MiniProyectoDistribuidos

Autores:
Andrés Felipe Piñeros
Dylan Torres
Johan David Ballesteros
Códigos: A00273344 - A00265772 - A00309824
Repositorio: https://github.com/DavidPDP/MiniProyectoDistribuidos

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 clústeres de computo.
- Especificar y desplegar ambientes conformados por contenedores virtuales para la realización de tareas específicas.

Pasos Para Automatizar

El problema inicial se basa en poder desplegar una infraestructura de contenedores virtuales que cuente con un *mirror*, el cual pueda almacenar paquetes que defina el usuario y un contenedor cliente que pueda descargar esos paquetes. Para dar solución al problema se decidió utilizar la tecnología *Aptly*, una herramienta que permite administrar repositorios Debian, reflejar repositorios remotos y crear *snapshots*.

Instalar Aptly

Primero se inicia con la creación de las llaves RSA que se utilizaran para la transferencia de archivos de manera segura. Se ejecutan los siguientes comandos. El primer comando se encarga de generar la llave y el segundo de generar la entropía para la llave. Los anteriores pasos no se tendrán en cuenta durante la automatización puesto que se pueden generar y compartir las llaves en el contenedor.

```
$ gpg --gen-key
$ cat /dev/urandom
```

Una vez generadas las llaves procedemos a importar la llave privada a la máquina que contendrá el *mirror*. Con el primer comando importamos una llave externa a la máquina del *mirror* y con el segundo importamos las llaves que tiene la máquina a la base de *trustedkeys*.

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

Procedemos ahora agregar el repositorio de Aptly al archivo sources list de la máquina, que es el archivo donde Apt guarda la lista de repositorios o canales de software, para esto empleamos el siguiente comando.

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

Importamos la llave pública del servidor Aptly para poder descargarlo.

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

Por último, actualizamos Apt e instalamos Aptly.

```
$ apt-get update
$ apt-get install aptly
```

Configurar Mirror

Con Aptly instalado se puede proceder a crear el *mirror*, para esto se ejecuta el siguiente comando. Este comando se compone de un filtro en el cual se especifica los paquetes a instalar (es aquí donde se deberá posteriormente ingresar los paquetes que el usuario defina sin proceder a dejar el comando en *hardcode*), seguido de esto se coloca la opción "*filter-with-deps*" para que se pueda descargar las dependencias de los paquetes definidos si tienen. Luego se asigna un nombre al *mirror* y una URL donde se descargarán los paquetes. El segundo comando actualiza el *mirror*.

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

Para poder publicar los paquetes del mirror se necesita primero realizar un snapshot del mismo.

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

Finalmente se publica el *snapshot* y se inicia el *mirror*.

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

Cliente

Debido a que la comunicación se establece por medio de llaves RSA, lo primero a realizar en el lado del cliente es la importación de la llave pública generada por el *mirror*.

```
$ apt-key add [namePublicKey].asc
```

Se procede apuntar a la URL del *mirror* publicado para poder descargar los paquetes desde ese repositorio.

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

Se actualiza el Apt para que consigne los cambios realizados.

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

Automatización

Mirror

A continuación, se encuentra los enlaces de los archivos utilizados para el aprovisionamiento automático del *mirror*, en cada uno se específica los pasos realizados.

Mirror Dockerfile

Para poder publicar el *snapshot*, Aptly pide la contraseña de la llave privada. Para esto se instaló la herramienta *Expect* la cual permite automatizar las respuestas de las preguntas que se generan en el bash.

Publish Snapshot File

Entry Point

Cliente

A continuación, se encuentra los enlaces de los archivos utilizados para la prueba del *mirror*.

