

Universidad ICESI

Christian López Morcillo
Juan Diego García
Luisa Vivas Mayor
Rodrigo Esteban Rivera

Curso: Sistemas Operativos

Docente: Daniel Barragán C.

Tema: Despliegue automático de un espejo

Correo: daniel.barragan at correo.icesi.edu.co

Objetivos

- Emplear herramientas de aprovisionamiento automático para la realización de tareas en infraestructura
- Instalar y configurar espejos de sistemas operativos en forma automática para el soporte al aprovisionamiento de máquinas virtuales en clústers de cómputo
- Especificar y desplegar ambientes conformados por contenedores virtuales para la realización de tareas específicas

Solución proyecto:

2. Consigne los comandos de linux necesarios para el aprovisionamiento de los servicios solicitados. En este punto no debe incluir archivos Vagrantfile, Dockerfile u otro tipo de fuentes, solo se requiere que usted identifique los comandos o acciones que debe automatizar (15%)

R/

Instalación de aptly:

Para el aprovisionamiento de los servicios solicitados fue necesario instalar Aptly y primero se debe crear las llaves RSA que se emplean para transferir los archivos de forma segura. para esto se debe utilizar los siguientes comandos:

```
$ gpg --gen-key
```

Abrimos una nueva sesión con el fin de generar la entropía:

```
$ cat /dev/urandom
```

Después de generar las llaves, pasamos a importar las llaves que contiene la máquina a la base trustedkeys.

```
$ gpg --no-default-keyring --keyring /usr/share/keyrings/ubuntu-archive-keyring.gpg --export |  
gpg --no-default-keyring --keyring trustedkeys.gpg --import
```

Ahora se agrega el repositorio de Aptly el archivo sources list de la máquina para la instalación de aptly y se realiza mediante el siguiente comando:

```
$ deb http://repo.aptly.info/ squeeze main > /etc/apt/sources.list
```

importamos la llave del servidor Aptly mediante el siguiente comando:

```
# sudo apt-key adv --keyserver keys.gnupg.net --recv-keys 9E3E53F19C7DE460
```

Después de realizar lo anterior podemos instalar apropiadamente como cualquier otro paquete de software:

```
# apt-get update  
# apt-get install aptly
```

Instalación del mirror :

Después de instalar aptly se crea el mirror, mediante el siguiente comando:

```
$ aptly mirror create -architectures=amd64 -filter='Priority (required) | Priority (important) |  
Priority (standard) ' -filter-with-deps mirror-xenial http://mirror.upb.edu.co/ubuntu/ xenial main  
$ aptly mirror update mirror-xenial
```

Este comando permite definir cuáles paquetes se instalarán y el segundo comando permite actualizar el mirror con estas configuraciones.

Seguido de esto realizamos un snapshot del mirror :

```
$ aptly snapshot create mirror-snap-xenial from mirror mirror-xenial
```

Se mezcla los snapshot:

```
$ aptly publish snapshot mirror-snap-xenial  
$ aptly serve
```

Configuración en el cliente:

Ahora bien, para el cliente, se debe agregar la llave pública generada por el mirror:

```
$ apt-key add my_key.pub
```

Luego, se configura para apuntar al mirror y de esta manera poder descargar los paquetes, desde el repositorio:

```
$ echo "deb http://direccionIpMirror:8080/ xenial main" > etc/apt/sources.list
```

Se actualiza el Apt para guardar los cambios.

```
$ apt-get update -y
```

3. Consigne los archivos empleados para el aprovisionamiento automático del espejo, tenga en cuenta agregar comentarios al código con una breve descripción de cada paso (Vagrantfile, Dockerfile y demás fuentes) (20%)

R/

Se crea un archivo denominado `docker-compose.yml` en el directorio del proyecto y se pegó lo siguiente:

```
version: '2.0'
services:
#Cliente de aptly
  aptly_client:
    build:
      context: ./aptly_client
      dockerfile: Dockerfile
    environment:
      - deps=python3,vim-gnome,byzanz
    ports:
      - "8090:80"
    depends_on:
      - aptly_mirror

#Servidor aptly
  aptly_mirror:
    build:
      context: ./aptly_mirror
      dockerfile: Dockerfile
    environment:
      - deps=python3,vim-gnome,byzanz
    expose:
      - "8080"
```

Este archivo Compose define dos máquinas, un cliente y un mirror. En cada uno de ellos se definen los nombres y las características:

- El mirror se construye a través del Dockerfile que se encuentra en la carpeta `aptly_mirror`. Se pasa por una variable un arreglo de variables de los elementos que se van a inyectar en los paquetes del mirror, en este caso `vim-gnome,python3`. También se exponen hacia las máquinas contenedoras el puerto 8080.
- El cliente se construye a través del Dockerfile que se encuentra en la carpeta `./aptly_client..` En caso del puerto lo que se hace es un mapeo del puerto 8080 a través 8090.

