Universidad Nacional Autónoma de México
Instituto de Investigaciones en Matemáticas
Aplicadas y en Sistemas (IIMAS)
Licenciatura en Ciencia de Datos

BASES DE DATOS NO ESTRUCTURADAS
EJERCICIO PRÁCTICO 03

1. EXPLORANDO RIAK EN CLUSTER

1.1. Objetivo

El presente ejercicio práctico le permitirá al alumno comprender y poner en práctica el proceso de creación, configuración y uso de un cluster de 3 nodos con Riak. El uso de este cluster permitirá explorar y comprender las principales funcionalidades que ofrece Riak en un ambiente clusterizado como son alta disponibilidad, tolerancia a la partición y replicación.

1.2. Instalar y configurar Docker

El primer paso es realizar la instalación de Docker en caso de no contar con una instalación existente. Esta herramienta permitirá la creación de contenedores en una misma máquina a partir de una imagen. Cada contenedor representará a un nodo del cluster con Riak instalado y configurado listo para interactuar con otros nodos.

1.2.1. Instalar Docker

Para realizar la instalación de Docker, seguir los pasos que se indican en la sección ***Instalación de Docker Engine*** del documento `COMUN/manual-instalacion-docker.pdf`. El documento incluye instrucciones tanto para Ubuntu/Mint como para CentOS/Fedora/RHEL. Se recomienda leer la introducción del documento en el que se explica las características de Docker.

1.2.2. Creación de una red para comunicar contenedores

Al hacer uso de contenedores como Docker, es posible crear redes personalizadas definidas por el usuario, y conectar múltiples contenedores a la misma red. Una vez conectados a una red definida por el usuario, los contenedores pueden comunicarse entre sí utilizando las direcciones IP de los contenedores o sus nombres. Para llevar a cabo este proceso, se puede emplear la siguiente instrucción:

```
sudo docker network create -d bridge \
--subnet=<subred> --gateway=<gateway> <id-red>
```

- La bandera `-d` indica el driver que utilizaremos. Una red bridge es un dispositivo de capa de enlace que reenvía el tráfico entre segmentos de red.
- Una red bridge en docker utiliza un puente de software que permite que los contenedores conectados a la misma red se comuniquen, al mismo tiempo que proporciona aislamiento de los contenedores que no están conectados a esa red de puente. El controlador bridge de Docker instala automáticamente reglas en la máquina anfitriona para que los contenedores en diferentes redes bridge no puedan comunicarse directamente entre sí.
- Se hará uso de la subred `172.18.0.0/16` debido a que por defecto, docker hace uso de la subred `172.17.0.0/16`.

  - Sustituir `<subnet>` con el valor `172.18.0.0/16`.
  - Sustituir `<gateway>` con el valor `172.18.0.1`
  - Sustituir `<id-red>` con el valor `<iniciales>-net-riak`. Iniciales está formada por 3 caracteres: la primera letra del primer nombre, primer letra de cada apellido
- Ejecutar el comando, mostrar la red creada. Se deberá obtener una salida similar al siguiente ejemplo

```
jorge@lap-red-mint:~$ sudo docker network create -d bridge \
> --subnet=172.18.0.0/16 --gateway=172.18.0.1 jrc-net-riak
[sudo] password for jorge:
cbea71165bf4400305da38b6cbfaca97e7e86b0e58fce47b8323ed07af6662bd

jorge@lap-red-mint:~$ docker network ls --filter driver=bridge
NETWORK ID     NAME           DRIVER    SCOPE
bbfdd7d24b40   bridge         bridge    local
cbea71165bf4   jrc-net-riak   bridge    local
jorge@lap-red-mint:~$
```