Mirror Dockerfile

Entry Point

Docker Compose

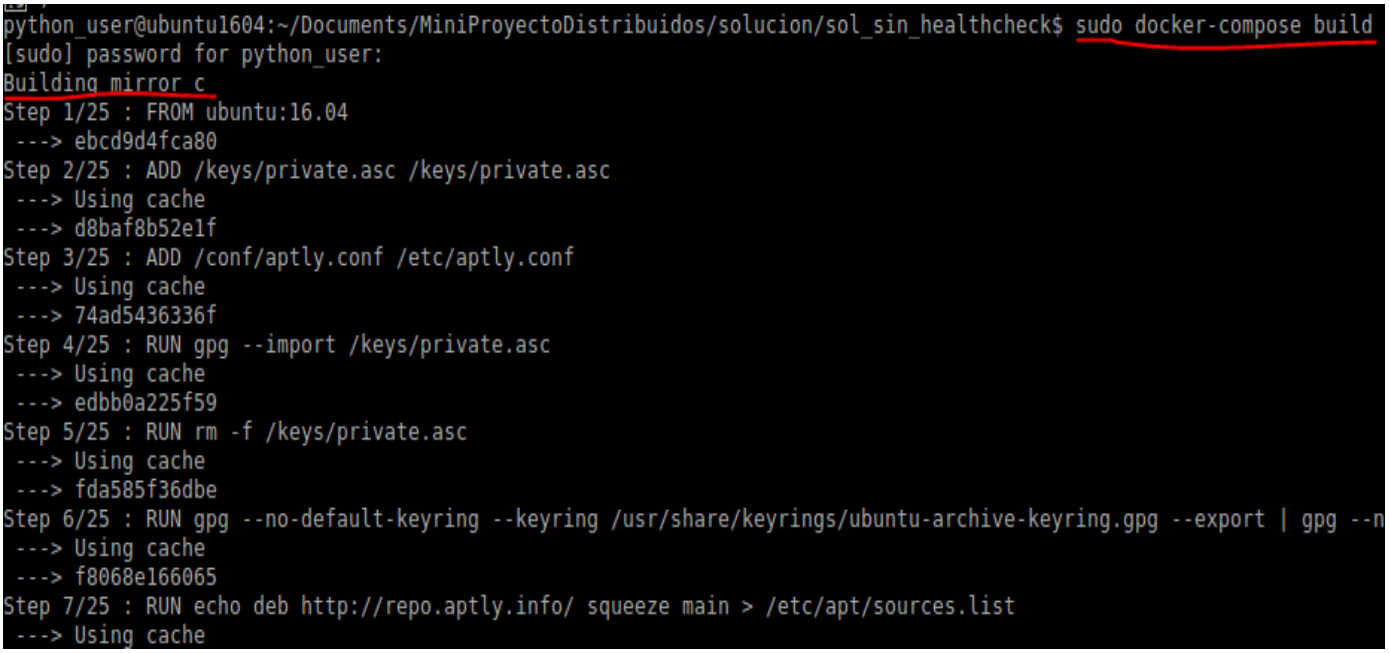
Pruebas Del Funcionamiento

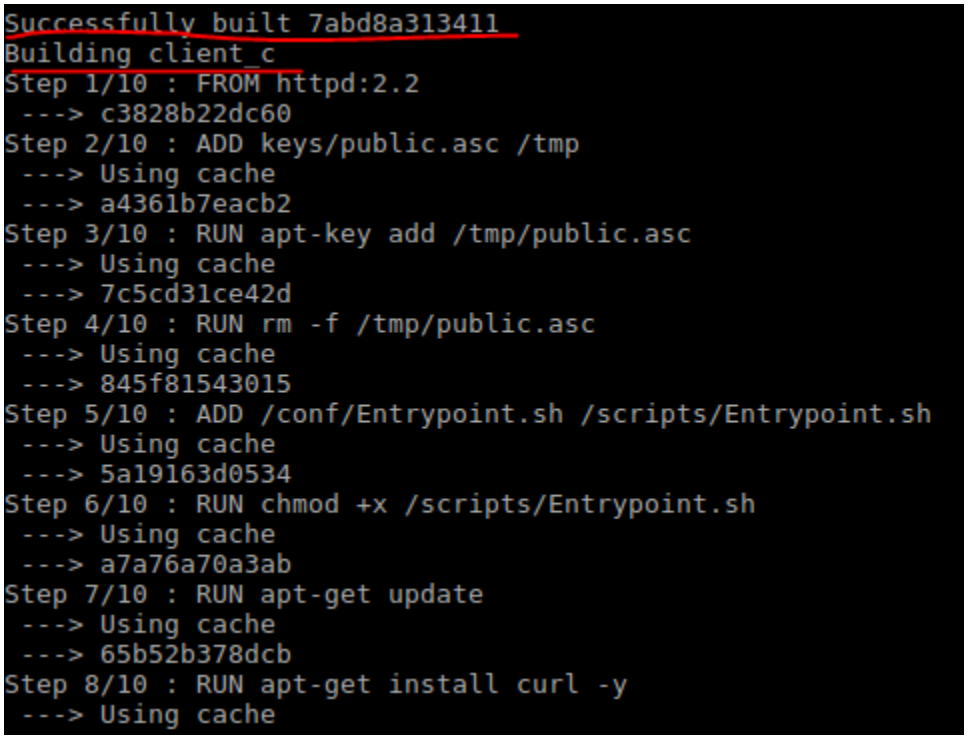
Construcción

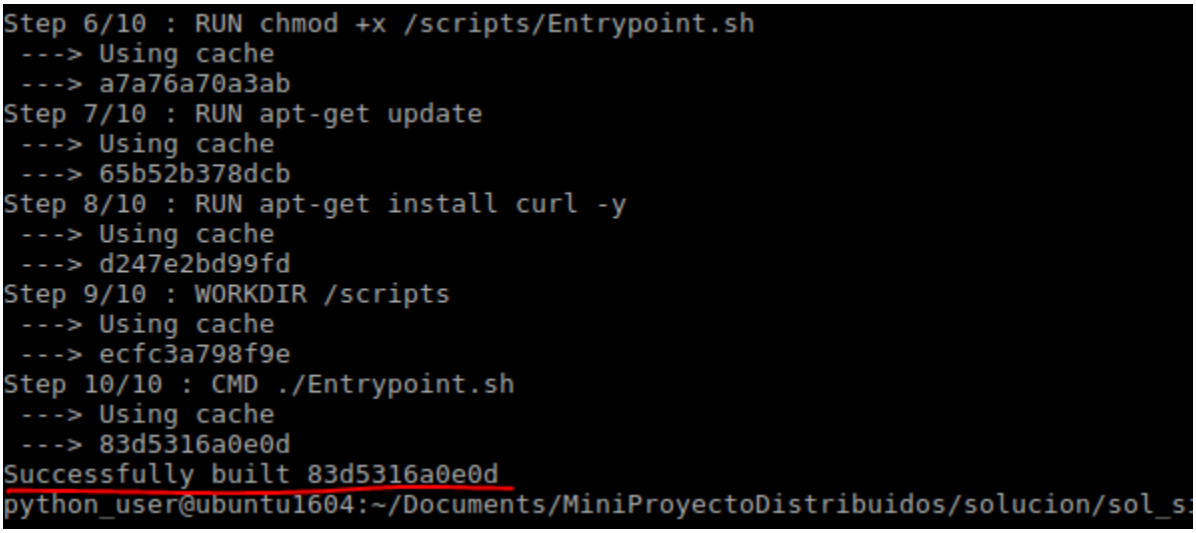
Para verificar el funcionamiento del *mirror* se procede a ejecutar el archivo docker-*compose*, el cual tiene la especificación para construir y desplegar el *mirror* y el cliente.

```
$ docker compose build
```

Se puede comprobar la construcción de los dos contenedores en las siguientes tres imágenes.



