Universidad Nacional de Quilmes

Trabajo Práctico

DRONE

Laboratorio de Redes y Sistemas Operativos

Integrantes:

- Uriel Quevedo
- Jose Luis Cassano

Índice

1. ¿Qué es Drone y cúal es su utilidad?

Drone es un proyecto de software libre. Está diseñado, a partir de la tecnología de contenedores. El sistema drone se denomina Continuous Delivery, cuya función es facilitar la Integración Continua (CI, de sus siglas en inglés Continuous Integration).

La Integración Continua es una práctica muy importante hoy en dia para cualquier proyecto de software. Por esta razón, este trabajo practica busca promover la creación de tu propio servidor de CI con el sistema drone.

Drone utiliza un archivo YAML, una capa de abstracción superior del docker-compose, que define y ejecuta Pipelines dentro de los contenedores de Docker (Docker containers).

Drone facilita mucho la creación de tu propio servidor de Continuous Delivery. Este provee una completa guía de instalación y configuración. Sin embargo, en el presente documento, daremos nuestra propia guía.

2. Prerrequisitos de la instalación

Para la instalación del servidor de Drone se requiere mínimamente cumplir con los siguientes ítems:

a. Docker

Se requiere esencialmente tener instalado docker, un servicio de contenedores. Este servicio tiene soporte y compatibilidad para una gran variedad de sistemas operativos.

Aca debajo dejamos un link al trabajo práctico Docker de un cuatrimestre anterior de la materia Laboratorio de Redes y Sistemas Operativos, el cual posee una guia completa de como instalar Docker.

http://www.interorganic.com.ar/josx/dockerio.pdf

b. Aplicación OAuth

Otro prerrequisito de instalación del servidor drone, es tener registrada una aplicación OAuth, en el dominio donde se va a hacer uso del servidor de Drone. Esto es necesario para que Drone pueda conectarse a un servicio de control de versionado de código(Github/Gitlab) y poder realizar su función de Continuous Delivery.

Nosotros solo contemplaremos Github y Gitlab. Cuidado, ya que uno no se puede utilizar para el otro.

Elegir con cautela porque esto va a definir con cual dominio se conectará el servidor de Drone.

i. Github

Instrucciones:

- Ir a https://github.com/settings/developers con un usuario registrado de Github.
- Apretar al botón new OAuth App.
- En Application name poner el nombre de nuestra aplicación. Ej: Drone.
- En Homepage URL se completa con el nombre o el IP del dominio donde se va a hostear el servidor de Drone.
  Si se va a realizar sin hosting, buscar cual es tu IP pública.
  Ej: http://182.156.255.231
- Por último, en Authorization callback URL ponemos lo que pusimos en Homepage URL agregando /login al final.
  Ej: http://182.156.255.231/login

- Le daremos al botón de Register Application y listo. Tendremos la aplicación OAuth registrada en Github.

ii. Gitlab

Instrucciones:

- Ir a https://gitlab.com/profile/applications con un usuario registrado de Gitlab.
- En Name poner el nombre de la Aplicación. Ej: Drone.
- En Redirect URI se completa con el nombre o el IP del dominio donde se va a hostear el servidor de Drone.
  Si se va a realizar sin hosting, buscar cual es tu IP pública.
  Ej: http://182.156.255.231
- Debajo, dentro de la sección Scopes, seleccionamos únicamente las opciones api y read_user.
- Le daremos al botón de Save Application y listo. Tendremos la aplicacion OAuth registrada en Gitlab.

3. Instalación del servidor drone

Para la instalación y el funcionamiento del servidor drone, utilizaremos Docker. Ya que, según los propios desarrolladores, facilitan la instalación y configuración de este con una imagen Docker, que se encuentra el repositorio de imágenes oficial de Docker.

a. Descarga de la imagen Docker

Básicamente, abrimos un terminal en Linux o un Powershell en Windows (asumiendo que Docker está instalado, ya que es un prerrequisito).

