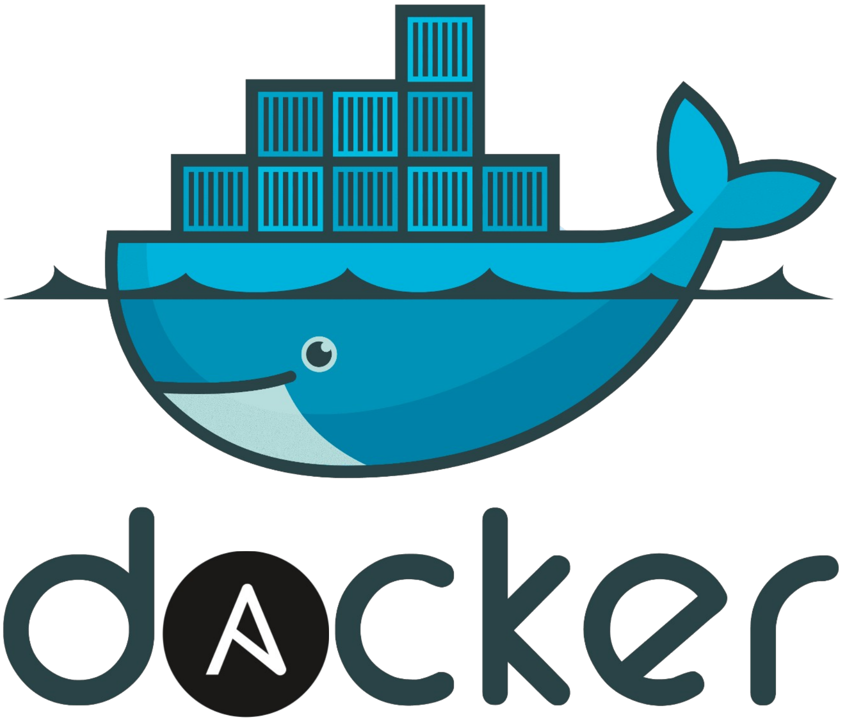
Proyecto Administración de Sistemas Informáticos en Red.

Automatización y Escalado horizontal de contenedores con Docker y Ansible.



2023

Matías Garrido Penni

m.garrido@liceolapaz.net

CPR Liceo la Paz, A Coruña.

ÍNDICE

1. INTRODUCCIÓN

En esta última década se requieren plataformas de servicios web para casi cualquier caso, como son plataformas de películas, series, tiendas, blogs, foros, etc. Y eso conlleva a la creación de equipos (servidores) encargados de mantener esas plataformas activas en todo momento.

Para que estén activas en todo momento hay que tener en cuenta la probabilidad de que el sistema del equipo falle, y en caso de que falle tomar medidas para solucionar rápidamente la caída. Una forma de mantener el servidor activo es aumentando los recursos del hardware (escalado vertical) o agregando más equipos a la red del servidor (escalado horizontal).

Este proyecto busca crear una infraestructura en la que se pueda mantener páginas web como Wordpress, Joomla y Moodle con Nginx manteniendo las páginas web activas mediante escalado horizontal y ofreciendo redundancia. Se usarán contenedores de Docker al igual que herramientas de automatización de instalación de otras herramientas, redes y creación de clústers.

Los beneficios de crear una infraestructura basada en la capacidad de aumentar el número de equipos que pueden ofrecer el servicio y de como recuperarse ante un fallo es que el servicio, que en este caso será una página web, podrá estar activa sin interrupción, disponible las 24 horas del día y los 7 días de la semana a la vez que se aumenta la capacidad de almacenamiento y velocidad de procesamiento gracias al balanceo de carga.

A continuación se mostrarán las herramientas o aplicaciones utilizadas para la creación del proyecto y su mantenimiento.

2. HERRAMIENTAS

Ansible

Es una herramienta de automatización que permite a los usuarios gestionar y configurar de manera eficiente sistemas, redes y aplicaciones de software. Ansible es una herramienta de software libre que utiliza una arquitectura de gestión de configuración sin agentes, lo que significa que no es necesario instalar ningún software adicional en los sistemas que se están gestionando.