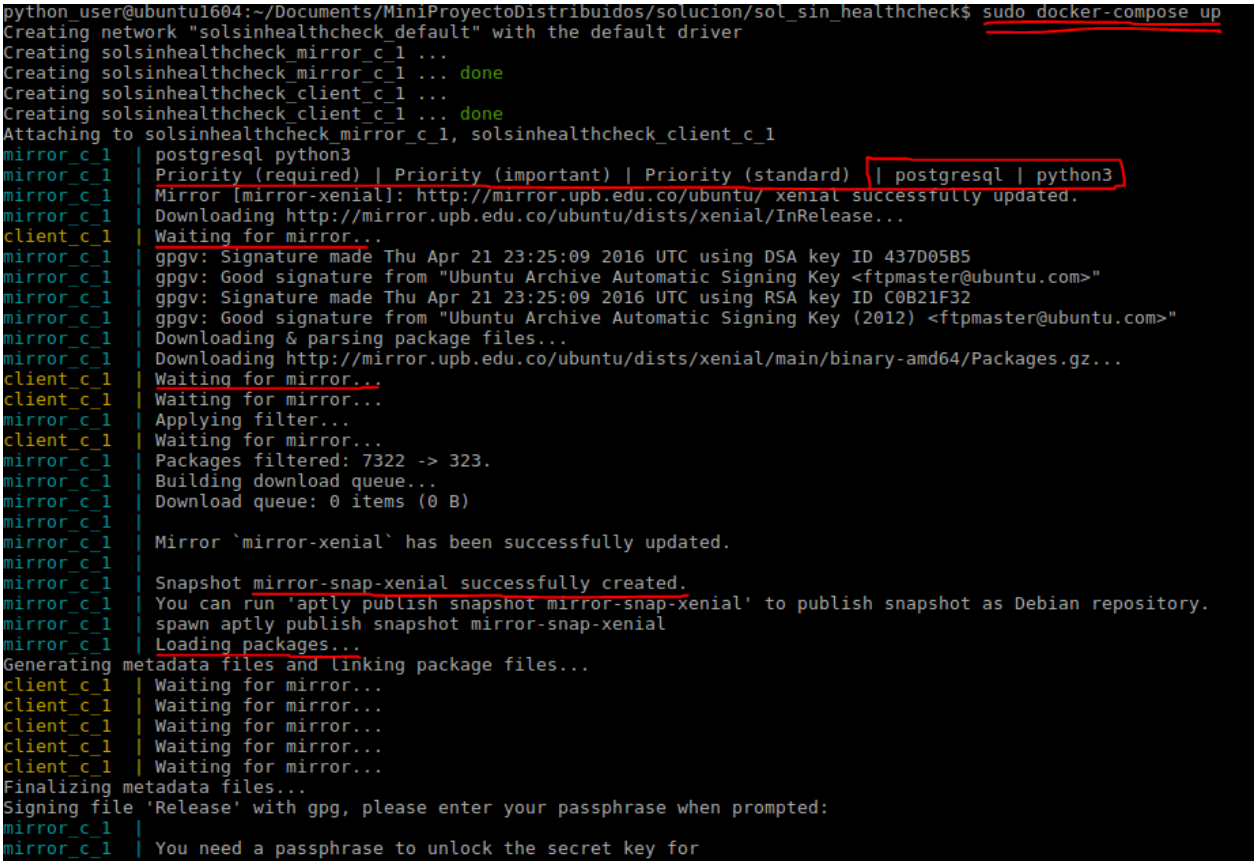




Despliegue

```
$ docker compose up
```

En la primera imagen se puede observar que el filtro que se le pasa al *mirror* para que se actualice contiene las dependencias definidas por el usuario en el docker-compose (python3 y postgresql). También se puede observar como el cliente espera mientras el *mirror* termina de desplegarse.