Luego, utilizamos el siguiente comando:

```
docker pull drone/drone:1
```

b. Configuración y ejecución del servidor

Para la configuración y ejecución del servidor de Drone, utilizaremos el siguiente comando, en Powershell o terminal bash:

Según la opción elegida en la sección [2.b]:

Comando si elegiste Github:

```
docker run \
  --volume=/var/run/docker.sock:/var/run/docker.sock \
  --volume=/var/lib/drone:/data \
  --env=DRONE_GITHUB_SERVER=https://github.com \
  --env=DRONE_GITHUB_CLIENT_ID={% your-github-client-id %} \
  --env=DRONE_GITHUB_CLIENT_SECRET={% your-github-client-secret %} \
  --env=DRONE_RUNNER_CAPACITY=2 \
  --env=DRONE_SERVER_HOST={% your-drone-server-host %} \
  --env=DRONE_SERVER_PROTO={% your-drone-server-protocol %} \
  --env=DRONE_TLS_AUTOCERT=true \
  --publish=80:80 \
  --publish=443:443 \
  --restart=always \
  --detach=true \
  --name=drone \
  drone/drone:1
```

Comando si elegiste Gitlab:

```
docker run \
  --volume=/var/run/docker.sock:/var/run/docker.sock \
  --volume=/var/lib/drone:/data \
  --env=DRONE_GIT_ALWAYS_AUTH=false \
  --env=DRONE_GITLAB_SERVER=https://gitlab.com \
  --env=DRONE_GITLAB_CLIENT_ID={% your-gitlab-client-id %} \
  --env=DRONE_GITLAB_CLIENT_SECRET={% your-gitlab-client-secret %} \
  --env=DRONE_RUNNER_CAPACITY=2 \
  --env=DRONE_SERVER_HOST={% your-drone-server-host %} \
  --env=DRONE_SERVER_PROTO={% your-drone-server-protocol %} \
  --env=DRONE_TLS_AUTOCERT=false \
  --publish=80:80 \
  --publish=443:443 \
  --restart=always \
  --detach=true \
  --name=drone \
  drone/drone:1
```

Guía de configuración:

La configuración del contenedor se hace a partir de las variables de entorno que se van a instanciar, luego de correr el comando anterior. Debajo dejamos una guía de referencia de la configuración.

Irrelevantes entre Github y Gitlab:

- DRONE_GIT_ALWAYS_AUTH:
  Aca se pone un booleano que configura a Drone si queres que autentique siempre que clone los repositorios públicos. Esto es solo requerido cuando tu source code management system (ej. GitHub Enterprise) tiene el modo privado activado.

  `DRONE_GIT_ALWAYS_AUTH=false`

- DRONE_RUNNER_CAPACITY:
  Aca se pone un entero que define la cantidad máxima de pipelines que el agente deberá ejecutar concurrentemente. El valor por defecto es de dos (2) pipelines.

```
DRONE_RUNNER_CAPACITY=2
```

- DRONE_SERVER_PROTO:

Aca va un String diciendo cual queres que sea el protocolo que use el servidor Drone. Este valor debe ser http o https.

```
DRONE_SERVER_PROTO=https
```

- DRONE_SERVER_HOST:

Esta variable tiene que contener el hostname o dirección IP donde va a estar alojado el servidor. Es importante que este sea idéntico al ingresado en la aplicación OAuth de Github/Gitlab. En mi caso, utilizaré localhost, ya que, tendra mi IP local.

```
DRONE_SERVER_HOST=localhost
```

- DRONE_TLS_AUTOCERT:

Aca va un booleano indicando que los logs de debug deben ser generados y configurados, a través del uso automático de certificados SSL.

```
DRONE_TLS_AUTOCERT=false
```

- Publish:

El servidor va a escuchar en los puertos estándar http y https dentro del contenedor, el cual debe ser publicado en el host (máquina donde tiene el contenedor).

```
 --publish=80:80
--publish=443:443
```

- Volumes
  - **Mount the Docker Socket:**

El servidor requiere acceso al socket de Docker de tu host. Este es usado para correr los pipelines en los contenedores Docker en tu host. Esto

es requerido si se está corriendo Drone con un contenedor solo, es decir, sin Agentes.

```
--volume=/var/run/docker.sock:/var/run/docker.sock
```

- **Mount the Data Volume:**

El servidor crea una base de datos sqlite y persiste el volumen del contenedor en /data. Esto es para prevenir una pérdida de datos. Se recomienda montar los datos en el host cuando se usa la base de datos sqlite por defecto.

```
--volume=/var/lib/drone:/data
```

Github:

Volveremos a https://github.com/settings/applications, y ahí, entraremos en la aplicación OAuth creada con antelación.