Galera

Galera es una tecnología de replicación de bases de datos de alta disponibilidad que se utiliza en sistemas de bases de datos relacionales, como MySQL y MariaDB. La tecnología Galera permite la replicación síncrona multi-maestro, lo que significa que se pueden tener múltiples nodos de base de datos que son capaces de escribir y leer al mismo tiempo.

Docker

Es una herramienta de virtualización de contenedores que permite crear, implementar y ejecutar aplicaciones de manera rápida y fácil en diferentes entornos. Los contenedores de Docker proporcionan un entorno de ejecución aislado y portátil para aplicaciones, se pueden ejecutar en cualquier lugar sin tener que preocuparse por las diferencias en el hardware o el software subyacente.

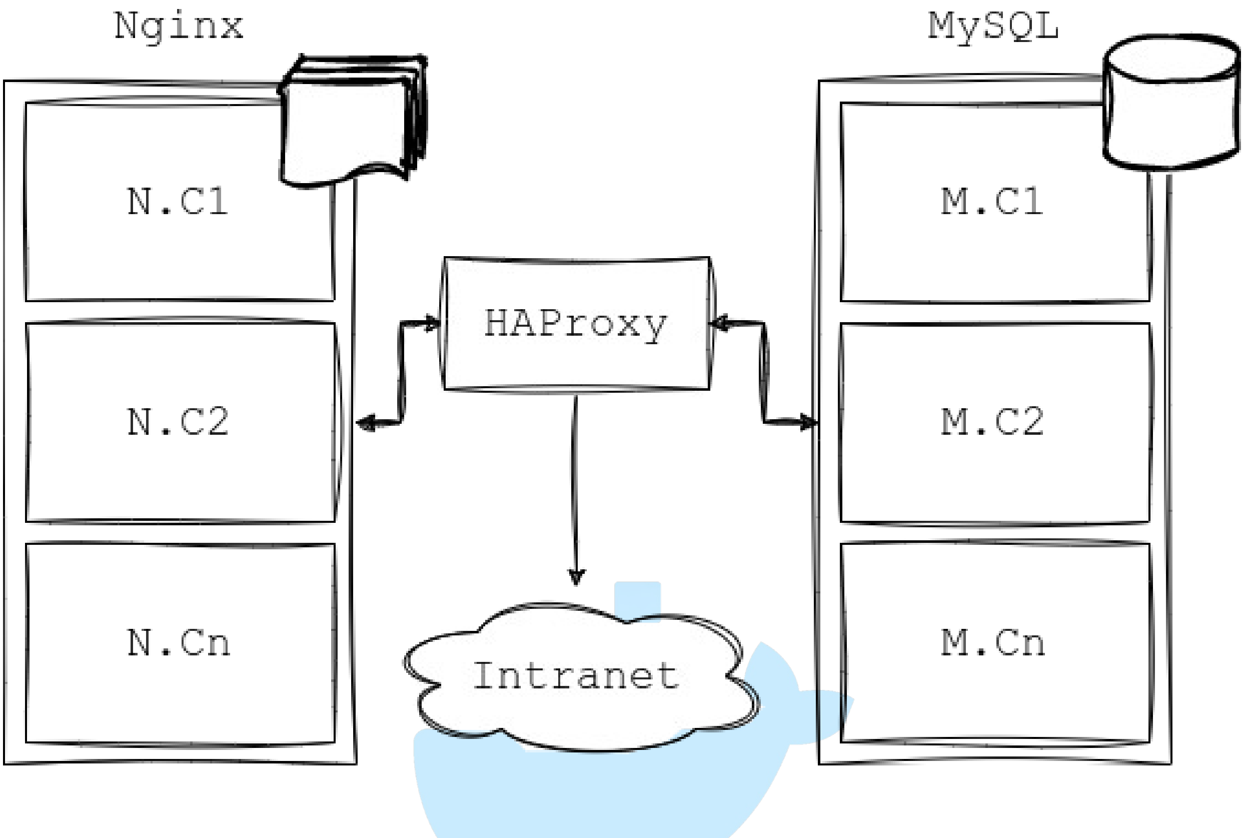
Docker-Compose

Es una herramienta de orquestación de contenedores que permite a los usuarios definir y ejecutar aplicaciones de múltiples contenedores de forma fácil y rápida. Con Docker Compose, se pueden definir varios servicios y contenedores para trabajar juntos como una sola aplicación.

