Cómo Montar un Clúster de Nextcloud

Pietro Marini

24 de Junio de 2022

Vigo, esLibre 2022

Detalles prácticos

- Conversación de grupo del taller:

  https://my.ncservice.cloud/call/eo79pkux
- Protegido por contraseña
- Compartiremos código, enlaces, archivos...
- Al entrar en la conversación, habría que poner tu nombre así que todos los demás sepan quien escribe
- Esta presentación se puede descargar desde allí
- Dejaré la conversación abierta unas semanas por si hay preguntas o comentarios

Agenda

1) Objetivos del taller
2) Introducción a la herramienta nc-env
3) Arquitectura de un clúster Nextcloud
4) Despliegue del clúster
5) Configuración y pruebas iniciales
6) Discusión final

Objetivos del taller

Objetivos del taller

- Aprender la arquitectura básica de un clúster de Nextcloud
- Aprender a utilizar la herramienta nc-env
- Montar un clúster en su propio ordenador

La herramienta nc-env

nc-env: Introducción

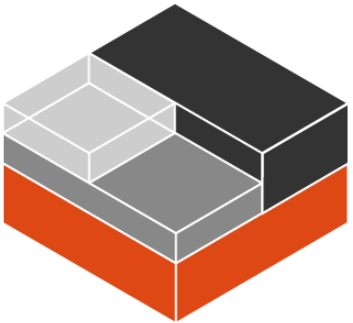
- nc-env permite la creación de entornos Nextcloud en un ordenador Linux local
- Los entornos:
  - pueden ir desde una sencilla instancia de Nextcloud, “todo en uno” hasta un clúster de simulación de la producción con servicios externo añadidos, ej. ElasticSearch
  - se crean /suprimen bajo demanda y no dejan rastro ni tienen impacto en el sistema anfitrión o en otros entornos
  - se pueden aislar/integrar según exigencias (ej. Instancia Nextcloud en un contenedor, SSO en otro y almacenamiento NFS en otro)

nc-env: Casos de Uso

- Prueba de una nueva versión de Nextcloud
- Prueba de una nueva versión de una app de Nextcloud
- Prueba de integración
  - ej. La nueva versión de MariaDB ha sido publicada, ¿Nextcloud X.Y.Z funciona bien con ella?
  - ej. Hay que migrar el sistema de autenticación desde Active Directory a SSO con el IdP Keycloak: cómo se integra Nextcloud con Keycloak?
- Reproducción de problemas de la plataforma en entornos de test y controlados, por ejemplo en espejos de la producción
- Demostraciones de producto
- Validación de procedimientos de instalación y configuración
  - ej. Migración del DB desde PostgreSQL hacia MySQL
  - ej. Modificación de un parámetro del config.php

nc-env: Implementación

- Utiliza Vagrant y LXD:
  - Vagrant: sistema de creación y provisión de contenedores/VM
  - LXD: gestor de contenedores de sistema
- Basado en plantillas (templates), scripts de provisión que instalan y lanzan un componente de la plataforma (ej. servidor web, base de datos, servidor de almacenamiento)



La herramienta nc-env: Templates

- Nextcloud “todo en uno”: servidor web, base de datos, almacenamiento en el mismo contenedor.
- servidor Keycloak
- servidor ElasticSearch
- servidor Collabora Online
- servidor MinIO (almacenamiento S3)
- servidor base de datos MariaDB
- … y más, lista completa en la documentación

nc-env: Creación de un nuevo servicio

- Configuración inicial del entorno en el ordenador anfitrión
- Copia de la plantilla que se quiere utilizar:

```
$ cp -r templates/template01-nextcloud-standalone my-standalone-nc
```

- Copia de los archivos necesarios en la carpeta `artifacts` (cada plantilla tiene un fichero `Readme.md` con las indicaciones)
- Adaptación de los ficheros `provision.sh` y `Vagrantfile` al entorno local, ej. configuración del hostname.
- Despliegue:

```
$ vagrant up
```

nc-env: Ciclo de vida de los entornos

- Una vez creado, el contenedor se puede gestionar a través de LXD o de Vagrant (arrancar, apagar, clonar, suprimir, dar más/menos recursos, configurar la red:

```
$ lxc start my-nc
$ lxc ls
+-------------+---------+----------------------+------+-----------+-----------+
|    NAME     |  STATE  |         IPV4         | IPV6 |   TYPE    | SNAPSHOTS |
+-------------+---------+----------------------+------+-----------+-----------+
| my-nc       | RUNNING | 10.23.46.172 (eth0)  |      | CONTAINER | 0         |
+-------------+---------+----------------------+------+-----------+-----------+
$ vagrant destroy -f
==> default: Stopping machine...
==> default: Destroying machine and associated data...
```