1.3. Creación de nodos

1.3.1. Creación y configuración del contenedor coordinador

Se sugiere leer la sección creación del contenedor en el documento `COMUN/manual-instalacion-docker.pdf` para familiarizarse con el proceso general para crear un contenedor Docker así como algunos conceptos básicos. En esta sección se explica el concepto de ***volumen*** empleado para compartir carpetas entre la máquina host y el contenedor. No se requiere crear ningún directorio ya que esto se realizó en ejercicios prácticos anteriores. Leer hasta el punto donde se crea el contenedor con la instrucción `docker run`. En lugar de utilizar la instrucción indicada en el documento, se deberá hacer uso de la siguiente:

```
sudo docker run -i -t \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /unam-bd-ne:/unam-bd-ne \
--network=<iniciales>-net-riak \
--ip=172.18.0.10 \
--name c-<iniciales>-riak-coord \
--hostname d-riak-coord-<iniciales>.iimas.unam \
--expose 8087 \
--expose 8098 \
--expose 8099 \
--expose 9080 \
-e DISPLAY=$DISPLAY \
<imagen>:<tag> \
bash
```

Explicación

- `docker run`: Crea un nuevo contenedor
- `-i` Mantiene abierto el STDIN (entrada estándar de comandos) del contenedor y permite enviar comandos al contenedor a través de la entrada estándar(terminal)
- `-t` Adjunta un pseudo-TTY al contenedor, lo que conecta la terminal a los flujos
- de entrada/salida del contenedor.
- La opción `-v` permite configurar los volúmenes descritos anteriormente. Se pueden agregar tantos volúmenes como sean necesarios. Se emplea la sintaxis `<src_dir>:<dest_dir>`
    - `src_dir` hace referencia al directorio origen ubicado en la máquina host
    - `dest_dir` indica el directorio que será creado en el contenedor. Como se mencionó anteriormente se recomienda que ambas rutas sean idénticas por simplicidad.
- `--network`: Conecta una red al contenedor. Sustituir las iniciales. Observar que el nombre de esta red corresponde con el nombre de la red bridge creada anteriormente.
- `--ip` Asigna una dirección IP al contenedor.
- `--name` Asigna un nombre al contenedor. Sustituir iniciales
- `--hostname` Asigna un hostname al contenedor. Sustituir iniciales.
- `--expose` Expone un puerto o un rango de puertos. El puerto podrá ser alcanzado únicamente por la máquina host y otros contenedores en la máquina host. Para el exterior se utiliza `--publish`.
- `-e` Establece variables de entorno.
- `bash` Indica la ejecución del intérprete bash en el contenedor.
- `8087` Puerto utilizado para la interfaz protobuf (formato binario para intercambio de datos).
- `8098` Puerto empleado para la interfaz HTTP.

- `8099` Puerto utilizado por Riak para handoff. Es un proceso mediante el cual los datos se transfieren de un nodo a otro dentro de un clúster. Se activa cuando un nodo se une o se desconecta del clúster, o cuando ocurre una redistribución de datos debido a cambios en la topología del clúster.
- `9080` Puerto empleado por el cluster manager para escuchar conexiones de clusters remotos.
- `DISPLAY=$DISPLAY`: Variable que especifica dónde se deben mostrar las
- aplicaciones gráficas. Se le asigna el mismo valor que tiene la máquina host.
- `<imagen>:<tag>` Para efectos de este ejercicio, se creará un contenedor con ubuntu. Y dentro de este contenedor se instalará Riak. Sustituir con los siguientes valores
    - **`ubuntu:jammy`**

Ejemplo

```
sudo docker run -i -t \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /unam-bd-ne:/unam-bd-ne \
--network=jrc-net-riak \
--ip=172.18.0.10 \
--name c-jrc-riak-coord \
--hostname d-riak-coord-jrc.iimas.unam \
--expose 8087 \
--expose 8098 \
--expose 8099 \
--expose 9080 \
-e DISPLAY=$DISPLAY \
ubuntu:jammy \
bash
```

Al ejecutar el comando anterior, se accede al contenedor ubuntu como usuario `root`. Ejecutar las instrucciones que se muestran en el siguiente ejemplo para verificar la salida esperada

```
root@d-riak-coord-jrc:/#
root@d-riak-coord-jrc:/# more /etc/os-release
```

Salida esperada:

```
PRETTY_NAME="Ubuntu 22.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.1 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
```

```
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-poli
cy"
UBUNTU_CODENAME=jammy
root@d-riak-coord-jrc:/#
```

1.3.1.1. Instalar paquetes básicos

Actualizar el sistema operativo e instalar los siguientes paquetes iniciales antes de comenzar con la instalación de Riak. Continuar con el usuario `root`.