Jinja

Jinja es un motor de plantillas para el lenguaje de programación Python. Utilizado sobre todo para desarrollo web o para crear plantillas de archivos de configuración o, en el caso para este proyecto, plantillas de archivos de Docker-Compose.

3. ESQUEMA



En este proyecto he decidido usar un tipo de nomeclatura específica para nombrar cada contenedor, sería “X.Cn” donde “X” es el servicio (Nginx = N | MySQL = M), “C” Contenedor y “n” el número de contenedor, ya que, aunque en el esquema se muestren tres contenedores por cada servicio, el objetivo principal de este proyecto es conseguir que se pueda escalar horizontalmente la cantidad de veces que sea necesario.

Agregar muchos contenedores aumenta la capacidad de los servicios que se otorgarán, pero no reparte la carga que recibe cada uno, por lo tanto habrá que añadir un balanceador de carga, las dos alternativas mas populares son Nginx y HAProxy, ya que HAProxy es la mas utilizada profesionalmente se ha decidido usar en este proyecto la misma.

4. PROGRESO

Para la creación de este proyecto se utilizará una máquina virtual debido a la sensibilidad de las herramientas que se usarán y de como afecta al sistema tener estos servicios activos. Se usará una máquina virtual Linux con distribución Rocky 9, una distribución hecha por Red Hat que surgió para proporcionar una alternativa sólida y segura a CentOS.

Para que no haya problemas a la hora de escalar los contenedores, se le asignará 6GB de RAM a la máquina virtual, cada contenedor tiene un consumo medio de 150MB de RAM. Los requisitos mínimos son 2GB (300 MB de Sistema, 900MB seis contenedores y el resto servicios activados para el correcto funcionamiento).

El gestor de paquetes por defecto de la mayoría de los equipos de Red Hat es “yum” o “dfn”, antes de instalar los paquetes necesarios para el proyecto se actualizará el gestor de paquetes y se instalará uno de prueba:

```
sudo yum update
sudo yum install epel-release
sudo yum install neofetch
```

```
[matias@rockyLinux ~]$ neofetch
        #####           matias@rockyLinux
       #######          -----------------
       ##O#O##          OS: Rocky Linux 9.1 (Blue Onyx)
       #######          Host: VirtualBox 1.2
     ###########        Kernel: 5.14.0-162.23.1.el9_1.x86_64
    #############       Uptime: 1 min
   ###############      Packages: 775 (rpm)
   ################     Shell: bash 5.1.8
  #################     Resolution: 800x600
#####################   Terminal: /dev/pts/0
#####################   CPU: Intel i3-1005G1 (1) @ 1.190GHz
  #################     GPU: 00:02.0 VMware SVGA II Adapter
                        Memory: 252MiB / 5665MiB
```

Ya que trabajar con una sóla máquina virtual puede provocar fallos, existirán un servidor de ansible y tres ramas para hacer el proyecto: producción (PRD), staging (STG) y laboratorio (LAB). Para que consuma menos cada una de las máquinas virtuales se ha decidido que no tengan interfáz gráfica y que se acceda a ellas mediante SSH.

Ansible

Para instalar Ansible anteriormente se debe de instalar el repositorio Rocky Linux 9 EPEL, usando, en este caso, el gestor de paquetes “dnf”. Se debe instalar en todos los hosts.

```
sudo dnf install epel-release
```

Tras instalar rel repositorio, se podrá instalar Ansible:

```
sudo dnf install ansible
```

Para generar un archivo de configuración de Ansible hay que usar el siguiente comando:

```
ansible-config init --disabled > /etc/ansible/ansible.cfg
```

La opción --disabled permite comentar todas las opciones para luego en el caso de querer una opción simplemente desactivarla.

Ansible sabe a que hosts dirigirse gracias a su archivo de “hosts” el cual se encuentra en /etc/ansible/hosts. Hay que escribir cada uno de los hosts a los que se quiere que se le aplique la configuración con Ansible. Para ello habrá que escribir con privilegios de root, y, en este caso, se usará como editor vi.

```
sudo vi /etc/ansible/hosts
```

Dentro de este archivo habrá una guía y ejemplos que serán útiles, en este caso se escribirá dentro de este archivo las siguientes líneas:

`/etc/ansible/hosts`

```
[maquinas_virtuales]
#rockyLinuxPRD ansible_ssh_host=192.168.56.105
#rockyLinuxSTG ansible_ssh_host=192.168.56.104
rockyLinuxLAB ansible_ssh_host=192.168.56.103
```

Se han comentado los dos primeros hosts ya que, por propósitos de seguridad, se hará todo primero en LAB.

