DOCUMENTACIÓN TÉCNICA OT N°4

Detalles de la Documentación

Tipo de Documentación:	Documentación Técnica
Cliente:	MITIC
Interesados:	OEE
Autores:	Freelancers del Paraguay
Fecha de Entrega:	13/03/2024

Versiones y Modificaciones

Versión	Fecha	Autor	Modificaciones
1.0	13/03/2024	Freelancers del Paraguay	Inicio

Contenido

Detalles del Sistema 2
Implementar GeoNode en un servidor local 3
Implementar GeoNode en un servidor de producción 6

Detalles del Sistema

Stack tecnológico

- Python 3.10
- Java 17
- Django 3.2
- PostgreSQL 15.3
- PostGIS 3.3
- GeoServer 2.23
- El servidor webproxy nginx 1.25
- El servidor web HTTP Apache Tomcat 9

Implementar GeoNode en un servidor local

Paso 1 - Configuración de Docker para Ubuntu (22.04)

Instale los siguientes paquetes a nivel de sistema operativo:

```
sudo add-apt-repository universe
sudo apt-get update -y
sudo apt-get install -y git-core git-buildpackage debhelper devscripts python3.10-dev
python3.10-venv virtualenvwrapper
sudo apt-get install -y apt-transport-https ca-certificates curl lsb-release gnupg
gnupg-agent software-properties-common vim
```

A continuación, agregue el repositorio y los paquetes de Docker:

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update -y
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose
sudo apt autoremove --purge
```

Para finalizar agregue su usuario al grupo de Docker:

```
sudo usermod -aG docker ${USER}
su ${USER}
```

Paso 2 – Crear una instancia de geonode-project

Para la siguiente configuración se utilizará el nombre de **my_geonode**, puede modificarlo por el que crea conveniente. A continuación, realice los siguientes pasos:

Clone el repositorio de **GeoNode**:

```
git clone https://github.com/GeoNode/geonode-project.git
```

A continuación, cambie a la rama deseada. Por ejemplo, 4.2.2 para generar un proyecto para GeoNode 4.2.2 (Ultima versión estable)

```
git checkout -b 4.2.2
```

Luego prepare y active un entorno virtual **Python** e instale **Django**.

```
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 my_geonode

pip install Django==3.2.13
```

Debe ver una salida similar a la siguiente, además de visualizar el nombre del entorno virtual:

```
iflores@Geoportal-ot4:~$ mkvirtualenv --python=/usr/bin/python3 my_geonode
created virtual environment CPython3.10.12.final.0-64 in 349ms
  creator CPython3Posix(dest=/home/iflores/.virtualenvs/my_geonode, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/iflores/.local/share/virtualenv)
    added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
virtualenvwrapper.user_scripts creating /home/iflores/.virtualenvs/my_geonode/bin/predeactivate
virtualenvwrapper.user_scripts creating /home/iflores/.virtualenvs/my_geonode/bin/postdeactivate
virtualenvwrapper.user_scripts creating /home/iflores/.virtualenvs/my_geonode/bin/preactivate
virtualenvwrapper.user_scripts creating /home/iflores/.virtualenvs/my_geonode/bin/postactivate
virtualenvwrapper.user_scripts creating /home/iflores/.virtualenvs/my_geonode/bin/get_env_details
(my_geonode) iflores@Geoportal-ot4:~$ pip install Django==3.2.13
Collecting Django==3.2.13
  Downloading Django-3.2.13-py3-none-any.whl (7.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.9/7.9 MB 9.3 MB/s eta 0:00:00
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.2/41.2 KB 8.2 MB/s eta 0:00:00
Collecting pytz
  Downloading pytz-2024.1-py2.py3-none-any.whl (505 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 505.5/505.5 KB 32.6 MB/s eta 0:00:00
Collecting asgiref<4,>=3.3.2
  Downloading asgiref-3.7.2-py3-none-any.whl (24 kB)
Collecting typing-extensions>=4
  Downloading typing_extensions-4.10.0-py3-none-any.whl (33 kB)
Installing collected packages: pytz, typing-extensions, sqlparse, asgiref, Django
Successfully installed Django-3.2.13 asgiref-3.7.2 pytz-2024.1 sqlparse-0.4.4 typing-extensions-4.10.0
```

Por último, cree un nuevo proyecto Django utilizando como plantilla el repositorio de **geonode-project**.