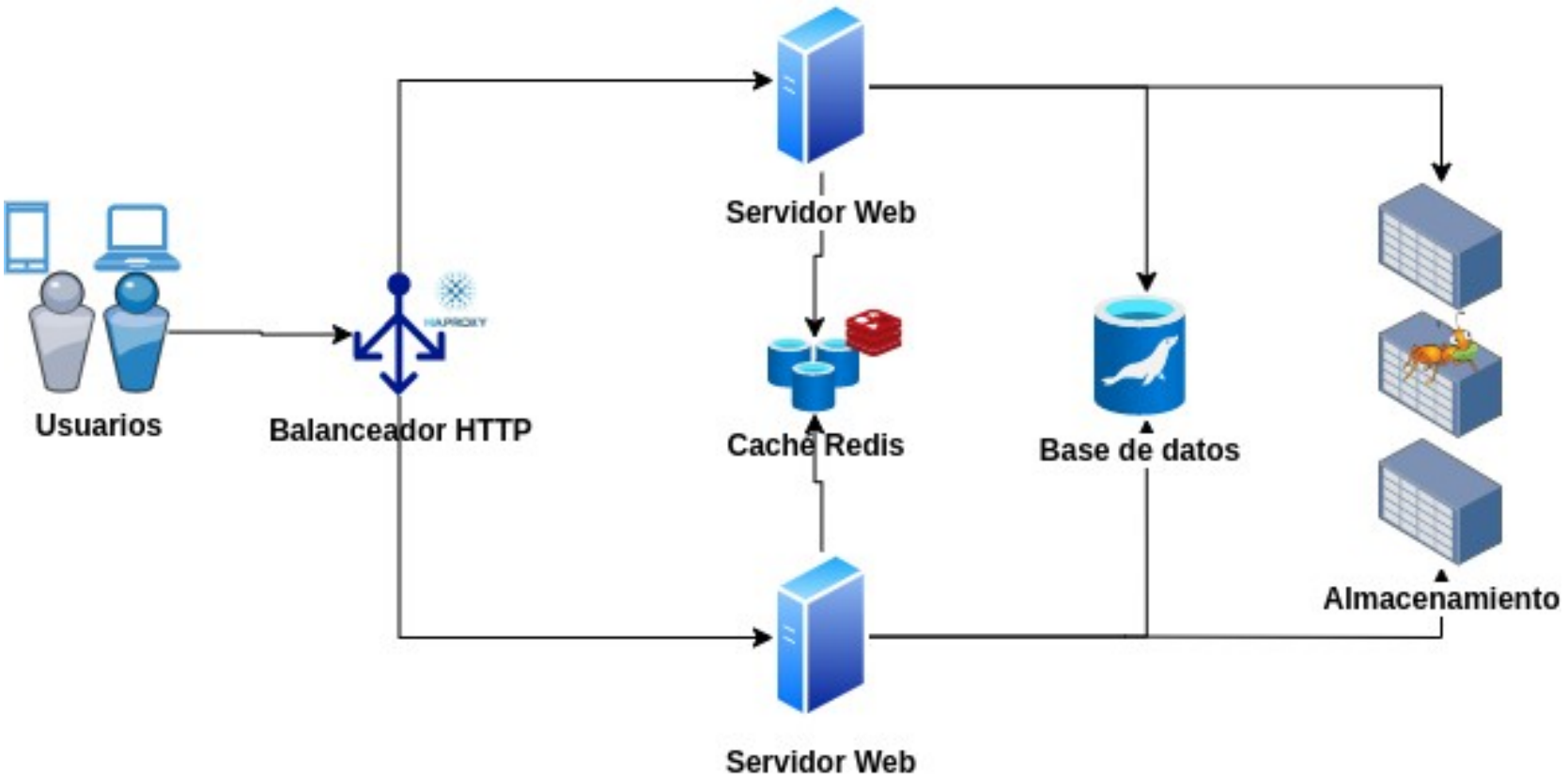
- Tiempo de primer arranque depende de principalmente del disco que se utiliza: ej. en un disco SSD, el Nextcloud “todo en uno” tarda menos de 3 minutos.
- Una vez creados, el arranque de los servicios es instantaneo.

Arquitectura de un clúster Nextcloud

Estructura de un clúster Nextcloud

- Componentes esenciales:
  - Web / Front-end
  - Almacenamiento
  - Base de datos
  - Caché
- Componentes añadidos:
  - Servicio de edición de documentos
  - Servicio de Full Text Search
  - Servicio de video-conferencias
  - Servicio de notificaciones Push (a.k.a. Files HPB)
  - ....