Desde Ansible se pueden ejecutar comandos para los demás hosts o ejecutar modulos específicos, esta es la estructura de uso de comandos:

```
ansible <host> [-m nombre_modulo] [-a args] [opciones]
```

De todos modos, al no configurar la autenticación en ninguno de los equipos no funcionarán los comandos o los modulos que se quieran utilizar. Se empezará creando el usuario, ya que es lo recomendado para mantener la seguridad a la hora de autenticarse y aumenta la facilidad a la hora de ejecutar comandos o enviar reglas; en todas las máquinas habrá que usar los siguientes comandos:

```
sudo useradd ansible
sudo usermod -aG wheel ansible
sudo passwd ansible
```

Luego modificar el archivo de configuración de “sudoers”.

```
sudo visudo
```

```
## Allows people in group wheel to run all commands
# %wheel  ALL=(ALL)       ALL

## Same thing without a password
%wheel        ALL=(ALL)       NOPASSWD: ALL
```

Luego para permitir la autenticación mediante SSH habrá que permitir que Ansible pregunte por la contraseña de la conexión, eso se consigue descomentando una regla en la configuración de Ansible y modificandola a True.

/etc/ansible/ansible.conf

```
ask_pass=True
```

Tras modificar el archivo y guardarlo, se podrá autenticar la conexión entre hosts con Ansible para la ejecución de ordenes, como por ejemplo, la ejecución del módulo ping:

```
[ansible@rockyLinuxANSIBLE ~]$ ansible rockyLinuxLAB -m ping
SSH password:
rockyLinuxLAB | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Ahora la autenticación con contraseña será remplaza por una autenticación con clave privada/pública, mucho más segura. Para crear una clave SSH se usará el comando “ssh-keygen” en la máquina que funciona como servidor Ansible.

```
[ansible@rockyLinuxANSIBLE ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/F8k0l3/BXvGeg1EIeWx8wrJqoqy8kBSptbQZzc19Cw
ansible@rockyLinuxANSIBLE
The key's randomart image is:
+---[RSA 3072]----+
|        .+  ..=. |
|  .     . +  + o |
| .o. o o E o  * .|
| +o o ... .o + B.|
|oo .    S . * = B|
|+        . o + *+|
|.         o   + +|
|...  .   . . . . |
| ooo. ...   .    |
+----[SHA256]-----+
```

Una vez creada la clave pública/privada se copiará a los demás hosts usando:

```
ssh-copy-id ansible@192.168.56.103
```

Posteriormente al copiado de la clave pública/privada en los hosts se deberá de comentar nuevamente la configuración de preguntar contraseña en Ansible.

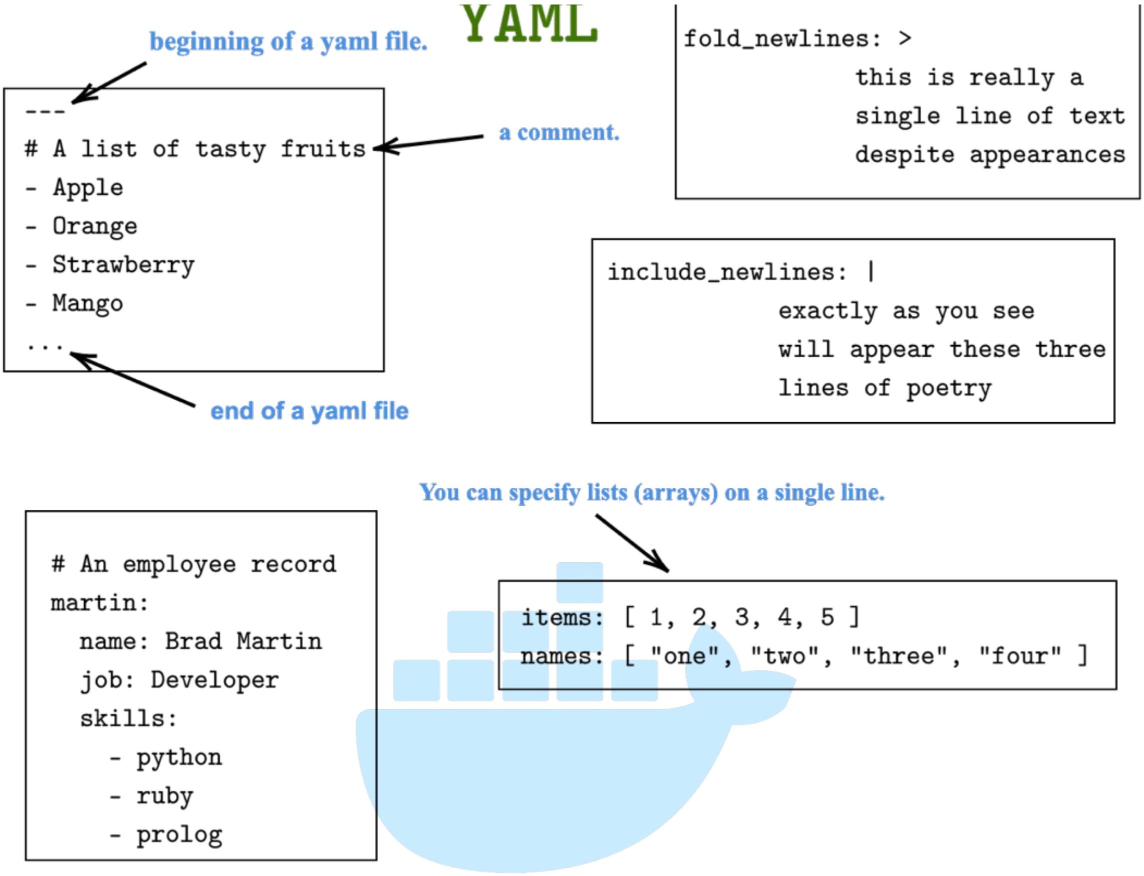
```
/etc/ansible/ansible.conf
```

```
# ask_pass=True
```

Tras hacer esto ya no hará falta especificar la contraseña cuando se ejecute un comando o playbook de Ansible, se mostrará el módulo ping nuevamente para demostrarlo:

```
[ansible@rockyLinuxANSIBLE ~]$ ansible rockyLinuxLAB -m ping
rockyLinuxLAB | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Las playbooks describen políticas para ser aplicadas en los sistemas remotos, para forzar su configuración. Son escritas en un formato de texto fácil de entender que agrupa una serie de tareas juntas, ese formato de texto es YAML.



```
---
# A list of tasty fruits
- Apple
- Orange
- Strawberry
- Mango
...
```

```
fold_newlines: >
            this is really a
            single line of text
            despite appearances
```

```
include_newlines: |
            exactly as you see
            will appear these three
            lines of poetry
```

```
# An employee record
martin:
  name: Brad Martin
  job: Developer
  skills:
    - python
    - ruby
    - prolog
```

```
items: [ 1, 2, 3, 4, 5 ]
names: [ "one", "two", "three", "four" ]
```

Se creará una playbook simple de ejemplo para mostrar la instalación de neofetch.

/home/ansible/.ansible/playbooks/neofetch.yml

```
---
- hosts: maquinas_virtuales
  become: true
  become_user: root

  tasks:

    - name: INSTALAR - Neofetch
      dnf: name=neofetch

    - name: EJECUTAR - Neofetch
      shell: neofetch
```

Esta playbook además de instalar Neofetch ejecutará el comando que se especifica, en este caso, neofetch, en los hosts del grupo “maquinas_virtuales”.

Ahora procede la instalación de Docker utilizando una playbook de Ansible llamada docker.yml, dentro de “/home/ansible/.ansible/playbooks/”.