Actualizar el caché de repositorios

```
apt update && apt upgrade
```

Instalación de dependencias base

```
apt-get install sudo
apt-get install nano
apt-get install unzip
apt-get install iputils-ping
apt-get install locales
apt-get install lshw
apt-get install passwd
apt-get install wget
apt-get install iproute2
```

1.3.1.2. Instalar Riak en el contenedor

El proceso de instalación de Riak en el contenedor es muy similar al realizado en ejercicios prácticos anteriores. Ejecutar las siguientes instrucciones en el nuevo contenedor empleando al usuario root.

Instalación de dependencias

```
apt-get install libc6 libc6-dev libc6-dbg cron libpopt0 logrotate
```

Descargar el Zip de Riak

```
cd /tmp
wget https://files.tiot.jp/riak/kv/3.2/3.2.0/ubuntu/jammy64/riak_3.2.0-OTP25_amd64.deb
```

Instalar Riak

```
sudo dpkg -i riak_3.2.0-OTP25_amd64.deb
```

Agregar parámetros al kernel para Riak

Configurar los siguientes parámetros del Kernel para aumentar el número de archivos abiertos que pueden existir en el sistema. Agregar las siguientes líneas en `/etc/security/limits.conf`

```
riak soft nofile 65536
riak hard nofile 200000
```

Iniciar y probar Riak

```
riak daemon
riak ping
```

1.3.1.3. Comandos docker run, docker start, attach

¿Qué sucede si cerramos la terminal o si salimos del contenedor ?

- Básicamente el contenedor se cierra y su contenido queda persistido en el directorio de contenedores de Docker.
- Para acceder nuevamente al contenedor creado, se emplean los comandos `docker start`, `docker attach`. Se sugiere leer la sección ***Comandos docker run, docker start, attach*** del documento `COMUN/manual-instalacion-docker.pdf` en el que se explica el uso de estos dos comandos.

1.3.1.4. Configuración inicial del nodo coordinador

Una vez creado el contenedor coordinador, será necesario ajustar algunos parámetros para asegurar el correcto uso de IPs y puertos. Para ello, realizar las siguientes configuraciones

A. Abrir el archivo de configuración de Riak `/etc/riak/riak.conf`

```
sudo nano /etc/riak/riak.conf
```

B. Modificar o agregar las siguientes configuraciones en caso de no existir.

- Dirección IP y puerto para el cluster. Riak emplea un protocolo llamado Protocol Buffers interface (`protobuf`) que ofrece un mayor desempeño respecto a trabajar con el protocolo `http`. Este parámetro ya existe en el archivo por lo que su valor se deberá actualizar por la siguiente línea:

```
listener.protobuf.internal = 172.18.0.10:8087
```

- Actualizar la IP y puerto para hacer uso del protocolo protobuf para la interfaz http de Riak. De forma similar este parámetro ya existe por lo que se deberá actualizar su valor como se muestra a continuación:

```
listener.http.internal = 172.18.0.10:8098
```

- Nombre del nodo.
  - Sustituir `<iniciales>` según corresponda
  - Sustituir `<hostname>` con el nombre del host del contenedor. Este valor fue configurado al crear el contenedor con el comando docker run empleando el parámetro `--hostname`, por ejemplo: `--hostname d-riak-coord-jrc.iimas.unam`

```
nodename = riak0@<hostname>
```

- Puerto para handoff. Agregar la siguiente instrucción al final del archivo:

```
handoff.port = 8099
```

C. Abrir el archivo `/etc/riak/advanced.config`, actualizar la siguiente configuración con el valor indicado:

```
{cluster_mgr, {"172.18.0.10", 9080 } }
```

Iniciar Riak en el nodo

- En caso de haber iniciado Riak en el nodo anteriormente, se deberán eliminar los siguientes archivos para que el nuevo nombre del nodo coordinador configurado anteriormente sea actualizado

```
cd /var/lib/riak/ring
rm *
```

- Ejecutar la siguiente instrucción para iniciar el nodo y comprobar la configuración

```
riak daemon
raak ping
```

