Eines per a l'administració remota de servidors

18 de març de 2016

1 Synctool

La missió principal d'aquesta eina és la de sincronitzar fitxers de configuració d'un conjunt de servidors remots en base a un repositori local. A l'igual que pssh utilitza eines com ssh o rsync per a la distribució dels fitxers que s'han de sincronitzar. Un cop sincronitzats en els servidors, synctool permet executar comandes del tipus `service dimoni restart`. Pots, a més, customitzar synctool per a executar més comandes.

Nota: **node** és un host dintre d'un grup de computadores o **cluster**.

Per treballar amb synctool hem de fer-ho per ssh sense password. O sigui, del node mestre fins als nodes del clúster ha d'haver autenticació per claus assimètriques. D'igual manera, si volem protegir les claus privades haurem de fer servir l'ssh-agent.

2 Puppet

2.1 Instal·lació

Ens hem de baixar la versió depenent de la nostra distribució:

```
root@master:~# wget https://apt.puppetlabs.com/puppetlabs-release-jessie.deb
root@master:~# dpkg -i puppetlabs-release-jessie.deb
```

La instal·lació del paquet que ens hem baixat ens configura nous repositoris on finalment estàn allotjats els binaris i el codi font de puppet.

```
root@master:/etc/apt/sources.list.d# apt-get install puppetmaster
```

Al final, a part de depèndencies requerides:

```
root@master:/etc/apt/sources.list.d# dpkg -l | grep puppet
ii  puppet-common          3.7.2-4   all   configuration management system
ii  puppetlabs-release     1.0-11    all   ‘‘Package to install Puppet Labs gpg key and ap
ii  puppetmaster           3.7.2-4   all   configuration management system, master service
ii  puppetmaster-common    3.7.2-4   all   configuration management system, master common
```

Pel client fem el mateix procès però en comptes d'instal·lar el paquet `puppetmaster` instal·lem `puppet`. Després configurem el fitxer `/etc/hosts` pròpiament per a que tots dos es vegin per el nom de la màquina.

2.2 Hello World

Hem de crear un fitxer sota de `/etc/puppet/manifiest` amb extensió `.pp`. Aquest fitxer és un fitxer de codi declaratiu de puppet on definim els nostres recursos. Cada recurs és d'un tipus determinat i té un nom. Això és el mínim que ha de tenir. Veurem que pot tenir propietats amb valors (parell de claus separades per `=>`). El codi té la següent aparença:

```
notify{'Hello world':
}
```

Aquest és un recurs de tipus *notify* Ara el que hem de fer és executar-lo:

```
$ puppet apply hello.pp
Notice: Compiled catalog for master.home in environment production in 0.05 seconds
Notice: Hello world
Notice: /Stage[main]/Main/Notify[Hello world]/message: defined 'message' as 'Hello world'
Notice: Finished catalog run in 0.04 seconds
```

Un altra tipus de recurs és *file* el qual serveix per crear un fitxer amb un determinat contingut. En aquest exemple anem a crear un fitxer a `/tmp` que es diu *demo.txt*. El fitxer s'ha de crear si no existeix i el seu contingut ha de ser 'Hello World':

```
usuario@master:/etc/puppet/manifests$ cat demo.pp
file {'/tmp/demo.txt':
    # si el fitxer no exixteix, que el crei
    ensure => present,
    # contingut del fitxer
    content => 'Hello World',
}
usuario@master:/etc/puppet/manifests$ puppet apply demo.pp
Notice: Compiled catalog for master.home in environment production in 0.21 seconds
Notice: Finished catalog run in 0.03 seconds
usuario@master:/etc/puppet/manifests$ ls /tmp
demo.txt
usuario@master:/etc/puppet/manifests$ cat /tmp/demo.txt
Hello Worldusuario@master:/etc/puppet/manifests$
```

2.3 Test Model Client-Servidor

Anem a fer que un agent apliqui un .pp del servidor. Per defecte, el .pp s'ha de dir *site.pp*. És el manifest per defecte que els clients llegeixen del master server. En arrancar l'agent, si tot va bé el procès és el següent (el primer cop que s'executa l'agent s'estableixen els certificats):

- Agent genera claus
- Master li envia certificat de la autoritat certificadora i certificats
- Agent li envia claus per ser signades per la CA
- Master signa claus. Ho fem a mà

- Agent es baixa .pp i executa la coniguració

```
root@client0:~# puppet agent --server master.home --verbose --no-daemonize
 --waitforcert 10
Info: Creating a new SSL key for client0.home
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for client0.home
Info: Certificate Request fingerprint (SHA256):
  6D:AD:C0:57:4E:82:67:79:0F:25:35:39:CF:C1:35:67:ED:12:6C:27:32:87:43:E1:55:8F:65:D9:94:8
Notice: Did not receive certificate
Notice: Did not receive certificate
Notice: Did not receive certificate
Notice: Did not receive certificate
Notice: Did not receive certificate
Notice: Did not receive certificate
Info: Caching certificate for client0.home
Notice: Starting Puppet client version 3.7.2
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for client0.home
Info: Applying configuration version '1457863005'
Notice: /Stage[main]/Main/File[/tmp/demo2.txt]/ensure: created
Notice: Finished catalog run in 0.12 seconds
^CNotice: Caught INT; calling stop
root@client0:~# ls /tmp
demo2.txt
root@client0:~#
```

