dépôt tp4, créez un nouveau fichier `.drone.yml` et placez-y le contenu suivant :

```
kind: pipeline
type: docker
name: app
clone:
  disable: true
environment:
  HTTP_PROXY: http://VOTRE_USER_C3:VOTRE_PASSWD_C3@192.168.0.2:3128
  HTTPS_PROXY: http://VOTRE_USER_C3:VOTRE_PASSWD_C3@192.168.0.2:3128
  NO_PROXY: 127.0.0.1,localhost,gitea
steps:
  - name: clone
    image: alpine/git
    commands:
      - echo "XXX.XXX.XXX.XXX gitea" >> /etc/hosts
      - git clone http://gitea:3000/etudiant/tp4.git .
  - name: tests unitaires
    image: prof:5000/angular-cli-chrome-root:latest
    commands:
      - npm install
      - npm run test
```

→ Vous devez remplacer `xxx.xxx.xxx.xxx` par l'adresse IP du serveur gitea.

Ce fichier définit un pipeline drone. Chaque pipeline contient plusieurs étapes. Chaque étape s'exécute séquentiellement et dans son propre conteneur. Par défaut, drone s'occupe de cloner automatiquement notre dépôt Git distant.

Notre commande `npm install` va télécharger les dépendances de notre application depuis internet. Vous devrez donc configurer le proxy dans le fichier `.drone.yml`.

→ Lorsque vous êtes prêt, rendez-vous sur drone et exécutez votre pipeline avec le bouton « + new build ». Pour l'instant, le déclenchement d'un *build* sur notre pipeline est manuel.

N.B : Le mécanisme de rafraîchissement automatique ne fonctionnant pas très bien, n'hésitez pas à rafraîchir votre page afin de voir l'avancement de votre pipeline via la console de logs.

4.2 Construction et publication d'une image docker de notre application

En vous aidant de la documentation de drone (https://plugins.drone.io/plugins/docker), modifiez votre fichier `.drone.yml` afin de construire l'image de votre application et de la publier sur le registre Docker Hub.

Vous utiliserez les identifiants d'un compte créé par votre professeur afin d'être habilité à faire un push sur le registre Docker Hub.

Voici les informations à renseigner :

- image : `plugins/docker`
- dépôt : `proftp4/app`
- tags : `groupe-2024-XX` (où `XX` est votre numéro de binôme)
- nom d'utilisateur : `proftp4`
- mot de passe : `IUTc314000`
- build_args : `HTTP_PROXY=http://USER_C3:PASS_C3`3@192.168.0.2`:3128`

→ Exécutez votre pipeline et s'il n'y a pas d'erreur, vous retrouverez votre image sur Docker Hub à l'adresse suivante : https://hub.docker.com/r/proftp4/app/tags

N.B : La commande npm install peut-être très longue.

5 Pour aller plus loin

Il y a une multitude de pistes d'amélioration, en voici quelques-unes :

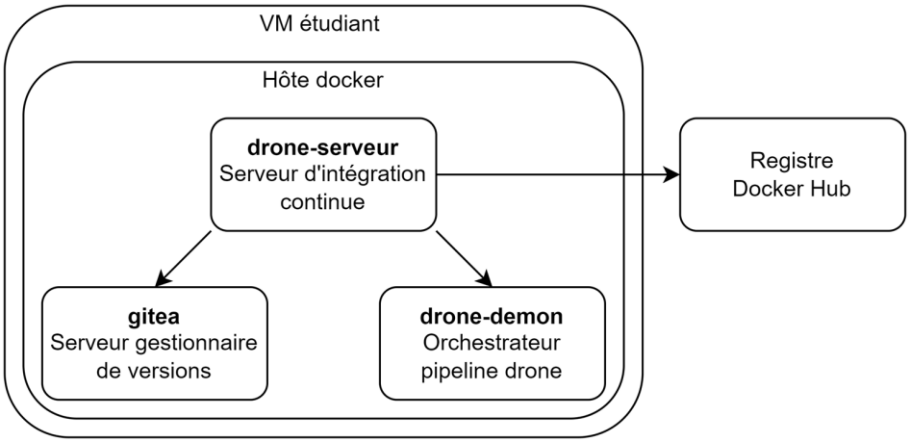
- Utilisez les secrets au lieu de mettre les mots de passe en clair
- Passez sur le protocole https
- Déployez nos conteneurs sur une VM dédiée
- Utilisez des volumes à place de montages liés
- Corrigez le décalage du fuseau horaire sur le serveur Gitea
- Déployez un serveur d'analyse de code (exemple : SonarQube)
- Déployez notre propre registre d'images Docker
- Nous avons arrêté notre déploiement à la livraison d'une image sur un registre, nous pourrions à la fin de notre pipeline lancer un conteneur avec la nouvelle version de notre image sur une machine distante.

6 Référence

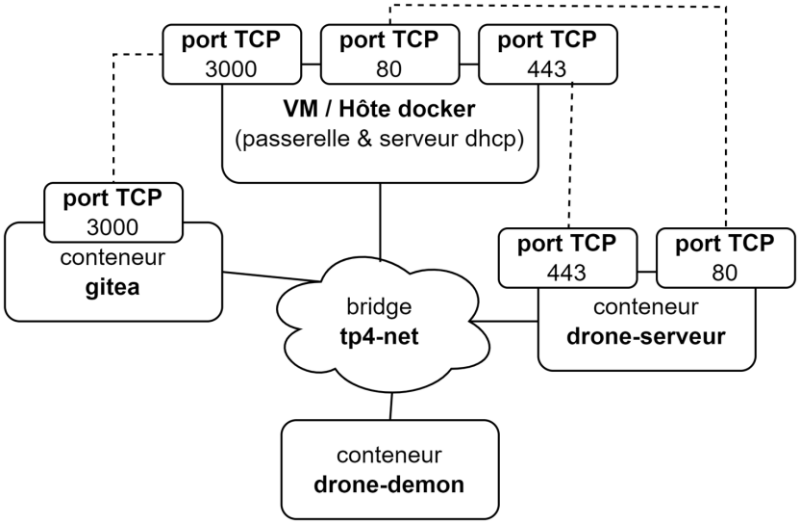
Ce TP est une adaptation du *chapitre 11 – La journée du DevOps* de cet ouvrage : MERCIER Pierre-Olivier. *Conteneurs et technologies du DevOps.* Gentilly ; Editions ALPO, 2022, 209p.

7 Annexes

Annexe 1 : Schéma de l'architecture



Annexe 2 : Schéma du réseau



Annexe 3 : Schéma du stockage