1.3.2. Creación de nodos adicionales

Una vez se cuenta con un nodo configurado, es posible crear una imagen a partir de dicho nodo que nos permita generar nuevos contenedores. Para crear una imagen del contenedor creado anteriormente, salir del contenedor y ejecutar la siguiente instrucción desde la máquina host.

```
sudo docker commit <id-contenedor> <id-imagen>:<tag>
```

- `<id-contenedor>` corresponde al nombre del contenedor asignado anteriormente, es decir: `c-<iniciales>-riak-coord`
- `<id-imagen>` asignar el valor con el valor `<iniciales>-riak-coord,` es decir, quitar `'c-'`.
- `<tag>` asignar el valor 1.0

Ejemplo

```
sudo docker commit c-jrc-riak-coord jrc-riak-coord:1.0
```

Para mostrar las imágenes que existen hasta el momento se puede emplear el siguiente comando:

```
sudo docker images -a
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
jrc-riak-coord          1.0       a87acd319d47   4 days ago      839MB
```

En caso de requerir eliminar una imagen, se puede emplear el siguiente comando haciendo uso del ID de la imagen mostrada en el ejemplo anterior

```
 docker rmi <image id>
```

El siguiente paso es la creación de nuevos contenedores a partir de la imagen generada. Se emplea nuevamente el comando `docker run`, pero con los siguientes cambios:

```
sudo docker run -i -t \
-v /tmp/.X11-unix:/tmp/.X11-unix \
```

```
-v /unam-bd-ne:/unam-bd-ne \
--network=<iniciales>-net-riak \
--ip=172.18.0.11 \
--name c-<iniciales>-riak-node1 \
--hostname d-riak-node1-<iniciales>.iimas.unam \
--expose 8187 \
--expose 8198 \
--expose 8199 \
--expose 9180 \
-e DISPLAY=$DISPLAY \
<iniciales>-riak-coord:1.0 \
bash
```

Cambios realizados:

- ip del contenedor: `172.18.0.10` → `172.18.0.11`
- Nombre del contenedor: `c-<iniciales>-riak-coord → c-<iniciales>-riak-node1`
- Puertos: se le suman 100, por ejemplo `8087` → `8187`
- Observar el valor de la imagen y su tag: `<iniciales>-riak-coord:1.0` corresponde con el nombre de la imagen y el tag creado anteriormente.

Posterior al a la creación del contenedor, se deberán realizar las mismas configuraciones que se realizaron en el nodo coordinador, es decir:

A. Ajustar los parámetros del archivo `/etc/riak/riak.conf`
   - `listener.protobuf.internal`
   - `listener.http.internal`
   - `nodename`
   - `handoff.port`
B. Ajustar los parámetros del archivo `advanced.config`
   - `cluster_mgr`
C. Eliminar los archivos del anillo
D. Iniciar Riak en el nodo

Realizar el proceso para crear 3 nodos más.

En general, se deben actualizar IPs, nombres de hosts, y puertos de tal forma que cada nodo tenga valores únicos. La siguiente tabla muestra un ejemplo de los valores asignados a los 4 nodos. Para el caso de los puertos, asegurar que los valores configurados con la instrucción docker run correspondan con los puertos actualizados en los archivos de configuración.