En la segunda imagen se puede observar como el *mirror* pide la contraseña de la llave privada y posteriormente a esto publica con éxito el *snapshot*, lo que significa que efectivamente con la herramienta Expect se pudo responder a las preguntas sobre el *passphrase* de la llave privada. También se puede observar que cuando inicia el servicio del *mirror*, el cliente sale del estado de espera y entra en el estado de despliegue.

```
mirror_c_1  |
mirror_c_1  | You need a passphrase to unlock the secret key for
mirror_c_1  | user: "Distribuidos"
mirror_c_1  | 2048-bit RSA key, ID 027F4B0F, created 2017-05-22
mirror_c_1  |
mirror_c_1  | gpg: gpg-agent is not available in this session
                r passphrase:
mirror_c_1  | Snapshot mirror-snap-xenial has been successfully published.
mirror_c_1  | Please setup your webserver to serve directory '/aptly/public' with autoindexing.
mirror_c_1  | Now you can add following line to apt sources:
mirror_c_1  |   deb http://your-server/ xenial main
mirror_c_1  | Don't forget to add your GPG key to apt with apt-key.
mirror_c_1  |
mirror_c_1  | You can also use `aptly serve` to publish your repositories over HTTP quickly.
mirror_c_1  | Serving published repositories, recommended apt sources list:
mirror_c_1  |
mirror_c_1  | # ./xenial [amd64] publishes {main: [mirror-snap-xenial]: Snapshot from mirror [mirror-xenial]:
mirror_c_1  | deb http://d6024e2c31b1:8080/ xenial main
mirror_c_1  |
mirror_c_1  | Starting web server at: :8080 (press Ctrl+C to quit)...
client_c_1  | connected...
client_c_1  | Get:1 http://mirror_c:8080 xenial InRelease [2810 B]
client_c_1  | Get:2 http://mirror_c:8080 xenial/main amd64 Packages [79.3 kB]
client_c_1  | Fetched 82.1 kB in 1s (44.9 kB/s)
client_c_1  | Reading package lists...
client_c_1  | Reading package lists...
client_c_1  | Building dependency tree...
client_c_1  | Reading state information...
```