El clúster que montamos hoy



Despliegue del clúster

Preliminar: Comprobaciones y Ajustes del LXD

- En LXD, comprobamos:
  - Interfaz de red lxdbr0: **`$ lxc network show lxdbr0`**
  - Pool de almacenamiento: **`$ lxc storage show <nombre pool>`**
- Creamos un proyecto para nuestro clúster y lo activamos:

```
$ lxc project create nc-cluster -c features.images=false -c features.profiles=false -c
features.storage.volumes=false
Project nc-cluster created
$ lxc project switch nc-cluster
```

- Creamos un contenedor y comprobamos que lo “vemos” con el nombre de dominio que hemos definido en fase de configuración de LXD. Después lo suprimimos.

```
$ lxc launch ubuntu:22.04 c1
$ host c1.localenv.com
c1.localenv.com has address 10.63.8.136
$ lxc ls
+------+---------+---------------------+------+-----------+-----------+
| NAME |  STATE  |        IPV4         | IPV6 |   TYPE    | SNAPSHOTS |
+------+---------+---------------------+------+-----------+-----------+
| c1   | RUNNING | 10.63.8.136 (eth0)  |      | CONTAINER | 0         |
+------+---------+---------------------+------+-----------+-----------+
$ lxc delete -f c1
```

Preliminar: Comprobaciones y Ajustes del LXD

- Creamos un perfil LXD que asignaremos a todos los contenedores:

```
$ lxc profile create nc-cluster-profile
Profile nc-cluster-profile created
$ lxc profile device add nc-cluster-profile root disk path=/ pool=lxdpool01 size=5GB
Device root added to nc-cluster-profile
$ lxc profile device add nc-cluster-profile eth0 nic nictype=bridged name=eth0 parent=lxdbr0
Device eth0 added to nc-cluster-profile
$ lxc profile show nc-cluster-profile
[...]
```

- Repetimos el test de creación de un contenedor pero con el perfil `nc-cluster-profile` asignado:

```
$ lxc launch ubuntu:22.04 c1 -p nc-cluster-profile
$ lxc exec c1 -- sh -c "df -hT"
Filesystem                          Type            Size  Used Avail Use% Mounted on
lxdpool01/containers/nc-cluster_c1  zfs             5.3G  597M  4.7G  12% /
[...]
$ lxc delete -f c1
```

- En el mismo directorio donde hemos expandido al archivo nc-env, creamos la carpeta nc-cluster y la abrimos en el terminal:

```
~/eslibre22 $ ls
nc-env  v20220618.tar.gz
~/eslibre22 $ mkdir nc-cluster
~/eslibre22 $ cd nc-cluster/
```

Etapa 1 – Almacenamiento – GlusterFS

- Copiar la plantilla **`template07-glusterfs-server`**, configurar y arrancar el proceso de creación del contenedor:

```
$ cp -r ../nc-env/templates/template07-glusterfs-server glusterfs-server
$ cd glusterfs-server
$ mkdir log; touch log/provision.log
# Leer y completar los pasos en Readme.md
$ vagrant up > log/provision.log
```

- vagrant-lxd soporta definir un proyecto en el cual crear el contenedor en Vagrantfile desde v0.6.0. Si tienes una versión más antigua, parar el contenedor y moverlo al proyecto nc-cluster, luego re-arrancarlo:

```
$ lxc stop glusterfs-server --project default
$ lxc move glusterfs-server --target-project nc-cluster --project nc-cluster
$ lxc start glusterfs-server
```

Etapa 1 – Almacenamiento – GlusterFS

- Conectarse al contenedor **`glusterfs-server`** y verificar que el servicio glusterd esté disponible y el volumen de glusterd definido y listo para acoger archivos:

```
$ lxc exec glusterfs-server bash

root@glusterfs-server:~# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
     Loaded: loaded (/lib/systemd/system/glusterd.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-05-25 16:48:10 CEST; 20min ago
       Docs: man:glusterd(8)
    Process: 96 ExecStart=/usr/sbin/glusterd -p /var/run/glusterd.pid --log-level $LOG_LEVEL $GLUSTERD_OPTIONS (code=exited,
status=0/S>
   Main PID: 99 (glusterd)
      Tasks: 28 (limit: 37735)
     Memory: 18.8M
        CPU: 1.416s
     CGroup: /system.slice/glusterd.service
             ├─ 99 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO
             └─138 /usr/sbin/glusterfsd -s glusterfs-server.localenv.com --volfile-id vol01.glusterfs-server.localenv.com.srv-
glusterfs>



root@glusterfs-server:~# gluster volume list

vol01
```