Mirror

Por parte del mirror, el Dockerfile se encarga de levantar los servicios para ejecutarse de la forma correcta.

```
#Base image used
FROM ubuntu:16.04

#Se agregan archivos para configurar los mirror.
ADD /files/configuracion_mirror.sh /configuracion_mirror.sh

RUN chmod +x /configuracion_mirror.sh
RUN chmod +x /aptly_expect.sh

#Importar la llave privada al repositorio local.
ADD /files/llave_privada.asc /llave_privada.asc
RUN gpg --import /llave_privada.asc
RUN rm -f /llave_privada.asc
RUN gpg --no-default-keyring --keyring /usr/share/keyrings/ubuntu-archive-keyring.gpg --export | gpg --no-default-keyring --keyring trustedkeys.gpg --import

#Se agregan todas los repositorios para las instalaciones.
ADD /files/aptly.conf /etc/aptly.conf
RUN echo deb http://repo.aptly.info/ squeeze main > /etc/apt/sources.list && \
    echo deb http://co.archive.ubuntu.com/ubuntu/ xenial main restricted >> \
    /etc/apt/sources.list && \
    echo deb http://co.archive.ubuntu.com/ubuntu/ xenial universe >> \
    /etc/apt/sources.list && \
    apt-key adv --keyserver keys.gnupg.net --recv-keys 9E3E53F19C7DE460

#Se actualiza apt-get
RUN apt-get update

#Se instala aptly y sus dependencias
RUN apt-get install aptly -y && \
    apt-get install xz-utils -y && \
    apt-get install bzip2 -y && \
    apt-get install expect -y

#Se configura aptly
RUN aptly mirror create -architectures=amd64 -filter='Priority (required) | Priority (important) | Priority (standard)' -filter-with-deps mirror-xenial \
    http://mirror.upb.edu.co/ubuntu/ xenial main
RUN aptly mirror update mirror-xenial

#Se modifica aptly con array de variables.
CMD ["/configuracion_mirror.sh"]
```

En este punto se debe tener muy en cuenta que se debe agregar una llave privada. Llave fue creada por aparte, pues dentro de un contenedor es imposible generar entropía. Esta llave privada es copiada de la carpeta /files/ al directorio / en el contenedor, así como el archivo aptly.conf al directorio del contenedor /etc/.

Aclaremos que estos pasos fueron realizados en base a la guía de instalación de aptly.

Se importan las llaves que previamente fueron agregadas para luego eliminar el archivo de la llave por seguridad. Se realizan los pasos de instalación de Aptly y sus dependencias. Entre estas dependencias son xz-utils y bzip2.

Se instala Except para crear respuestas automáticas cuando en una función o instancia se pide el ingreso de datos. Se implementa Except agregando a la lista del repositorio del contenedor y se actualiza.

4. Consigne los archivos empleados para la prueba del espejo (Vagrantfile, Dockerfile y demás fuentes) (10%)

Cliente:

Empezando por el lado del cliente, el primer archivo que es ejecutado el Dockerfile en la carpeta del cliente. Este archivo se encuentra la configuración necesaria:

```
FROM httpd:2.2

#Se agrega la llave publica correspondiente al mirror creado en el contenedor aptly
ADD files/llave_publica.asc /tmp
RUN apt-key add /tmp/llave_publica.asc
RUN rm -f /tmp/llave_publica.asc

#Se instalan los paquetes. Install_packages es un escript que comprueba que el
contenedor mirror este disponible antes de descargar los paquetes.
RUN apt-get update
RUN apt-get install curl -y
ADD files/install_packages.sh /install_packages.sh
RUN chmod +x /install_packages.sh

#Se corre la instalacion de los paquetes.
WORKDIR /
CMD ["/install_packages.sh"]
```

Se define el contenedor cliente, se hace una definición de cómo se va a crear el contenedor. Cogemos la imagen del contenedor que ya tiene instalado, en este caso apache httpd:2.2

Agregamos una llave pública para hacer una conexión segura encriptada que luego será usada con una llave privada. La llave privada es importada sobre el mirror y la pública es dada a todos los clientes. Es decir, el cliente la comunicación la encripta con una llave pública y el mirror con una llave privada.

