Documentación para despliegue de aplicación

Pagina oficial DigitalOcean: https://www.digitalocean.com/

Si se está utilizando un servidor externo como DigitalOcean, cuando creaste el servidor te dieron una dirección IP para él. Puedes abrir una sesión de terminal con tu nuevo servidor con el siguiente comando:

ssh root@<server-ip-address>

Se le pedirá que introduzca una contraseña. Dependiendo del servicio, la contraseña puede haber sido generada automáticamente y mostrada a usted después de crear el servidor, o puede haber dado la opción de elegir su propia contraseña.

**Creación de cuenta:**

se recomienda crear una cuenta de usuario normal para hacer su trabajo de despliegue, y configurar esta cuenta para que inicie la sesión sin usar una contraseña.

Para crear esta cuenta de usuario, inicie sesión en la cuenta root de su servidor utilizando las instrucciones de ssh de la sección anterior, y luego escriba los siguientes comandos para crear el usuario, darle poderes sudo, y finalmente cambiar a él: (en este caso el servidor se llama Ubuntu)

$ adduser --gecos "" ubuntu

$ usermod -aG sudo ubuntu

$ su ubuntu

Luego se va a configurar esta nueva cuenta de ubuntu para utilizar la autenticación de clave pública para que pueda iniciar sesión sin tener que escribir una contraseña.

Deja la sesión de terminal que tienes abierta en tu servidor por un momento, e inicia una segunda terminal en tu máquina local. En esa sesión de terminal, comprueba el contenido del directorio ~/.ssh:

$ ls ~/.ssh

id\_rsa id\_rsa.pub

Si el listado de directorios muestra archivos llamados id\_rsa e id\_rsa.pub como los anteriores, entonces ya se tiene una clave. Si no se tienen estos dos archivos, o si no está el directorio ~/.ssh en absoluto, entonces se necesita crear tu par de claves SSH ejecutando el siguiente comando, también parte del conjunto de herramientas de OpenSSH:

$ ssh-keygen

Es recomendable aceptar todas las configuraciones por defectos. En otro caso si se sabe que se esta aceptando hacerlo de manera customizada.

Después de que este comando se ejecute, se deben tener los dos archivos mencionados anteriormente. El archivo id\_rsa.pub es la clave pública, que es un archivo que se proporciona a terceros como forma de identificación. El archivo id\_rsa es la clave privada, que no debe ser compartida con nadie. Ahora se necesita configurar tu clave pública como un host autorizado en el servidor. En el terminal que has abierto en tu propio ordenador, imprime tu clave pública en la pantalla:

$ cat ~/.ssh/id\_rsa.pub

Esto va a imprimir una secuencia larga de caracteres. Se debe copiar esto al portapapeles y luego volver a la terminal del servidor remoto donde se ejecutan los siguientes c comandos para almacenar la llave pública.

$ echo <paste-your-key-here> >> ~/.ssh/authorized\_keys

$ chmod 600 ~/.ssh/authorized\_keys

El inicio de sesión sin contraseña debería funcionar ahora. La idea es que ssh en su máquina se identificará ante el servidor realizando una operación criptográfica que requiere la clave privada. El servidor entonces verifica que la operación es válida usando la clave pública.

Ahora puede salir de la sesión de ubuntu, y luego de la sesión de root, y luego tratar de ingresar directamente a la cuenta de ubuntu con:

$ ssh ubuntu@<server-ip-address>

**Seguirdad del servidor**

Para minimizar el riesgo de que su servidor se vea comprometido, hay algunos pasos que puede tomar, dirigidos a cerrar una serie de puertas potenciales a través de las cuales un atacante puede obtener acceso.

El primer cambio que se debe hacer es desactivar los inicios de sesión de root a través de SSH. Ahora se tiene acceso sin contraseña a la cuenta de ubuntu, y se pueden ejecutar comandos de administrador desde esta cuenta a través de sudo, así que realmente no hay necesidad de exponer la cuenta de root. Para deshabilitar los inicios de sesión de root, se debe editar el archivo **/etc/ssh/sshd\_config** en el servidor. Probablemente se tienen los editores de texto vi y nano instalados en el servidor y se pueden usar para editar los archivos. Necesitará anteponer a su editor el prefijo sudo, porque la configuración SSH no es accesible a los usuarios normales (es decir, **sudo vi /etc/ssh/sshd\_config).** Necesita cambiar una sola línea en este archivo:

*/etc/ssh/sshd\_config*: Disable root logins.