Etapa 2 – Base de Datos - MariaDB

- Copiar la plantilla **`template06-nextcloud-db-standalone`**, preparar y arrancar el contenedor:

```
$ pwd
~/eslibre22/nc-cluster
$ cp -r ../nc-env/templates/template06-nextcloud-db-standalone/ db-server
$ cd db-server
$ mkdir log; touch log/provision.log
# Leer y completar los pasos en Readme.md
$ vagrant up > log/provision.log
```

- Si el contenedor no está todavía en el proyecto nc-cluster, pararlo y moverlo a dicho proyecto:

```
$ lxc stop db-server --project default
$ lxc move db-server --project default --target-project nc-cluster
$ lxc start db-server
```

Etapa 2 – Base de Datos - MariaDB

- Conectarse al contenedor **`db-server`** y verificar que la base de datos y las tablas hayan sido creados correctamente:

```
$ lxc exec db-server bash
root@db-server:~# mysql -unextcloud_usr -p
[...]
MariaDB [(none)]> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| nextcloud_db       |
+--------------------+
2 rows in set (0.001 sec)
```

Etapa 3 – Caché - Redis

- Copiar el `template10-redis-server,` luego preparar y arrancar el contenedor:

```
$ pwd
~/eslibre22/nc-cluster
$ cp -r ../nc-env/templates/template10-redis-server/ redis-server
$ cd redis-server
$ mkdir log; touch log/provision.log
# Leer y completar los pasos en Readme.md
$ vagrant up > log/provision.log
```

- Si el contenedor no está todavía en el proyecto nc-cluster, pararlo y moverlo a dicho proyecto:

```
$ lxc stop redis-server --project default
$ lxc move redis-server --project default --target-project nc-cluster
$ lxc start redis-server
```

Etapa 3 – Caché - Redis

- Verificar que el servicio systemd de Redis esté arrancado y listo para responder a las consultas:

```
$ lxc exec redis-server systemctl status redis
● redis-server.service - Advanced key-value store
     Loaded: loaded (/lib/systemd/system/redis-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-05-25 17:43:50 CEST; 9s ago
       Docs: http://redis.io/documentation,
             man:redis-server(1)
    Process: 96 ExecStart=/usr/bin/redis-server /etc/redis/redis.conf (code=exited,
status=0/SUCCESS)
   Main PID: 101 (redis-server)
      Tasks: 4 (limit: 37735)
     Memory: 3.8M
        CPU: 61ms
     CGroup: /system.slice/redis-server.service
             └─101 /usr/bin/redis-server 0.0.0.0:6379

$ lxc exec redis-server redis-cli
127.0.0.1:6379>
127.0.0.1:6379> PING
PONG
```

Etapa 4 – Servidor Web – Apache (nodo 1)

- Copiar, configurar la plantilla `template09-web-server-node` y arrancar el contenedor:

```
$ pwd
~/eslibre22/nc-cluster
$ cp -r ../nc-env/templates/template09-web-server-node/ webserver-node01
$ cd webserver-node01
```

- Editar el fichero `provision.sh` rellenando todos los parámetros necesarios (nombre de dominio de GlusterFS, Redis y base de datos). Luego:

```
$ mkdir log; touch log/provision.log
$ vagrant up > log/provision.log
```

- Si el contenedor no está todavía en el proyecto nc-cluster, pararlo y moverlo a dicho proyecto:

```
$ lxc stop webserver-node01 --project default
$ lxc move webserver-node01 --project default --target-project nc-cluster
$ lxc start webserver-node01
```

Etapa 4 – Servidor Web – Apache (nodo 1)

- Probar que todo esté bien, efectuando los siguientes tests:
  - conectarse a la url: <http://webserver-node01.localenv.com> y efectuar el login con el usuario admin, contraseña admin
  - navegar entre las páginas del aplicativo
  - intentar crear, suprimir, mover ficheros

Etapa 4 – Servidor Web – Apache (nodo 2)

- Copiar el contenedor **webserver-node01** hacia **webserver-node02**:

```
$ lxc copy webserver-node01 webserver-node02
```

- Arrancar el contenedor webserver-node02

```
$ lxc start webserver-node02
```

- Conectarse a **webserver-node02** para cambiar el hostname y reemplazar todas las ocurrencias del antiguo nombre de dominio con el nuevo en el fichero de configuración de Nextcloud

```
$ lxc exec webserver-node02 bash
root@webserver-node01:~# hostnamectl set-hostname webserver-node02.localenv.com
# Reemplazar webserver-node01 con webserver-node02
root@webserver-node01:~#  vim /var/www/nextcloud/config/config.php
root@webserver-node01:~#  reboot
```

- Repetir los testes efectuados en **webserver-node01** para verificar que la configuración es correcta.