Entonces se copia el archivo llave_publica.asc contenido de la carpeta files en el directorio /tmp del contenedor, seguidamente se agrega a la llave con el comando apt-key y finalmente se elimina el archivo temporal.

Después se copia el archivo `install_packages.sh` al directorio `/` del contenedor.

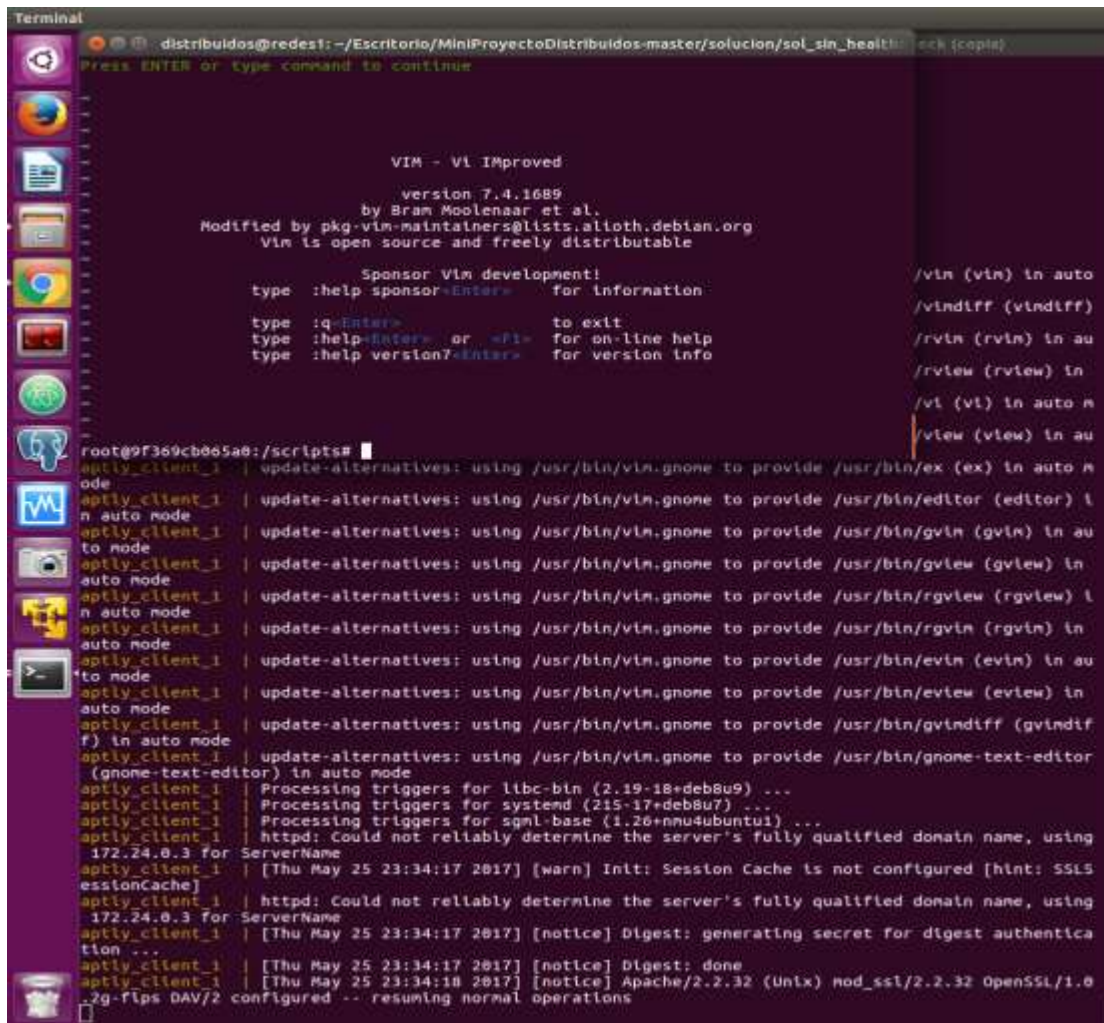
Finalmente, se prueba que el mirror está conectando. Se agrega la dirección del mirror en el directorio `/etc/apt/sources.list` que corresponde a la lista de los repositorios. Luego, se instalan los paquetes se limpia y se actualiza.