`/home/ansible/.ansible/playbooks/docker.yml`

```
- name: Instalar Docker
  hosts: maquinas_virtuales
  become: true
  become_user: root
  tasks:

    - name: HABILITAR - Repositorio CRB
      shell: dnf config-manager --set-enabled crb

    - name: INSTALAR - Repositorio EPEL
      dnf:
        name:
          - epel-release
        state: latest
        update_cache: true

    - name: INSTALAR - Paquetes requeridos
      dnf:
        name:
          - yum-utils
          - device-mapper-persistent-data
          - lvm2
        state: latest
        update_cache: true

    - name: AÑADIR - Clave GPG Docker
      rpm_key:
        key: https://download.docker.com/linux/centos/gpg
        state: present

    - name: AÑADIR - Repositorio Docker
      get_url:
        url: https://download.docker.com/linux/centos/docker-ce.repo
        dest: /etc/yum.repos.d/docker-ce.repo
        mode: 0644

    - name: INSTALAR - Docker CE
      dnf:
        name: docker-ce
        state: present

    - name: VERIFICAR - Servicio Docker Iniciado
      systemd:
        name: docker
        enabled: yes
```

Para el correcto funcionamiento del proyecto se deberá de crear el contenedor del balanceador de carga HAProxy, se empezará creando con Ansible el contenedor y se importará su configuración en cada uno de los hosts que haya en el inventario de Ansible.

Se ha creado un archivo de configuración dentro del controlador de Ansible para su posterior configuración en los hosts, este archivo contiene lo siguiente:

`/home/ansible/.ansible/archivos/haproxy.cfg`

```
global
    maxconn 1000

defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend http-in
    bind *:80
    default_backend servers

backend servers
    mode http
    balance roundrobin
    option httpchk HEAD / HTTP/1.1\r\nHost:localhost
    server nginx1 nginx:80 check
    server nginx2 nginx:80 check
    server nginx3 nginx:80 check
    server mariadb1 mariadb:3306 check
    server mariadb2 mariadb:3306 check
    server mariadb3 mariadb:3306 check
```

Las últimas líneas es donde irán los contenedores en los que se necesita que se balancee la carga para el puerto 80, por lo tanto, serán los contenedores de httpd (Nginx). Una vez creado el archivo de configuración se creará la playbook en la que estarán los contenedores de httpd, haproxy y mariadb galera.

Para la configuración de los contenedores y el inicio de los mismos se ha creado otra Ansible Playbook, que contiene la siguiente estructura:

/home/ansible/.ansible/playbooks/docker-conts.yml

```
---
- hosts: maquinas_virtuales
  vars:
  # Nodo inicial + numNodos (MariaDB)
    numNodos: 3
  become: true
  tasks:

    - name: COMPROBACION - Eliminar contenedores activos, huerfanos.
      shell: docker stop $(docker ps -aq) && docker rm $(docker ps -aq) && docker-compose down --remove-orphans || true

    - name: COMPROBACION - Eliminar redes antiguas.
      shell:  docker network prune -f  || true

    - name: COMPROBACION - Eliminar volumenes antiguos.
      docker_volume:
        name: mariadb
        state: absent

    - name: COMPROBACION - Salir de Docker Swarm
      shell: docker swarm leave --force || true

    - name: CREACION - haproxy.cfg  con Jinja2
      template:
        src: /home/ansible/.ansible/templates/jinjaHaproxy.j2
        dest: /home/ansible/haproxy.cfg
        mode: '0644'

    - name: CREACION - docker-compose.yml con Jinja2
      template:
        src: /home/ansible/.ansible/templates/jinjaCompose.j2
        dest: /home/ansible/docker-compose.yml
        mode: '0644'

    - name: INICIAR - Contenedores
      docker_compose:
        project_src: /home/ansible/
        files: docker-compose.yml
```

En la Ansible Playbook que se ha nombrado y mostrado previamente se puede observar las siguientes reglas u ordenes:

“`numNodos: 3`”

Esta variable indica el número de nodos que se crearán para el Clúster de MariaDB.

“`COMPROBACION - Eliminar contenedores activos, huerfanos`”

En esta parte se utilizan comandos como “`docker stop`” para parar los contenedores y “`docker rm`” para eliminarlos, ya que por defecto si se intenta crear contenedores y ya han sido creados, simplemente se iniciarán y no surgirán efecto los cambios que se han hecho o puede provocar algún tipo de fallo si la configuración actual no coincide con la ya existente.

“`COMPROBACION - Eliminar redes antiguas`”

Esta regla se encargará de eliminar todas las interfaces de red creadas por docker creadas con anterioridad, para que no haya fallos a la hora de asignar una nueva y que coincida el nombre con una creada anteriormente.