En la tercera y cuarta imagen se puede observar como el cliente descarga las dependencias (paquetes) por medio del host mirror_c que hace referencia al contenedor del *mirror* previamente desplegado. Lo que significa que efectivamente está descargando los paquetes almacenados en ese *mirror*.

```
client_c_1  | Reading state information...
client_c_1  | The following extra packages will be installed:
client_c_1  |   dh-python file libmagic1 libmpdec2 libncurses5 libncursesw5
client_c_1  |   libpython3-stdlib libpython3.5-minimal libpython3.5-stdlib libsqlite3-0
client_c_1  |   libssl1.0.0 libtinfo5 mime-support python3-minimal python3.5
client_c_1  |   python3.5-minimal
client_c_1  | Suggested packages:
client_c_1  |   libdpkg-perl python3-doc python3-tk python3-venv python3.5-venv
client_c_1  |   python3.5-doc binutils binfmt-support
client_c_1  | Recommended packages:
client_c_1  |   libgpm2
client_c_1  | The following NEW packages will be installed:
client_c_1  |   dh-python file libmagic1 libmpdec2 libpython3-stdlib libpython3.5-minimal
client_c_1  |   libpython3.5-stdlib libsqlite3-0 mime-support python3 python3-minimal
client_c_1  |   python3.5 python3.5-minimal
client_c_1  | The following packages will be upgraded:
client_c_1  |   libncurses5 libncursesw5 libssl1.0.0 libtinfo5
client_c_1  | 4 upgraded, 13 newly installed, 0 to remove and 101 not upgraded.
client_c_1  | Need to get 6651 kB of archives.
client_c_1  | After this operation, 29.6 MB of additional disk space will be used.
client_c_1  | Get:1 http://mirror_c:8080/ xenial/main libssl1.0.0 amd64 1.0.2g-1ubuntu4 [1121 kB]
client_c_1  | Get:2 http://mirror_c:8080/ xenial/main libpython3.5-minimal amd64 3.5.1-10 [521 kB]
client_c_1  | Get:3 http://mirror_c:8080/ xenial/main python3.5-minimal amd64 3.5.1-10 [1588 kB]
client_c_1  | Get:4 http://mirror_c:8080/ xenial/main python3-minimal amd64 3.5.1-3 [23.3 kB]
client_c_1  | Get:5 http://mirror_c:8080/ xenial/main mime-support all 3.59ubuntu1 [31.0 kB]
client_c_1  | Get:6 http://mirror_c:8080/ xenial/main libmpdec2 amd64 2.4.2-1 [82.6 kB]
client_c_1  | Get:7 http://mirror_c:8080/ xenial/main libtinfo5 amd64 6.0+20160213-1ubuntu1 [76.8 kB]
```

```
client_c_1  | The following extra packages will be installed:
client_c_1  |   cron distro-info-data gcc-5-base init-system-helpers libbsd0 libc-bin libc6
client_c_1  |   libedit2 libfdisk1 libicu55 libpopt0 libpq5 libsmartcols1 libstdc++6 libxml2
client_c_1  |   locales logrotate lsb-release postgresql-9.5 postgresql-client-9.5
client_c_1  |   postgresql-client-common postgresql-common sgml-base ssl-cert sysvinit-utils
client_c_1  |   ucf util-linux xml-core
client_c_1  | Suggested packages:
client_c_1  |   anacron checksecurity exim4 postfix mail-transport-agent manpages glibc-doc
client_c_1  |   mailx lsb postgresql-doc locales-all postgresql-doc-9.5 sgml-base-doc
client_c_1  |   openssl-blacklist bootlogd sash dosfstools kbd console-tools
client_c_1  |   util-linux-locales debhelper
client_c_1  | Recommended packages:
client_c_1  |   postgresql-contrib-9.5 sysstat
client_c_1  | The following packages will be REMOVED:
client_c_1  |   libapr1-dev libaprutil1-dev libc-dev-bin libc6-dev libexpat1-dev libsctp-dev
client_c_1  |   uuid-dev
client_c_1  | The following NEW packages will be installed:
client_c_1  |   cron distro-info-data gcc-5-base init-system-helpers libbsd0 libedit2
client_c_1  |   libfdisk1 libicu55 libpopt0 libpq5 libxml2 locales logrotate lsb-release
client_c_1  |   postgresql postgresql-9.5 postgresql-client-9.5 postgresql-client-common
client_c_1  |   postgresql-common sgml-base ssl-cert ucf xml-core
client_c_1  | The following packages will be upgraded:
client_c_1  |   libc-bin libc6 libsmartcols1 libstdc++6 sysvinit-utils util-linux
client_c_1  | 6 upgraded, 23 newly installed, 7 to remove and 95 not upgraded.
client_c_1  | Need to get 20.8 MB of archives.
client_c_1  | After this operation, 41.2 MB of additional disk space will be used.
client_c_1  | Get:1 http://mirror_c:8080/ xenial/main libfdisk1 amd64 2.27.1-6ubuntu3 [138 kB]
client_c_1  | Get:2 http://mirror_c:8080/ xenial/main libsmartcols1 amd64 2.27.1-6ubuntu3 [63.4 kB]
client_c_1  | Get:3 http://mirror_c:8080/ xenial/main init-system-helpers all 1.29ubuntu1 [32.1 kB]
client_c_1  | Get:4 http://mirror_c:8080/ xenial/main sysvinit-utils amd64 2.88dsf-59.3ubuntu2 [21.7 kB]
```