Allí se encontrarán la información relevante para completar la siguiente configuración:

- DRONE_GITHUB_CLIENT_ID:
  Un String que contiene su ID de cliente GitHub oauth.

```
DRONE_GITHUB_CLIENT_ID=05136e57d80189bef462
```

- DRONE_GITHUB_CLIENT_SECRET:
  Un String que contiene tu GitHub oauth Client Secret.

```
DRONE_GITHUB_CLIENT_SECRET=7c229228a77d2cbddaa61ddc78d45e
```

- DRONE_GITHUB_SERVER:
  Un String que contiene la dirección de su servidor GitHub. El valor predeterminado es la **https://github.com**dirección oficial del servidor.

```
DRONE_GITHUB_SERVER=https://github.com
```

Gitlab:

Volveremos a https://gitlab.com/oauth/applications, y ahí, entraremos en la aplicación OAuth creada con antelación.

Allí se encontrarán la información relevante para completar la siguiente configuración:

- DRONE_GITLAB_CLIENT_ID:
  Se coloca el String que contiene el ID de cliente de GitLab OAuth.

```
DRONE_GITLAB_CLIENT_ID=05136e57d80189bef462
```

- DRONE_GITLAB_CLIENT_SECRET:
  Se coloca el String que contiene tu GitLab OAuth Client Secret.

```
DRONE_GITLAB_CLIENT_SECRET=7c229228a77d2cbddaa61ddc78d45e
```

- DRONE_GITLAB_SERVER:
  Se coloca el String que contiene su dirección de servidor GitLab. El valor predeterminado es la **https://gitlab.com** dirección oficial del servidor.

```
DRONE_GITLAB_SERVER=https://gitlab.com
```

4. Utilizar el servicio Drone

A continuación, se hará una breve guia de como utilizar el servicio que ofrece Drone, a partir de la ejecución del servidor.

a. SignUp/LogIn (con Github ó Gitlab)

Abriremos un navegador, donde pondremos como URL la dirección de IP o el hostname donde esta alojado el servidor (es decir, la dirección que pusimos en la aplicación OAuth).

Luego, la aplicación te pedirá registrarte con tu cuenta de Github ó Gitlab, según la cual hayas elegido anteriormente. Y te pedirá que permisos de Github/Gitlab queres brindarle a la aplicación.

Por último, te pedirá que ingreses la contraseña del usuario registrado. Y, con esto prácticamente, ya podemos utilizar la aplicación vinculada con nuestro usuario de Github/Gitlab.

b. Administrando tus repositorios

Una vez dentro de la aplicación, donde ya tendremos iniciada la sesión, se podrá localizar todos los repositorios públicos y/o privados (según los permisos dados).

Ahora elegiremos el repositorio donde queremos aplicar la Integración Continua. Hacemos clic en el botón de Activate Repository, y se nos va a habilitar las configuraciones que queremos aplicar al repositorio.

La configuración más importante es la que dice “Configuration” el cual posee un input de String que buscará ese String como el nombre del archivo .YML de configuración Drone, en el directorio root del repositorio.
En la próxima sección diremos, cómo utilizar este archivo de configuración junto al servicio.

c. ¿Cómo utilizar el servicio en mi repositorio?

Una vez habilitado el repositorio a utilizar con el servicio Drone, debemos configurar, en el directorio root del repositorio, el archivo .drone.yml ó el input puesto, en la configuración del repositorio de la aplicación Drone.

El archivo de configuración .yml es una capa de abstracción del docker compose. En este archivo, le da instrucciones al servidor Drone acerca de toda la configuración del contenedor, donde se va a realizar el buildeo del repositorio.