“`COMPROBACION - Eliminar volumenes antiguos`”

Servirá para eliminar los volúmenes creados anteriormente, utiliza una regla del módulo de docker_compose de Ansible para cumplir con el borrado de volúmenes.

“`COMPROBACION - Salir de Docker Swarm`”

Esta orden utiliza el módulo de shell, elimina la creación de un Swarm creado previamente.

“`CREACION - haproxy.cfg  con Jinja2`”

Esta orden creará el archivo haproxy.cfg con al configuración deseada, se hablará más adelante de como se ha producido su creación.

“`CREACION - docker-compose.yml con Jinja2`”

La orden es similar a la anterior, pero para la creación del archivo docker-compose.yml, del cual se explicará su funcionamiento posteriormente.

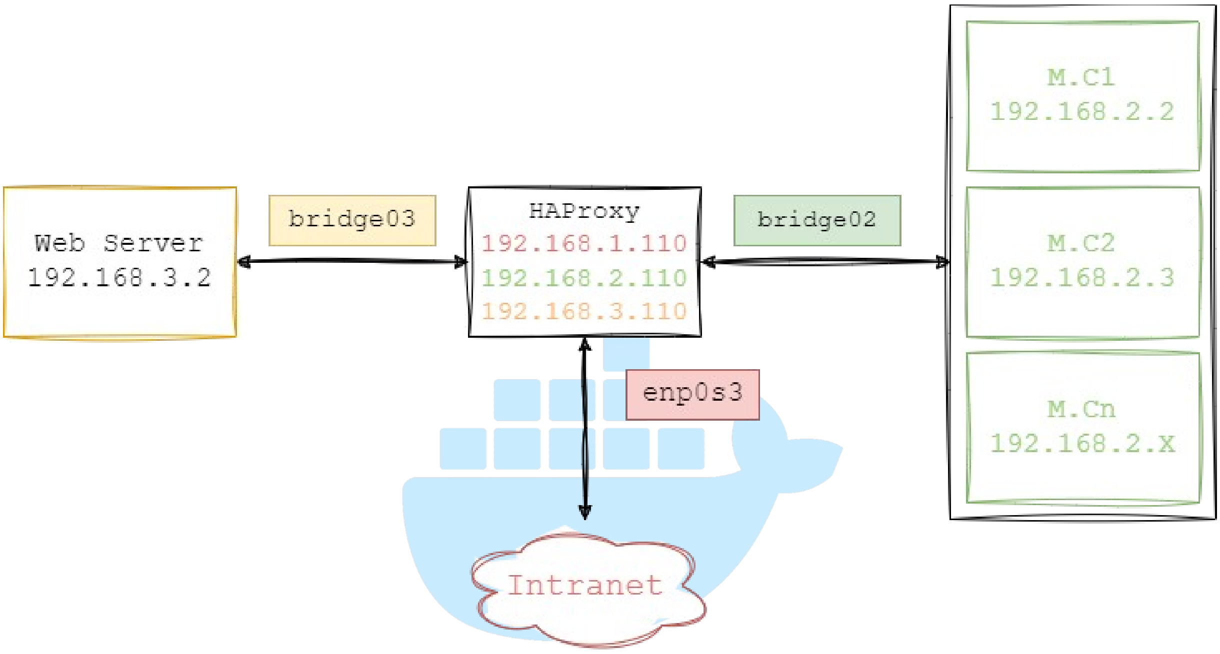
“`INICIAR - Contenedores`”

En este paso se ejecuta el archivo de docker-compose.yml que se ha copiado previamente.

Networking

Para una mayor organización se separará cada grupo de contenedores o de servicios en interfaces virtuales, ya que utilizar las interfaces de Docker puede conllevar a problemas, lo más adecuado es utilizar interfaces propias de la máquina, en este caso, virtual.

La estructura de red será la siguiente:

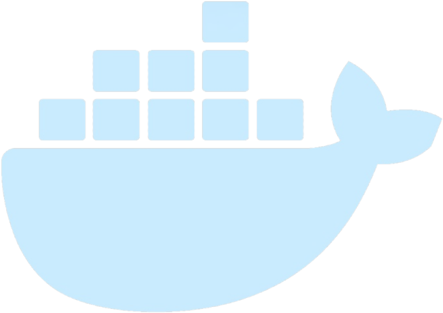


“Enp0s3” (192.168.1.1/24) es la interfaz de Ethernet, que simula la interfaz física de Wi-Fi utilizada por el equipo se utilizará para recibir o enviar las peticiones de / para Intranet. Las demás interfaces (bridge02 y bridge03) dependen de enp0s3 para su funcionamiento, su deber es dividir y organizar la información y las conexiones.