comando docker run				riak.conf				advanced.config
ip contenedor	nombre contenedor	hostname	puertos --expose	listener .protobuf. internal	listener .http. internal	nodename	handoff. port	cluster_mgr
172.18.0.1**0**	c-jrc-riak-**coord**	d-riak-**coord**-jrc. iimas.unam	80**8**7 80**9**8 80**9**9 9**0**80	172.18.0.1**0**: 80**8**7	172.18.0.1**0**: 80**9**8	riak**0**@ d-riak-coord-jrc. iimas.unam	80**9**9	{"172.18.0.1**0**", 90**8**0 }
172.18.0.1**1**	c-jrc-riak-**node1**	d-riak-**node1**-jrc. iimas.unam	81**8**7 81**9**8 81**9**9 91**8**0	172.18.0.1**1**: 81**8**7	172.18.0.1**1**:81**9**8	riak**1**@ d-riak-**node1**-jrc. iimas.unam	81**9**9	{"172.18.0.1**1**", 91**8**0 }
172.18.0.1**2**	c-jrc-riak-**node2**	d-riak-**node2**-jrc. iimas.unam	82**8**7 82**9**8 82**9**9 92**8**0	172.18.0.1**2**: 82**8**7	172.18.0.1**2**: 82**9**8	riak**2**@ d-riak-**node2**-jrc. iimas.unam	82**9**9	{"172.18.0.1**2**", 92**8**0 }
172.18.0.1**3**	c-jrc-riak-**node3**	d-riak-**node3**-jrc. iimas.unam	83**8**7 83**9**8 83**9**9 93**8**0	172.18.0.1**3**: 83**8**7	172.18.0.1**3**: 83**9**8	riak**3**@ d-riak-**node3**-jrc. iimas.unam	83**9**9	{"172.18.0.1**3**", 93**8**0 }

En caso de requerir eliminar un contenedor, se puede emplear la siguiente instrucción en la máquina host:

```
sudo docker container rm <nombre_o_id_contenedor>
```

Para ver la lista de contenedores existentes en la que se puede apreciar el id y su nombre se emplea el siguiente comando:

```
sudo docker ps -a
#Salida obtenida:
CONTAINER ID   IMAGE          COMMAND   CREATED      STATUS          PORTS                                NAMES
fa823623a040   ubuntu:jammy   "bash"    8 days ago   Up 26 minutes   8087/tcp, 8098-8099/tcp, 9080/tcp    c-jrc-riak-coord
```

1.3.3. Comprobar comunicación entre nodos

A. Abrir 4 terminales, acceder a cada uno de los contenedores
B. ejecutar el comando ping en cada terminal indicando las diferentes direcciones IP de los 3 nodos restantes para comprobar disponibilidad. Se deberá obtener una salida similar a la siguiente

```
root@d-riak-coord-jrc:/# ping 172.18.0.12
```

```
PING 172.18.0.12 (172.18.0.12) 56(84) bytes of data.
64 bytes from 172.18.0.12: icmp_seq=1 ttl=64 time=0.150 ms
```

1.4. Creación de un cluster Riak

Después de realizar las configuraciones para los nodos que se emplearán en el cluster y verificar la conexiÓn entre ellos, se podr´a crear un cluster al iniciar riak en cada nodo y posteriormente agregarlos a un mismo cluster.
Para llevar a cabo este proceso, se utilizan las siguientes instrucciones.

A. En cada nodo iniciar Riak

```
riak daemon
```

B. Acceder a cada uno de los nodos que NO son coordinadores y ejecutar la siguiente instrucción que permite asociar al nodo con el nodo coordinador.

```
riak admin cluster join riak0@<hostname_root>
#ejemplo:
riak admin cluster join riak0@d-riak-coord-jrc.iimas.unam
```

1.4.1. Confirmar cambios para crear un cluster

A. Ejecutar las siguientes instrucciones en cualquier nodo para revisar la configuración y finalmente para confirmar la configuración del nuevo cluster

```
riak admin cluster plan
riak admin cluster commit
```

B. Finalmente, ejecutar la siguiente instrucción para revisar la configuración del cluster

```
riak admin cluster status
```

Se obtendrá una salida similar al siguiente ejemplo. Notar la letra (C) que indica al nodo coordinador.

```
---- Cluster Status ----
Ring ready: false

+-------------------------------------------+-------+-------+-----+-------+
|                    node                   |status | avail |ring |pending|
+-------------------------------------------+-------+-------+-----+-------+
|   (C) riak0@d-riak-coord-jrc.iimas.unam   | valid |  up   |100.0|  25.0 |
|       riak1@d-riak-node1-jrc.iimas.unam   | valid |  up   |  0.0|  25.0 |
|       riak2@d-riak-node2-jrc.iimas.unam   | valid |  up   |  0.0|  25.0 |
|       riak3@d-riak-node3-jrc.iimas.unam   | valid |  up   |  0.0|  25.0 |
+-------------------------------------------+-------+-------+-----+-------+
```