



osones

Travaux Pratiques Docker Compose

Version du 04/2017

Introduction

Dans ce TP nous allons aborder Docker Compose, l'outil intégré à Docker pour gérer des stacks applicatives.

Nous montrerons comment :

- décrire une stack
- lancer une stack
- gérer les volumes et les réseaux
- lier les applications dans une stack
- upgrader notre application

Pré-requis :

- Connaître les bases de la CLI Docker
- Posséder un compte sur DockerHub et y être connecté avec la CLI Docker
- Connaître les paramètres principaux pour lancer un conteneur

Installation

Deux méthodes d'installation sont disponibles, l'une via les packages pip, l'autre via les releases obtenues sur la page GitHub du projet docker-compose. Nous utiliserons cette dernière :

```
# curl -L
"https://github.com/docker/compose/releases/download/1.11.2/docker-compose-$(un
ame -s)-$(uname -m)" -o /usr/local/bin/docker-compose
# chmod +x /usr/local/bin/docker-compose
# docker-compose --version
docker-compose version 1.11.2, build b31ff33
```

Préparation de l'environnement

Afin de travailler dans un environnement vierge et ne impacter d'autres stacks docker-compose, créez un dossier et placez vous à l'intérieur. Toutes les commandes qui suivront seront lancées depuis ce répertoire.

```
~ $ mkdir mystack
~ $ cd mystack
```

Notre stack

Nous utiliserons l'application Gogs comme démonstrateur. Gogs est un serveur Git écrit en Go concurrent de Gitlab ou Bitbucket par exemple. L'application se compose du serveur lui même ainsi que d'une base de données. Ces deux parties seront déployées par Docker Compose.

Copier/coller cette stack dans un fichier docker-compose.yml

```
version: '2'
services:
  gogs_server:
    image: gogs/gogs:0.10.18
    ports:
      - "3000:3000"
      - "10022:22"
    volumes:
      - gogs_server_data:/data
    networks:
      - gogs_network
  gogs_db:
    image: mysql
    volumes:
      - gogs_db_data:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=password
      - MYSQL_DATABASE=gogs
    networks:
      gogs_network:
        aliases:
          - mysql

volumes:
  gogs_server_data:
    driver: local
  gogs_db_data:
    driver: local

networks:
  gogs_network:
```

Avant de lancer votre stack, essayez de retrouver :

- la version de Gogs utilisée
- La façon dont MySQL récupère son master password
- Le nom que va prendre MySQL grâce au service discovery de Docker
- Le port SSH sur lequel votre serveur Gogs écoutera

Vous pouvez maintenant lancer votre stack :

```
$ docker-compose up -d
```

Vérifiez que votre stack est correctement lancée :

```
$ docker-compose ps
```

Name	Command	State	Ports
10dockercomposegogs_gogs_db_1	docker-entrypoint.sh mysqld	Up	3306/tcp
10dockercomposegogs_gogs_server_1	/app/gogs/docker/start.sh ...	Up	0.0.0.0:10022->22/tcp, 0.0.0.0:3000->3000/tcp

Vérifiez que :

- Les volumes ont bien été créés
 - Trouvez leur emplacement sur votre système hôte
- Le réseau a bien été créé
- Les conteneurs sont bien démarrés et visibles par le démon Docker

Configuration de Gogs

Rendez vous sur votre l'adresse de votre démon Docker au port en écoute par Gogs, probablement localhost:3000 et procédez à l'installation de Gogs

Install Steps For First-time Run

If you're running Gogs inside Docker, please read [Guidelines](#) carefully before you change anything in this page!

Database Settings

Gogs requires MySQL, PostgreSQL, SQLite3, MSSQL or TiDB.

Database Type *

Host *

User *

Password *


Database Name *

Please use INNODB engine with utf8_general_ci charset for MySQL.

- Sur quelle adresse se trouve votre container MySQL ?
- Quel est le password root de la base de données ?

Il n'est pas utile de remplir toutes les informations. Seule la partie présentée dans le screenshot a besoin d'être mise à jour en fonction de votre docker-compose.yml.

Une fois l'installation terminée, vous devez vous créer un compte et ensuite vous loguer.

 [Register](#)

Créer un repository et ajoutez y un fichier quelconque. Nous souhaitons simplement tester la persistance des données.

Vérification de la persistance

Notre application fonctionne et nous afin de vérifier que nos conteneurs ne disposent pas de données locales, nous allons les détruire :

```
$ docker-compose stop
$ docker-compose rm -f
```

- Vérifiez que les volumes sont toujours présents sur le système
- Vérifiez que vous avez bien perdu l'accès à l'application

On peut maintenant relancer notre application

```
$ docker-compose up -d
```

- Vérifiez que vous avez récupéré l'accès à l'application
- Vérifiez que les données que vous y aviez mis y sont toujours

Upgrade de l'application

Notre serveur Gogs est en version 0.10.18.

© 2017 Gogs Version: 0.10.18.0313 Page: 8ms Template: 8ms

Mais une version 0.11.4 est [disponible](#).

Nous allons modifier notre fichier docker-compose.yml et changer le tag de notre image gogs/gogs pour utiliser la version 0.11.4 et relancer notre stack

```
$ sed -i 's/0.10.18/0.11.4/g' docker-compose.yml
$ docker-compose stop
$ docker-compose rm -f
$ docker-compose up -d
```

Si vous retournez sur la page web Gogs, vous devriez voir la version changer :

© 2017 Gogs Version: 0.11.4.0405 Page: 1ms Template: 1ms

Bien entendu, Docker ne rend pas cette mise à jour magique. Il convient de tester cet upgrade dans un environnement de test pour s'assurer que la nouvelle version n'impacte pas le schéma de la base de données par exemple.