Guía para la Instalación de entorno de CI/CD en Rasperry Pi

De Wiki de EGC

Contenido

- 1 Prerrequisitos
- 2 Instalación del SO
- 3 Arranque del sistema
- 4 Configuración inicial
- 5 Instalación de Jenkins
- 6 Configuración de Jenkins
- 7 Configuración de Decide
- 8 Autorecarga de Decide
- 9 Apertura del sistema
- 10 Sincronización con Github
- 11 Finalización
- 12 Extras

Prerrequisitos

- Raspberry Pi Model B (1/2/4/8) GB
- Cargador USB Tipo-C
- Tarjeta Micro SD (16 GB como mínimo)
- Cable Ethernet
- PuTTY "https://www.putty.org/"
- VNC Viewer "https://www.realvnc.com/es/connect/download/viewer/"

Instalación del SO

Primero vamos a instalar el sistema operativo en la tarjeta micro SD. Para ello iremos a "https://www.raspberrypi.com/software/" y descargaremos Raspberry Pi OS. Una vez instalado, insertamos la tarjeta SD con un adaptador y seleccionamos:

- Raspberry Pi OS (32-bit)
- Almacenamiento: Seleccionamos la tarjeta micro SD

Le damos a escribir y a esperar. Una vez acabe el proceso, vamos al directorio raiz de la tarjeta y creamos un archivo llamado "ssh" sin extension. Esto habilitará SSH en la raspberry.

Arranque del sistema

1. Insertamos la tarjeta SD en la ranura de la Raspberry. (localizada en el lado contrario de los puertos USB)
2. Conectamos un extremo del cable Ethernet a la Raspberry y el otro extremo a un ordenador.
3. Conectamos el cable de alimentación a la Raspberry.
4. Abrimos PuTTY y nos conectamos a raspberrypi.local

Configuración inicial

Una vez abierta la sesión nos van a pedir usuario y contraseña, que por defecto son usuario: "pi", contraseña: "raspberry". Luego de iniciar sesión en la cuenta principal vamos a escribir:

```
sudo raspi-config
```

Y con las flechas navegamos hasta "3 INTERFACE OPTIONS", habilitamos SSH y VNC. Cerramos la conexión y abrimos VNC Viewer. Nos conectamos de nuevo a raspberrypi.local y ya podremos ver la interfaz gráfica de nuestra máquina. Ahora debemos seguir los pasos de configuración que aparecen en la pantalla, introduciendo la zona horaria, idioma y una contraseña nueva para el usuario pi. Para finalizar configuraremos la red WiFi y dejaremos que el asistente busque e instale las actualizaciones. (Esto puede tardar un rato). Cuando acabe abrimos una terminal, escribimos:

```
$ sudo apt update
$ sudo apt upgrade
```

Para finalizar nos quedaremos con la IP de nuestra máquina con el siguiente comando:

```
$ hostname -I
```

y con esto hecho ya podemos reiniciar la raspberry quitando el cable Ethernet y conectarnos a la IP usando VNC Viewer.

Instalación de Jenkins

Primero necesitamos instalar el JRE de Java para que Jenkins funcione:

```
$ sudo apt install openjdk-11-jre
```

Una vez hecho esto vamos a crear las claves para añadir el repositorio de paquetes de jenkins al repositorio de la raspberry. Lo haremos con:

```
$ curl https://pkg.jenkins.io/debian/jenkins.io.key | gpg --dearmor | sudo tee /usr/share/keyrings/jenkins-archi
$ sudo nano /etc/apt/sources.list.d/jenkins.list
```

Y añadiremos la siguiente línea:

```
deb [signed-by=/usr/share/keyrings/jenkins-archive-keyring.gpg] https://pkg.jenkins.io/debian binary/
```

Luego actualizaremos los paquetes e instalaremos Jenkins:

```
$ sudo apt update
$ sudo apt install jenkins
```

Una vez instalado, Jenkins genera una contraseña inicial que tendremos que dar en el primer arranque. La podemos ver con el siguiente comando:

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Copiamos la clave, entramos a [IP_DE_TU_RASPBERRY]:8080 y la ponemos, luego seleccionamos "Instalar Plugins Recomendados" y esperamos a que acabe la instalación. Jenkins está ahora instalado como servicio del sistema, que es justo lo que queremos. Esto significa que siempre que el sistema arranque jenkins también lo hará.