PermitRootLogin no

El siguiente cambio es en el mismo archivo. Corresponde a deshabilitar los inicios de sesión con contraseña para todas las cuentas. Se tiene un inicio de sesión sin contraseña configurado, por lo que no hay necesidad de permitir contraseñas en absoluto. Para deshabilitar los inicios de sesión con contraseña, cambie la siguiente línea en **/etc/ssh/sshd\_config:**

*/etc/ssh/sshd\_config*: Disable password logins.

PasswordAuthentication no

Una vez que haya terminado de editar la configuración SSH, es necesario reiniciar el servicio para que los cambios surtan efecto:

$ sudo service ssh restart

El tercer cambio que se debe hacer es instalar el firewall. Se trata de un software que bloquea los accesos al servidor en cualquier puerto que no esté explícitamente habilitado:

$ sudo apt-get install -y ufw

$ sudo ufw allow ssh

$ sudo ufw allow http

$ sudo ufw allow 443/tcp

$ sudo ufw --force enable

$ sudo ufw status

Estos comandos instalan ufw, el Firewall sin complicaciones, y lo configuran para que sólo permita el tráfico externo en los puertos 22 (ssh), 80 (http) y 443 (https). Cualquier otro puerto no será permitido.

Instalar dependencias base

El intérprete base de Python probablemente esté preinstalado en tu servidor, pero hay algunos paquetes adicionales que probablemente no lo estén, y también hay algunos otros paquetes fuera de Python que van a ser útiles para crear un despliegue robusto y listo para la producción. Se va a utilizar mysql como base de datos, también postfix que es un agente de transferencia de correo, que utilizaré para enviar correos electrónicos. La herramienta supervisora monitorizará el proceso del servidor Flask y lo reiniciará automáticamente si alguna vez se bloquea, o también si el servidor se reinicia. El servidor nginx va a aceptar todas las peticiones que vengan del mundo exterior, y las reenviará a la aplicación. Por último, se utilizara git como mi herramienta de elección para descargar la aplicación directamente desde su repositorio git.

$ sudo apt-get -y update

$ sudo apt-get -y install python3 python3-venv python3-dev

$ sudo apt-get -y install mysql-server postfix supervisor nginx git

mientras se ejecuta la tercera sentencia de instalación se solicitará elegir una contraseña de root para el servicio MySQL, y también se harán un par de preguntas sobre la instalación del paquete postfix que se pueden aceptar con sus respuestas por defecto.

Si quieres aprender a configurar completamente un servidor de correo electrónico para que pase las pruebas de seguridad estándar, consulta las siguientes guías de Digital Ocean:

* [Postfix Configuration](http://do.co/2FhdIes)
* [Adding an SPF Record](http://do.co/2Ff8ksk)
* [DKIM Installation and Configuration](http://do.co/2HW2oTD)

**Instalar la aplicación**

Ahora se utilizará git para descargar la aplicación del repositorio de GitHub. Para descargar la aplicación en el servidor, asegúrese de estar en el directorio de inicio del usuario de ubuntu y luego ejecute:

$ git clone https://github.com/miguelgrinberg/microblog

$ cd microblog

Para el caso propio de aplicación cambiar las palabras en rojo por el link del repositorio y después del **cd** por el nombre de la carpeta que contiene la aplicación.

Se necesita ahora entonces crear un entorno virtual y en caso de que se tenga un requirements.txt correr el siguiente código

$ python3 -m venv venv

$ source venv/bin/activate

(venv) $ pip install -r requirements.txt

Se van a utilizar tres paquetes que son específicos para este despliegue de producción, por lo que no están incluidos en el archivo de requisitos comunes. El paquete gunicorn es un servidor web de producción para aplicaciones Python. El paquete pymysql contiene el controlador MySQL que permite a SQLAlchemy trabajar con bases de datos MySQL. El paquete cryptography es utilizado por pymsql para autenticar contra el servidor de bases de datos MySQL.