```
django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,env,sample,properties -n monitoring-cron -n Dockerfile my_geonode
```

Debe ver una salida similar a la siguiente:

```
(my_geonode) iflores@Geoportal-ot4:~$ django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,env,sample,properties -n monitoring-cron -n Dockerfile my_geonode
(my_geonode) iflores@Geoportal-ot4:~$ ls
geonode-project  my_geonode
```

Paso 3 – Preparar el archivo .env

GeoNode utiliza un archivo .env donde se configuran las variables de entorno requeridas por **Docker** y el propio **GeoNode**. Una forma sencilla de generar un archivo .env básico es utilizar el comando de utilidad **create-envfile.py**. Si se ejecuta sin argumentos, generará un archivo .env válido para una implementación local.

Como hostname se utilizará **192.168.183.130**, y como contraseña para los servicios **PZv4Fn+y*A**. Puede modificarlos según sus necesidades.

Acceda al directorio **my_geonode** creado con el comando anterior y ejecute el siguiente comando:

```
python create-envfile.py --hostname 192.168.183.130 --geonodepwd PZv4Fn+y*A --
geoserverpwd PZv4Fn+y*A --pgpwd PZv4Fn+y*A --dbpwd PZv4Fn+y*A --geodbpwd PZv4Fn+y*A
```

Si desea modificar alguna de las variables, el comando acepta los siguientes argumentos:

- **https**: Habilitar SSL. Está deshabilitado por defecto.
- **env_type**:
    - Cuando se establece en **prod**, DEBUG está deshabilitado y se solicita la creación de un SSL válido al servidor ACME de Letsencrypt.
    - Cuando se establece en **test**, DEBUG está deshabilitado y se genera un certificado SSL de prueba para realizar pruebas locales.
    - Cuando se establece en **dev**, DEBUG está habilitado y no se genera ningún certificado SSL.
- **hostname**: La URL que servirá a GeoNode (localhost por defecto).
- **email**: El correo electrónico del administrador. Tenga en cuenta que se requiere un correo electrónico real y configuraciones SMPT válidas si **env_type** está configurado en **prod**. Letsencrypt utiliza el correo electrónico para emitir el certificado SSL.
- **geonodepwd**: Contraseña de administrador de GeoNode. Se establece un valor aleatorio si se deja vacío.
- **geoserverpwd**: Contraseña de administrador de GeoNode. Se establece un valor aleatorio si se deja vacío.
- **pgpwd**: Contraseña de administrador de PostgreSQL. Se establece un valor aleatorio si se deja vacío.
- **dbpwd**: Contraseña del rol de usuario de GeoNode DB. Se establece un valor aleatorio si se deja vacío.
- **geodbpwd**: Contraseña del rol de usuario de la base de datos de datos GeoNode. Se establece un valor aleatorio si se deja vacío.
- **clientid**: ID de cliente del cliente GeoNode Oauth2 de Geoserver. Se establece un valor aleatorio si se deja vacío.
- **clientsecret**: clave secreta del cliente GeoNode Oauth2 de Geoserver. Se establece un valor aleatorio si se deja vacío.

Paso 4 – Construir y ejecutar contenedores con Docker

Los siguientes comandos compilan las imágenes de los servicios, y luego, lanzan los contenedores en un modo desacoplado, permitiendo que los servicios corran en segundo plano.

```
sudo docker compose build
sudo docker compose up -d
```

Debe ver una salida similar a la siguiente:

```
[+] Running 9/9
 ✔Container rabbitmq4my_geonode     Running
 ✔Container gsconf4my_geonode       Healthy
 ✔Container db4my_geonode           Healthy
 ✔Container django4my_geonode       Healthy
 ✔Container celery4my_geonode       Running
 ✔Container geoserver4my_geonode    Running
 ✔Container nginx4my_geonode        Running
 ✔Container memcached4my_geonode    Running
 ✔Container letsencrypt4my_geonode  Started
```

En caso de que se presente algún error, reinicie su sesión e intente nuevamente con los siguientes comandos:

```
sudo docker compose down
sudo docker compose build
sudo docker compose up -d
```

Implementar GeoNode en un servidor de producción

Paso 1 – Modificar el archivo .env

En el caso de que desee implementar, por ejemplo, **https://my_geonode.geonode.org/**, deberá realizar los siguientes cambios:

```
DOCKER_ENV=production
SITEURL=https://my_geonode.geonode.org/
NGINX_BASE_URL=https://my_geonode.geonode.org/
ALLOWED_HOSTS=['django',]
GEOSERVER_WEB_UI_LOCATION=https://my_geonode.geonode.org/geoserver/
GEOSERVER_PUBLIC_LOCATION=https://my_geonode.geonode.org/geoserver/
HTTP_HOST=
HTTPS_HOST=my_geonode.geonode.org
HTTP_PORT=80
HTTPS_PORT=443
LETSENCRYPT_MODE=production # Esto utilizará Letsencrypt y el servidor ACME para
generar certificados SSL válidos
```

Estas variables las establece automáticamente el script **create-envfile.py** si se utilizan las variables **--https** y **--hostname**.

```
python create-envfile.py –https –-hostname https://my_geonode.geonode.org
```

Cuando **LETSENCRYPT_MODE** está configurado en **production**, se requiere un correo electrónico válido y un servidor SMTP de correo electrónico para que el sistema genere un certificado válido.

Paso 2 – Reiniciar los contenedores

Siempre que modifique el archivo .env, deberá reconstruir el contenedor:

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

Una vez finalizado al acceder al **hostname** configurado anteriormente, para este ejemplo **192.168.183.130**, debe visualizar la pagina de inicio de la aplicación.

Firmado digitalmente por JUAN CARLOS LOPEZ AGUAYO
Fecha: 2024.03.13 12:55:20 -03'00'