Configuración de Jenkins

Vamos a instalar los plugins extras que vamos a necesitar, para ello iremos a:

```
Panel de Control > Administrar Jenkins > Administrar Plugins
```

Seleccionamos "Todos los plugins" e instalamos PostBuildScript. Una vez instaladas todas las herramientas necesarias, vamos al menú principal y seleccionamos "Nueva Tarea". Aquí podemos configurar proyectos de distintas maneras, para nuestro caso vamos a utilizar "Crear proyecto de estilo libre".

- En General, seleccionamos "Desechar ejecuciones antiguas".
- En Configurar el origen del código fuente, seleccionamos Git.
- En URL ponemos la URL de nuestro repositorio "https://github.com/your_github_username/decide.git".
- Podemos configurar varias ramas, en mi caso uso "*/master".
- En Disparadores de ejecuciones seleccionamos "GitHub hook trigger for GITScm polling"
- En Ejecutar, damos a añadir nuevo paso, ejecutar linea de comando shell y copiamos el siguiente código:

```
# Create/Activate  virtualenv
python3 -m venv decide-enviroment
. ./decide-enviroment/bin/activate

# Install Requirements
pip install -r requirements.txt
```

```
# Run tests
cd decide
./manage.py test -v 2
```

- En Acciones para ejecutar después vamos a seleccionar "Add generic script file"

```
/usr/sbin/restart-decide.sh (seleccionamos on SUCCESS)
```

Configuración de Decide

Vamos a hacer una build manual, que deberá dar fallo (esto es de forma intencionada). El fallo es que el sistema no detecta un local_settings.py, ya que no hemos configurado el proyecto decide que acaba de descargar jenkins en su workspace. Para ello vamos a realizar los contenidos vistos en la práctica 1 de la asignatura usando como proyecto var/lib/jenkins/workspace/Decide. Para ello vamos:

```
$ cd var/lib/jenkins/workspace

$ sudo apt-get install python3 python3-venv python3-pip postgresql libpq-dev

$ source decide-enviroment/bin/activate

$ sudo pip install wheel
$ sudo pip install -r requirements.txt

$ sudo su - postgres

psql -c "create user decide with password 'decide'"
psql -c "create database decidedb owner decide"
psql -c "ALTER USER decide CREATEDB;"
```

Debememos crear un fichero "local_settings.py" en var/lib/jenkins/workspace/Decide/decide, y configurarlo como hemos visto en la práctica 1:

```
ALLOWED_HOSTS = ["*"]

# Modules in use, commented modules that you won't use
MODULES = [
    'authentication',
    'base',
    'booth',
    'census',
    'mixnet',
    'postproc',
    'store',
    'visualizer',
    'voting',
]

BASEURL = 'http://127.0.0.1:8000'

APIS = {
    'authentication': BASEURL,
    'base': BASEURL,
    'booth': BASEURL,
    'census': BASEURL,
    'mixnet': BASEURL,
    'postproc': BASEURL,
    'store': BASEURL,
    'visualizer': BASEURL,
    'voting': BASEURL,
}

DATABASES = {
    'default': {
```

```
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'decidedb',
        'USER': 'decide',
        'HOST': '127.0.0.1',
        'PASSWORD': 'decide',
        'PORT': '5432',
    }
}

# number of bits for the key, all auths should use the same number of bits
KEYBITS = 256
```

Una vez hecho esto, iremos a var/lib/jenkins/worspace/Decide, y haremos:

```
$ source decide-enviroment/bin/activate
$ cd decide
$ ./manage.py migrate
$ ./manage.py createsuperuser
$ ./manage.py runserver
```

Probamos a hacer una votación para comprobar que todo funciona correctamente, cerramos la terminal y volvemos a jenkins.