Un ejemplo del archivo .drone.yml en un proyecto gradle con Java de lenguaje:

```
kind: pipeline
name: default

steps:
- name: droneExample
  image: gradle:jdk8
  commands:
  - ./gradlew assemble
  - ./gradlew check
```

5. Problemas de hosting

a. Planteo del Problema

El principal problema que se nos presento con este trabajo fue el hosting del servidor Drone. Esto es porque Github/Gitlab necesita conectarse al servidor, para realizar el webhook y el delivery del repositorio.

Nosotros, al no poseer un server de hosting, decidimos realizar un forwarding de los puertos desde un router nuestro. Pero, no podíamos lograr comunicarnos con el servidor, ya que, nuestro proveedor de Internet hacia un filtro de puertos (creemos que es un Double-Nat). Intentamos comunicarnos con el proveedor, pero debido al poco tiempo, seguimos sin respuesta.

Otra solución que se nos propuso, es utilizar Heroku, una plataforma que hostea aplicaciones. Sin embargo, intentamos de todo para hacerlo correr y no tuvimos suerte. Y buscando en los foros de drone, nos encontramos que unos de los desarrolladores de Drone alias bradrydzewski comento lo siguiente: ***‘’ Heroku is not a supported environment because it runs applications inside limited sandboxes. Drone needs to launch Docker containers, which requires access to the host machine Docker***

daemon, which would not be supported in the Heroku sandbox. **" [Link del comentario]**

Básicamente el desarrollador comenta que un container de servidor Drone, no puede ser levantado por Heroku, ya que este no conoce sus requerimientos, y con ello no se puede configurar correctamente. Tampoco, Heroku permite a Drone tener acceso al daemon de Docker en la máquina host.

b. Solución al problema

Una vez planteado este problema de hosting, decidimos buscar una alternativa "gratis" para hosting. Y una que nos recomendaron fue Ngrok, cuyo servicio posee un pequeño ancho de banda suficiente para resolver este problema.

Para utilizarlo en Linux:

- Ir a https://ngrok.com/ y registrarse.
- Ir a https://dashboard.ngrok.com/get-started y descargarlo para tu distro.
- Una vez descargado, ir a la carpeta donde se localiza el .zip, y descomprimirlo donde quieras ejecutar el bash.
- Antes de ejecutar el script. Vamos a esta pagina https://dashboard.ngrok.com/auth donde deberiamos estar logueados con una cuenta en Ngrok, y copiamos el token que nos brinda la plataforma.
- Una vez con el token copiado, dentro de la carpeta donde descomprimimos el script, ejecutaremos el script con el siguiente comando:

  `./ngrok authtoken tokenCopiado`
- Luego de estar autenticado, crearemos un forwarding del puerto 80 y 443 de nuestro localhost con un nombre de dominio de esta plataforma. Para realizar esto, utilizar el siguiente comando:

  `./ngrok http 80`

- Ahora copiaremos el enlace que nos genera Ngrok, y cambiaremos en nuestra aplicación OAuth Github/Gitlab el dominio que poseiamos antes, y pondremos este que fue generado por Ngrok. Modificaremos los enlaces que posee /login y el que no lo posee.

- Listo ya podemos utilizar nuestro servidor corriendo, en el enlace que nos genero Ngrok, y utilizar el Continuous Delivery de Drone con un navegador.

6. Bibliografía

https://docs.drone.io/
https://github.com/drone/drone
https://docs.drone.io/installation/github/single-machine/
https://docs.drone.io/installation/github/multi-machine/
https://docs.drone.io/installation/gitlab/single-machine/
https://docs.drone.io/installation/gitlab/multi-machine/
https://docs.drone.io/user-guide/
https://docs.drone.io/user-guide/pipeline/steps/
https://docs.drone.io/administration/agents/linux-amd64/
https://drone.io/
https://discourse.drone.io/t/cloud-deployment/1921
https://ngrok.com/
https://dashboard.ngrok.com/get-started
https://dashboard.ngrok.com/auth
https://ngrok.com/docs