En el master hem de tenir el *site.pp*:

```
file {'/tmp/demo2.txt':
  # si el fitxer no exixteix, que el crei
  ensure => present,
  # contingut del fitxer
  content => 'Hello World',
  owner => 'root',
  mode => 777,
}
```

En el client, l'avís `Notice: Did not receive certificate` vol dir que no ha rebut el seu certificat signat per el master. Per fer-ho, des del master:

```
root@master:/etc/puppet/manifests# puppet cert sign client0.home
```

Per veure els certificats dels clients:

```
root@master:/etc/puppet/manifests# puppet cert list
```

Al final, en aquest exemple, se'ns haurà creat un fitxer a tmp amb propietari i permisos com els configurats en el *site.pp*.

Problemes

1. que no coincideixin els noms de les màquines amb els noms que s'han posat en els certificats. Podem tocar el paràmetre certname a la configuració del master, o afegir en el `/etc/hosts` com a nom a les màquines el que hem posat en el certificat i en la comanda de l'agent, `--server <posem el nom que surt al certificat>`.

2. que no s'envii el certificat des de l'agent. Tornem a començar: des del master, esborrem el certificat de l'agent i, des del client, igual:

   - Master:

     `root@master:/etc/puppet/manifests# puppet cert clean client0.home`

   - client:

     `root@client0:~# find /var/lib/puppet/ssl -name client0.home.pem -delete`

     I tornem a executar la comanda en l'agent.

3. en el client, l'agent no s'arrenca donant un error. Amb això, hem de tenir clar les següents coses:

   (a) En instal·lar *puppet* en el client tenim un servei *puppet agent* iniciat per defecte.

   (b) Si hi ha un *agent puppet* iniciat no es pot iniciar cap altre, per la qual cosa, hem de parar el que hi hagi arrancat i hem de treure el bloqueig de que s'inicii qualsevol agent amb la comanda `puppet agent --enable`

   (c) Un cop desbloquejat, ja podem executar la comanda per rebre la configuració del master.

2.4 Nodes

Els nodes permeten a un servidor *Puppet* tenir diferents recursos diferenciats per clients. Els nodes es defineixen en el manifest *nodes.pp* el qual s'ha de importar des del fitxer *site.pp*. Exemple del fitxer *nodes.pp*:

```
# el nom que li donem al node correspon al nom de la màquina
node client0 {
  package {'nginx':
    # ens hem d'assegurar que el paquet estigui instal·lat
    # si no ho està, s'instal·la
    ensure => installed,
  }
}

# nodes no definits explícitament
node default {
  notify {'Default node':
  }
}
```

Exemple de fitxer *site.pp*:

```
import 'nodes.pp'
```

Agent *client0*, aleshores, en recuperar la configuració, haurà instal·lat *nginx* (abans no hi era):

```
root@client0:/home/usuario# puppet agent --server master.home --verbose
                                         --no-daemonize --waitforcert 10
Notice: Starting Puppet client version 3.7.2
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for client0.home
Info: Applying configuration version '1458290609'
Notice: /Stage[main]/Main/Node[client0]/Package[nginx]/ensure:
                           ensure changed 'purged' to 'present'
Notice: Finished catalog run in 60.09 seconds
^CNotice: Caught INT; calling stop
root@client0:/home/usuario# dpkg -l nginx
ii  nginx          1.6.2-5+deb8 all          small, powerful, scalable web/prox
```

Si a la propietat *ensure* li donem el valor *absent*, succeirà el contrari.

2.5 Recursos

- **notify**
- **file**
- **package**: defineix un paquet de software a instal·lar o desinstal·lar en un node.
- **service**: Possibles propietats:
  - `ensure`, que pot prendre els valors *stopped* o *running*
  - `enable`, per configurar el seu arranc automátic, que pot prendre els valors *true* o *false*.

3 Dependències entre recursos

Si tenim un recurs de tipus *service* el que volem, per exemple, és que estigui iniciat però sempre i quan tinguem el paquet instal·lat, o que es notifiqui cada canvi en el fitxer de configuració d'aquest servei ja que no es reflectiràn aquests canvis si no es fa, com a mínim, un *reload* del servei.

Qualsevol crida que fem a un recurs des d'un altre recurs, com a regla de sitaxi, hem de començar a nomenar el tipus de recurs amb lletra majúscula i, com si fos un array, entre corxets i cometes, especificar el nom del recurs al que fem referència:

4 bcfg2

[2]

5 Sobre aquest document

Referències

[1] `http://walterdejong.github.io/synctool/`

[2] `http://docs.bcfg2.org/index.html`

[3] `https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software`

[4] `https://forge.puppetlabs.com/`