Autorecarga de Decide

Ahora vamos a registrar a decide como servicio de sistema. Para ello vamos a crear un fichero decide.sh en /usr/sbin/ con el siguiente contenido:

```
#!/bin/bash
cd /var/lib/jenkins/workspace/Decide
source decide-enviroment/bin/activate
cd decide
./manage.py runserver
```

Lo registramos como servicio de sistema creando un fichero decide.service en /etc/systemd/system/ con el siguiente contenido:

```
[Unit]
Description= Decide auto startup tool
After=network.target

[Service]
Type=simple
ExecStart=/bin/bash /usr/sbin/decide.sh
TimeoutStartSec=0

[Install]
WantedBy=default.target
```

Ahora vamos a darle permisos a jenkins para que pueda ejecutar comandos shell con sudo, que nos hará falta posteriormente. Para ello vamos a ejecutar el siguiente comando:

```
$ sudo visudo /etc/sudoers
```

y justo debajo de la linea:

```
%sudo   ALL=(ALL:ALL) ALL
```

vamos a insertar:

```
jenkins ALL= NOPASSWD: ALL
```

Con todo esto configurado, ya podemos ir a Jenkins y hacer una build manual, que deberá funcionar sin problemas y en 127.0.0.1/8000 podremos ver decide desplegado.

Apertura del sistema

Vamos a instalar Nginx y Ngrok, que son las herramientas que nos van a permitir poner en línea nuestro sistema. Para ello vamos a ejecutar los siguientes comandos:

```
$ sudo apt install nginx

$ sudo apt update
$ sudo apt install snapd
$ sudo reboot

$ sudo snap install core
$ sudo snap install ngrok
```

Arrancamos Ngrok con:

```
$ ngrok http 80
```

Y se nos abrirá una terminal donde debemos copiar la URL que nos proporciona en forwarding, en mi caso es "http://19ab-217-217-114-58.ngrok.io". Ahora vamos a /etc/nginx/sites-enabled, borramos default y creamos un nuevo archivo "decide" sin extensión con el siguiente contenido:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    root /var/www/html;
    server_name _;

    location /jenkins {
        proxy_pass http://127.0.0.1:8080/jenkins;
    proxy_set_header Host  19ab-217-217-114-58.ngrok.io;
    }

    location / {
        proxy_pass http://127.0.0.1:8000;
    proxy_set_header Host  19ab-217-217-114-58.ngrok.io;
    }

}
```

Vamos a /etc/default y modificamos el archivo jenkins, añadiendo el argumento --prefix=/jenkins a la última línea. Nos deberá quedar tal que:

```
JENKINS_ARGS="--webroot=/var/cache/$NAME/war --httpPort=$HTTP_PORT --prefix=/jenkins"
```

Con esto hecho, ya tenemos decide en https://19ab-217-217-114-58.ngrok.io y jenkins en https://19ab-217-217-114-58.ngrok.io/jenkins

Sincronización con Github

Para acabar, vamos a hacer que cuando haya un commit en GitHub se arranque el ciclo CI/CD automáticamente. Este proceso es muy sencillo, solo debemos irnos nuestro repositorio en GitHub, navegar hasta Configuración y seleccionar Webhooks. Creamos uno nuevo con los siguientes datos:

```
http://19ab-217-217-114-58.ngrok.io/jenkins/github-webhook/
application/json
just push the event
```

Añadimos el Webhook y listo, ahora si hacemos un commit en master podremos ver que jenkins ha empezado a trabajar.

Finalización

Con todos los pasos previos realizados correctamente, hemos conseguido nuestra propia herramienta de CI/CD, sin utilizar servicios como Travis CI o Heroku. Para acabar, si la máquina se reinicia o cierras la terminal de Ngrok, la URL quedará inaccesible, por consiguiente debemos entrar en la máquina con VNC, ejecutar:

```
$ ngrok http 80
```

Y deberemos copiar esta URL en el archivo de configuración de Nginx situado en /etc/nginx/sites-enabled. Reiniciamos el servicio de Nginx y ya está todo funcionando de nuevo.

Extras

Te recomiendo ver 2 funcionalidades para añadir a tu Jenkins, son muy rápidas y fáciles de añadir.

1. Envío de correos automáticos una vez ejecutada la build. Con crearte una cuenta de gmail para el proyecto y configurar el plugin de correo puedes hacer que se te envíe el log automáticamente informándote del estado de la build cuando se ejecuta.
2. Sistema y gestión por roles (Se instala con un plugin, es muy sencillo de configurar). Con esta puedes crear una cuenta de usuario para cada miembro del grupo, y tener un registro de las builds y cambios hechos por cada persona en jenkins.

- Se editó esta página por última vez el 11 dic 2021 a las 19:46.
-