(venv) $ pip install gunicorn pymysql cryptography

Se debe crear un archivo .env, con todas las variables de entorno necesarias:

*/home/ubuntu/microblog/.env*: Environment configuration.

SECRET\_KEY=52cb883e323b48d78a0a36e8e951ba4a

MAIL\_SERVER=localhost

MAIL\_PORT=25

DATABASE\_URL=mysql+pymysql://microblog:<db-password>@localhost:3306/microblog

MS\_TRANSLATOR\_KEY=<your-translator-key-here>

En color rojo esta la carpeta que se debe cambiar en un caso especifico

En color azul están las variables de entorno para emails y traducciones a tiempo real en caso de que se tengan

En amarillo está la llave secreta que se puede configurar con el siguiente comando que la genera

python3 -c "import uuid; print(uuid.uuid4().hex)"

Ahora se necesita establecer la variable de entorno FLASK\_APP en el punto de entrada de la aplicación para permitir que el comando flask funcione, pero esta variable es necesaria antes de que el archivo .env sea analizado, por lo que necesita ser establecida manualmente. Para evitar tener que establecerla cada vez, se debe añadirla al final del archivo **~/.profile** para la cuenta de ubuntu, para que se establezca automáticamente cada vez que me conecte:

$ echo "export FLASK\_APP=microblog.py" >> ~/.profile

Si se cierra la sesión y vuelve a entrar, ahora FLASK\_APP estará configurado. Se puede confirmar que está establecido ejecutando flask --help.

**Configurar mysql**

Para gestionar el servidor de bases de datos se va a utilizar el comando mysql, que debería estar ya instalado en el servidor:

$ sudo mysql -u root

Estos son los comandos que crean una nueva base de datos llamada microblog, y un usuario con el mismo nombre que tiene acceso completo a ella: (cambiar por el nombre real de la base de datos)

mysql> create database microblog character set utf8 collate utf8\_bin;

mysql> create user 'microblog'@'localhost' identified by '<db-password>';

mysql> grant all privileges on microblog.\* to 'microblog'@'localhost';

mysql> flush privileges;

mysql> quit;

Tendrá que reemplazar <db-password> con una contraseña de su elección. Esta será la contraseña para el usuario de la base de datos del microblog, por lo que es una buena idea no utilizar la misma contraseña que seleccionó para el usuario root. La contraseña para el usuario de microblog debe coincidir con la contraseña que incluyó en la variable DATABASE\_URL en el archivo .env. Si la configuración de la base de datos es correcta, ahora debería poder ejecutar las migraciones de la base de datos que crean todas las tablas:

(venv) $ flask db upgrade

**Configurar Gunicorn y el supervisor:**

Gunicorn es un servidor web de Python puro, pero a diferencia del de Flask, es un servidor de producción robusto que es utilizado por mucha gente, y al mismo tiempo es muy fácil de usar. Para iniciar Microblog bajo gunicorn puedes usar el siguiente comando:

(venv) $ gunicorn -b localhost:8000 -w 4 microblog:app

La opción -w configura el número de trabajadores que gunicorn ejecutará. Tener cuatro trabajadores permite a la aplicación manejar hasta cuatro clientes concurrentemente, lo que para una aplicación web suele ser suficiente para manejar una cantidad decente de clientes, ya que no todos ellos están constantemente solicitando contenido. Dependiendo de la cantidad de RAM que tenga el servidor, puede que se necesite ajustar el número de trabajadores para no quedarse sin memoria.

El argumento microblog:app indica a gunicorn cómo cargar la instancia de la aplicación. El nombre antes de los dos puntos es el módulo que contiene la aplicación, y el nombre después de los dos puntos es el nombre de esta aplicación.

El supervisor se usa para correr la aplicación en el background y cuando colapsa la aplicación vuelve y la despliega. También cuando se reinicia el servidor automáticamente se reinicia la aplicación

La utilidad del supervisor utiliza archivos de configuración que le indican qué programas debe supervisar y cómo reiniciarlos cuando sea necesario. Los archivos de configuración deben ser almacenados en /etc/supervisor/conf.d. Aquí hay un archivo de configuración para Microblog, que se va a llamar microblog.conf:

*/etc/supervisor/conf.d/microblog.conf*: Supervisor configuration.

