Utilización de las APIs de OpenStack

Proyecto de Innovación

Implantación y puesta a punto de la infraestructura de un cloud computing privado para el despliegue de servicios en la nube

IES Gonzalo Nazareno
Dos Hermanas (Sevilla)

IES Los Albares
Cieza (Murcia)

IES La Campiña
Arahal (Sevilla)

IES Ingeniero de la Cierva
Murcia

Cofinanciado por:

Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

REST (RESTful web API)

- *Representational State Transfer* (REST)
- Utilizado masivamente en Internet para la transferencia automática y controlada de información
- Utiliza HTTP para la comunicación entre el cliente y el servidor
- Se define una URI base en el servidor
- Comunicación entre cliente y servidor:
  - El cliente realiza una petición HTTP (GET, POST, PUT o DELETE)
  - El servidor contesta con un mensaje en un determinado formato (los más usados son XML y JSON)
- Es más sencillo de implementar que otro protocolos como SOAP y está utilizándose de forma muy amplia
- Google code, Yahoo Developer Network, twitter API, ...

APIs de OpenStack

- Los diferentes servicios de OpenStack se comunican entre sí mediante APIs RESTful
- Para cada API se define una URL:

keystone	`http://192.168.222.1:5000/v2.0`
glance	`http://192.168.222.1:9292/v1`
Compute	`http://192.168.222.1:8774/v2`
Volume	`http://192.168.222.1:8776/v1`
EC2	`http://192.168.222.1:8773/services/Cloud`

- Algunos servicios distinguen entre URL para administrar, red privada y red pública
- Estas URLs se especifican durante la configuración de keystone
- La descripción de los métodos de cada API están disponibles en `http://api.openstack.org`

Ejemplo de utilización

- Los propios clientes de OpenStack de línea de comandos utilizan las respectivas APIS:

```
$ nova --debug list
connect: (172.22.222.1, 5000)
send: 'POST /v2.0/tokens HTTP/1.1\r\nHost: 172.22.222.1:5000\r\nContent-Length:124
\r\ncontent-type: application/json\r\naccept-encoding: gzip, deflate\r\naccept: ap
plication/json\r\nuser-agent: python-novaclient\r\n\r\n{"auth": {"tenantName": "te
st", "passwordCredentials": {"username": "user", "password": "testpass"}}}'
reply: 'HTTP/1.1 200 OK\r\n'
connect: (172.22.222.1, 8774)
send: u'GET /v2/aaaaaaaa5894473c8a98f89a895c6b2c/servers/detail HTTP/1.1\r\nHost:
172.22.222.1:8774\r\nx-auth-project-id: test\r\nx-auth-token: e9233fef4ce34ee49f7d
b1aaaaaaa13f\r\naccept-encoding: gzip, deflate\r\naccept: application/json\r\nuser
-agent: python-novaclient\r\n\r\n'
reply: 'HTTP/1.1 200 OK\r\n'
+--------------------------------------+--------+---------------+----------------+
|                  ID                  |  Name  |     Status    |    Networks    |
+--------------------------------------+--------+---------------+----------------+
| b1724bd0-34f4-4bf1-9444-110eb3531602 | demo9  | VERIFY_RESIZE | vlan5=10.0.5.6 |
| e82814aa-fb1d-4c29-81ab-c39f99184413 | demo10 | ACTIVE        | vlan5=10.0.5.3 |
+--------------------------------------+--------+---------------+----------------+
```

Ejemplo de aplicación propia

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import requests
import json
from getpass import getpass
import ConfigParser

def obtener_token(url,user,passwd):
    """
    Recibe usuario y password de Keystone y devuelve el token de sesion
    """
    cabecera1 = {'Content-type': 'application/json'}
    datos = '{"auth":{"passwordCredentials":{"username": "%s", "password": \
            "%s"}, "tenantName":"service"}}' % (user,passwd)
    solicitud = requests.post(url+'tokens', headers = cabecera1, data=datos)
    if solicitud.status_code == 200:
        token = json.loads(solicitud.text)["access"]["token"]["id"]
        return token

url = config.get("keystone","url")
while True:
    adminuser = raw_input("Usuario de Keystone: ")
    adminpass = getpass("Password: ")
    admintoken = obtener_token(url,adminuser,adminpass)
    if len(admintoken) != 0:
        break
```

El ecosistema

- La utilización de APIs propicia la creación de muchas aplicaciones no oficiales
- Es muy fácil comunicarse con cualquier componente de OpenStack, por lo que cualquiera puede implementar una funcionalidad nueva, automatizar procesos o hacer aplicaciones completas.
- Ejemplos: ceilometer, moniker, heat, ...
- La filosofía abierta y libre de OpenStack hace que algunas de estas aplicaciones se incluyan posteriormente en la versión oficial (quantum en Folsom o ceilometer en Grizzly)
- Estilo OpenStack:
  - Python
  - API
  - Integración continua (jenkins)
  - Licencia Apache
  - Github