Etapa 5 – Balanceador HTTP – HAProxy

- Copiar, configurar la plantilla `template08-haproxy-server` y arrancar el contenedor:

```
$ pwd
~/eslibre22/nc-cluster
$ cp -r ../nc-env/templates/template08-haproxy-server/ haproxy-server
$ cd haproxy-server
$ mkdir log; touch log/provision.log
# Leer y seguir los pasos en Readme.md
$ vagrant up > log/provision.log
```

- Si el contenedor no está todavía en el proyecto nc-cluster, pararlo y moverlo a dicho proyecto:

```
$  lxc stop haproxy-server --project default
$  lxc move haproxy-server --project default --target-project nc-cluster
$  lxc start haproxy-server
```

Etapa 5 – Balanceador HTTP – HAProxy

- Verificar que el servicio systemd del haproxy esté activo:

```
$ lxc exec haproxy-server systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
     Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2022-06-18 17:25:07 CEST; 15s ago
       Docs: man:haproxy(1)
             file:/usr/share/doc/haproxy/configuration.txt.gz
    Process: 105 ExecStartPre=/usr/sbin/haproxy -Ws -f $CONFIG -c -q $EXTRAOPTS (code=exited,
status=0/SUCCESS)
   Main PID: 107 (haproxy)
      Tasks: 17 (limit: 37735)
     Memory: 145.9M
        CPU: 287ms
     CGroup: /system.slice/haproxy.service
             ├─107 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S
/run/haproxy-master.sock

             └─109 /usr/sbin/haproxy -Ws -f  /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S
/run/haproxy-master.sock
```

Configuración y pruebas iniciales

- En ambos nodos web, en el fichero **`config.php`**:
  - Poner los parámetros **`'overwriteprotocol' = 'https'`** y **`'overwritehost' = '<nombre de dominio del balanceador>'`**. Documentación aquí.
  - Añadir el nombre de dominio del balanceador a la lista **`'trusted_domains'`**
  - Añadir parámetro **`'upgrade.disable-web' => true`**
- Conectarse con el navegador a <https://haproxy-server.localenv.com> (o cualquiera nombre de dominio elegido) y entrar en la aplicación con el usuario **`admin`**.
- Crear un grupo y un usuario.
- Revisar y corregir todas las alertas de configuración y seguridad en ajustes.
- Cambiar el agente de Background Jobs a cron. Documentación aquí.
- Conectarse con el usuario y probar el aplicativo: navegación de carpetas, subida/bajada de archivos ...

Configuración y pruebas iniciales

- Instalar la aplicación groupfolders y configurarla. La instalación se tiene que hacer en todos los nodos de servidor web.

```
webserver-node01 $ occ app:install groupfolders
groupfolders 12.0.1 installed
groupfolders enabled
webserver-node02 $ occ app:install groupfolders
```

- Si obtienes el error ”APCu not available for local cache”, hay que añadir **`apc.enable_cli=1`** al fichero **`/etc/php/7.4/mods-available/apcu.ini`**

Gestión del cluster desde LXD

- Obtener la lista de contenedores en un proyecto

```
$ lxc ls
```

- Arrancar / parar un contenedor

```
$ lxc start <nombre contenedor>
$ lxc stop <nombre contenedor>
```

- Arrancar / parar todos los contenedores en el proyecto

```
$ lxc start --all
$ lxc stop -all
```

- Obtener información agregada de los recursos utilizados en el proyecto:

```
$ lxc project info nc-cluster
```

- Obtener información detallada de un contenedor

```
$ lxc info <nombre contenedor>
```

- Fijar limites de utilización de recursos a nivel de instancia/perfil/proyecto

```
#instancia      $ lxc config set <nombre-contenedor> limits.cpu=2
#perfil         $ lxc profile set <nombre-perfil> limits.cpu=1
#proyecto       $ lxc project set <nombre proyecto> limits.containers=n
```

Conclusión

Discusión final

Que hemos logrado en el taller?

- Hemos aprendido la arquitectura de un clúster básico Nextcloud
- hemos montado un clúster básico de Nextcloud utilizando nc-env
- el clúster se puede modificar según necesidad y mejorar con otros elementos

Características de nc-env:

- se ejecuta en local en tu portátil o sobremesa
- basado en LXD y Vagrant
- permite bajar considerablemente el tiempo de pruebas de nuevas funcionalidades y integraciones

¡Gracias!