6. Incluya evidencias que demuestren que su solución cumple con lo solicitado (15%)

Se muestra evidencia del contenedor cliente apuntado al contenedor mirror:

[illegible]

Se muestra a continuación pantallazo del contenedor cliente ejecutando GVIM uno de los paquetes instalados en el mirror:



```
Terminal
distribuidos@redest:~/Escritorio/MiniProyectoDistribuidos-master/solucion/sol_sin_health:
Press ENTER or type command to continue

VIM - Vi IMproved
      version 7.4.1689
      by Bram Moolenaar et al.
Modified by pkg-vim-maintainers@lists.alioth.debian.org
Vim is open source and freely distributable

Sponsor Vim development!
type :help sponsor<Enter> for Information
type :help<Enter> for on-line help
type :help version7<Enter> for version info

/vim (vim) in auto
/vindiff (vindiff)
/rvim (rvim) in au
/rview (rview) in
/vi (vi) in auto M
/view (view) in au

root@9f369cb065a0:/scripts#
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/ex (ex) in auto M
ode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/editor (editor) l
n auto mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/gvim (gvim) in au
to mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/gvview (gvview) in
auto mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/rgvview (rgvview) l
n auto mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/rgvim (rgvim) in
auto mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/evim (evim) in au
to mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/evview (evview) in
auto mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/gvindiff (gvindif
f) in auto mode
aptly_client_1 | update-alternatives: using /usr/bin/vim.gnome to provide /usr/bin/gnome-text-editor
(gnome-text-editor) in auto mode
aptly_client_1 | Processing triggers for libc-bin (2.19-18+deb8u9) ...
aptly_client_1 | Processing triggers for systemd (215-17+deb8u7) ...
aptly_client_1 | Processing triggers for sgml-base (1.26+nmu4ubuntu1) ...
aptly_client_1 | httpd: Could not reliably determine the server's fully qualified domain name, using
172.24.0.3 for ServerName
aptly_client_1 | [Thu May 25 23:34:17 2017] [warn] Init: Session Cache is not configured [hint: SSL5
essionCache]
aptly_client_1 | httpd: Could not reliably determine the server's fully qualified domain name, using
172.24.0.3 for ServerName
aptly_client_1 | [Thu May 25 23:34:17 2017] [notice] Digest: generating secret for digest authentica
tion ...
aptly_client_1 | [Thu May 25 23:34:17 2017] [notice] Digest: done
aptly_client_1 | [Thu May 25 23:34:18 2017] [notice] Apache/2.2.32 (Unix) mod_ssl/2.2.32 OpenSSL/1.0
.2g-fips DAV/2 configured -- resuming normal operations
```

Además, se incluye en en la carpeta de capturas, 3 videos del aprovisionamiento y las pruebas hechas.

7. Documento algunos de los problemas encontrados y las acciones efectuadas para su solución al aprovisionar la infraestructura y aplicaciones (10%)

R/

1. A la hora de crear una llave pública para el mirror y una llave privada para cada cliente, no se lograba generar una conexión. Ante el problema anterior, se usó un modo promiscuo, es decir, se usaron llaves públicas y de este modo los clientes se lograron autenticar ante el mirror. Se genera en un computador a parte, se importa la pública en los clientes del mirror y la privada en el mirror.
2. Un problema que surgió en el desarrollo del proyecto fué la dificultad de la identificación de la dirección ip de cada contenedor. Lo anterior se soluciona identificando los FQDN (id) que proporciona Docker para cada contenedor.

3. Otro problema fue garantizar que primero se iniciará el mirror antes que el cliente, esto se mitigó, utilizando `depends_on`, lo cual es una utilidad que permite definir el orden de ejecución de los contenedores.
4. Para los programas en los cuales se requería de interacción humana, por ejemplo, crear el mirror, se utilizó la herramienta `expect` que, se encarga de emular la interacción humana.
5. Una dificultad encontrada en el desarrollo fue, como inyectar un arreglo de variables dentro de los comandos de Docker. Se solucionó creando una variable de sistema separada por comas y luego se hizo un `split` para crear un array, se usó el array para realizar los comandos de Docker.