[program:microblog]

command=/home/ubuntu/microblog/venv/bin/gunicorn -b localhost:8000 -w 4 microblog:app

directory=/home/ubuntu/microblog

user=ubuntu

autostart=true

autorestart=true

stopasgroup=true

killasgroup=true

Después de escribir este archivo de configuración, tiene que recargar el servicio de supervisión para que se importe:

$ sudo supervisorctl reload

**Configurar Nginx**

El servidor de la aplicación de microblogs impulsado por gunicorn está ahora ejecutando de forma privada el puerto 8000. Lo que se debe hacer ahora para exponer la aplicación al mundo exterior es habilitar el servidor web de cara al público en los puertos 80 y 443, los dos puertos que se abrieron en el firewall para manejar el tráfico web de la aplicación.

Se va a configurar el puerto 80 para reenviar todo el tráfico al puerto 443, que va a ser encriptado. Así que se debe empezar por crear un certificado SSL. Por ahora se crea un certificado SSL auto firmado, que está bien para probar todo pero no es bueno para un despliegue real porque los navegadores web advertirán a los usuarios que el certificado no fue emitido por una autoridad de certificación de confianza. El comando para crear el certificado SSL es:

$ mkdir certs

$ openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 \

-keyout certs/key.pem -out certs/cert.pem

El comando le va a pedir alguna información sobre la aplicación y sobre el usuario. Esta es la información que se incluirá en el certificado SSL, y que los navegadores web mostrarán a los usuarios si solicitan verlo. El resultado del comando anterior va a ser dos archivos llamados key.pem y cert.pem, que se he colocado en un subdirectorio certs del directorio raíz de Microblog.

Para que un sitio web sea servido por nginx, es necesario escribir un archivo de configuración para él. En la mayoría de las instalaciones de nginx este archivo debe estar en el directorio **/etc/nginx/sites-enabled.** Nginx instala un sitio de prueba en esta ubicación que realmente no se necesita así que se empieza por eliminarlo:

$ sudo rm /etc/nginx/sites-enabled/default

A continuación puede ver el archivo de configuración de nginx para Microblog, que va en /etc/nginx/sites-enabled/microblog:

*/etc/nginx/sites-enabled/microblog*: Nginx configuration.

server {

# listen on port 80 (http)

listen 80;

server\_name \_;

location / {

# redirect any requests to the same URL but on https

return 301 https://$host$request\_uri;

}

}

server {

# listen on port 443 (https)

listen 443 ssl;

server\_name \_;

# location of the self-signed SSL certificate

ssl\_certificate /home/ubuntu/microblog/certs/cert.pem;

ssl\_certificate\_key /home/ubuntu/microblog/certs/key.pem;

# write access and error logs to /var/log

access\_log /var/log/microblog\_access.log;

error\_log /var/log/microblog\_error.log;

location / {

# forward application requests to the gunicorn server

proxy\_pass http://localhost:8000;

proxy\_redirect off;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

location /static {

# handle static files directly, without forwarding to the application

alias /home/ubuntu/microblog/app/static;

expires 30d;

}

}

Después de añadir este archivo, Se debe decirle a nginx que recargue la configuración para activarla:

$ sudo service nginx reload

Y ahora la aplicación debería estar desplegada. En tu navegador web, puedes escribir la dirección IP del servidor y eso conectará con la aplicación. Debido a que está utilizando un certificado auto firmado, obtendrá una advertencia del navegador web, que tendrá que descartar

Después de completar un despliegue con las instrucciones anteriores para el proyecto se sugiere encarecidamente que sustituya el certificado auto firmado por uno real, para que el navegador no advierta a sus usuarios sobre el sitio. Para ello, primero se debe comprar un nombre de dominio y configurarlo para que apunte a la dirección IP del servidor. Una vez que tenga el dominio, se puede solicitar un certificado SSL gratuito de [Let's Encrypt](https://letsencrypt.org/).

**Para Actualizar la aplicación**

(venv) $ git pull # download the new version

(venv) $ sudo supervisorctl stop microblog # stop the current server

(venv) $ flask db upgrade # upgrade the database

(venv) $ flask translate compile # upgrade the translations

(venv) $ sudo supervisorctl start microblog # start a new server