“Bridge02” (192.168.2.1/24) servirá para establecer la conexión entre los nodos de MariaDB Galera Cluster y para que HAProxy balancee la carga entre ellos correctamente.

“Bridge03” (192.168.3.1/24) tendrá el servidor web (Apache, Nginx, etc.) que mandará a través del HAProxy conexiones a las bases de datos, se ha agregado una interfaz aparte para que, en caso de querer formar un clúster de servidores web, sea más fácil.

Para crear las interfaces virtuales se usará NetworkManager, el cual viene por defecto en RockyLinux 9 y en la mayoría de las distribuciones de Linux.



MariaDB Galera

Para el almacenamiento de datos y funcionamiento de un servicio web hará falta los contenedores de MariaDB, que utilizarán una tecnología de clúster como es la herramienta Galera, proporcionada para otorgar redundancia y escalado horizontal a los hosts utilizados como base de datos.

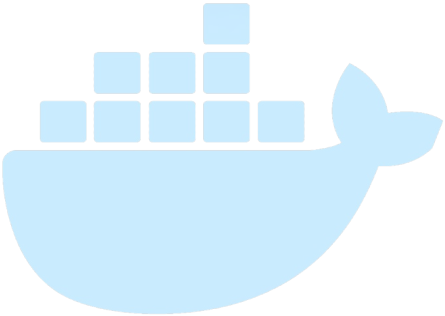
En este caso se utilizarán contenedores, por lo tanto, se hará uso de Ansible Playbooks y herramientas como Docker y Docker-Compose.

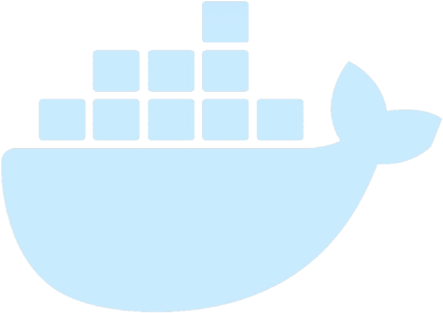
Para que MariaDB Galera Cluster funcione correctamente habrá que usar un balanceador de carga para que las peticiones naveguen a los contenedores que se quieren añadir, y no solo a uno, ya que eso seguiría sin resolver el problema del rendimiento y podría provocar un cuello de botella. Se usará HAProxy y se explicará posteriormente.

Se utilizará como imagen hauptmedia/mariadb:10.1, donde 10.1 es la versión específica de la imagen que se usará. Esta imagen a diferencia de una imagen de MariaDB común, o de una instalación limpia del sistema con MariaDB instalado manualmente, es que trae diferentes variables que se pueden utilizar para crear / iniciar los contenedores, tanto utilizando el comando `docker run`, como utilizando un archivo de Docker-Compose.

Para este proyecto se utilizará Docker-Compose, ya que, al sumarle Ansible, facilita la automatización de los servicios que se desean prestar.

Se crea el archivo docker-compose.yml dentro de “/home/ansible/docker/” para que funcione la orden que se ha especificado en la Ansible Playbook (**“`CREACION - docker-compose.yml con Jinja2`”**)





WEBGRAFÍA

Docker | Imágen mariadb
https://hub.docker.com/_/mariadb

Docker | Imágen httpd
https://hub.docker.com/_/httpd

Docker | Imágen mariadb-galera
https://hub.docker.com/r/bitnami/mariadb-galera#!

Ansible | Community.Docker
https://docs.ansible.com/ansible/latest/collections/community/docker/docker_image_module.html#parameters

Red Hat | How to get started with the Vi editor
https://www.redhat.com/sysadmin/get-started-vi-editor

RockyLinux | Learning Ansible
https://docs.rockylinux.org/books/learning_ansible/01-basic/

GitHub | Hauptmedia Docker-MariaDB
https://github.com/hauptmedia/docker-mariadb

Docker | Hauptmedia MariaDB
https://hub.docker.com/r/hauptmedia/mariadb