En la quinta imagen se puede observar que los contenedores quedaron desplegados correctamente. También se procede a ingresar dentro del contenedor del cliente comprobando que inició el servicio.

```
client_c_1  | httpd: Could not reliably determine the server's fully qualified domain name, using 172.26.0.3 for ServerName
client_c_1  | [Wed May 24 05:18:09 2017] [warn] Init: Session Cache is not configured [hint: SSLSessionCache]
client_c_1  | httpd: Could not reliably determine the server's fully qualified domain name, using 172.26.0.3 for ServerName
client_c_1  | [Wed May 24 05:18:09 2017] [notice] Digest: generating secret for digest authentication ...
client_c_1  | [Wed May 24 05:18:09 2017] [notice] Digest: done
client_c_1  | [Wed May 24 05:18:10 2017] [notice] Apache/2.2.32 (Unix) mod_ssl/2.2.32 OpenSSL/1.0.2g-fips DAV/2 configured -- resuming normal operati
client_c_1  | ::1 - - [24/May/2017:05:34:25 +0000] "GET / HTTP/1.1" 200 44
```

```
python_user@ubuntu1604: ~/Documents/MiniProyectoDistribuidos/solucion/sol_sin_healthc
python_user@ubuntu1604:~/Documents/MiniProyectoDistribuidos/solucion/sol_sin_healthcheck$ sudo docker exec -it client_c_1 /bin/bash
[sudo] password for python_user:
Error response from daemon: No such container: client_c_1
python_user@ubuntu1604:~/Documents/MiniProyectoDistribuidos/solucion/sol_sin_healthcheck$ sudo docker exec -it sol_sin_healthcheck_client_c_1 /bin/ba
Error response from daemon: No such container: sol_sin_healthcheck_client_c_1
python_user@ubuntu1604:~/Documents/MiniProyectoDistribuidos/solucion/sol_sin_healthcheck$ sudo docker ps
CONTAINER ID        IMAGE                       COMMAND                  CREATED             STATUS                    PORTS
cede0fd73bb1        solsinhealthcheck_client_c   "./Entrypoint.sh"        16 minutes ago      Up 16 minutes             80/tcp, 0.0.0.0:8082->8080/tcp
d6024e2c31b1        solsinhealthcheck_mirror_c   "./scripts/Entrypo..."   16 minutes ago      Up 16 minutes             8080/tcp
c298e5f5fd94        solconhealthcheck_mirror_c   "./scripts/Entrypo..."   3 hours ago         Up 3 hours (unhealthy)    8080/tcp
python_user@ubuntu1604:~/Documents/MiniProyectoDistribuidos/solucion/sol_sin_healthcheck$ sudo docker exec -it solsinhealthcheck_client_c_1 /bin/bash
root@cede0fd73bb1:/scripts# ls
Entrypoint.sh
root@cede0fd73bb1:/scripts# curl localhost:80
<html><body><h1>It works!</h1></body></html>root@cede0fd73bb1:/scripts#
```

Dificultades

- Dependencias Del Contenedor: debido a que se busca hacer a los contenedores lo más ligeros posible, encontramos que algunos de los paquetes del sistema son eliminados. En este caso Aptly dependía de dos librerías (xz-utils y bzip2) que fueron eliminadas de la imagen del contenedor (ubuntu:16.04), por lo tanto, debieron instalarse nuevamente.
- Importación De Llaves: debido a que no se tenía en claro el concepto del manejo de las llaves RSA, no sabíamos cómo realizar la importación de las llaves a los contenedores. Finalmente, se investigó sobre el paquete de seguridad GPG el cual nos permitió generar e importar las llaves para poder autenticar la fuente de los paquetes en la interacción *mirror* <---> cliente.
- Sincronización De Despliegue De Contenedores: debido a que el contenedor del cliente obtiene sus dependencias (paquetes) del *mirror*, es necesario desplegar completamente el servicio *mirror* antes de que el cliente inicie la descarga de dependencias. Para esto se investigó sobre dos posibles soluciones: la primera utilizando el comando de docker healthcheck, que es una función nativa de docker, la cual finalmente no se pudo utilizar debido a que no se pudo configurar correctamente el comando de este. La otra solución fue por medio de un script que verificara por medio de la herramienta Curl, que permite diagnosticar si un servicio se encuentra iniciado, si el contenedor del *mirror* ya se había iniciado.