



Integración Continua

Proyecto de software basado en herramientas de integración continua

Andrés Felipe Rojas Parra - Código:100234216

Daniel Ricardo Gómez Riveros - Código 1520010641

Nelson Lainelec Santisteban Toloza - Código: 1721026128

Yohanny Murillo Urrutia - Código: 1721025369

Cesar Augusto Salamanca Panqueva - Código: 1411024392

Tutor:

Natalia Martinez Rojas

Politécnico Gran Colombiano

Ingeniería Diseño e Innovación

Integración continua

2021



Entrega I Semana 3

1. Creación de Repositorio en Github

De acuerdo a lo planteado por la docente, adjuntamos el link correspondiente a nuestro repositorio y su respecta captura:

<https://github.com/nelaisanty9/Integracion-Continua>

- Creación de repositorio en Github:

The screenshot shows the GitHub repository page for 'nelaisanty9/Integracion-Continua'. The repository is the 'principal' branch with 2 branches and 0 tags. It was updated by 'danigomr' 1 hour ago. The file list includes 'Integrantes.txt' (updated 1 hour ago) and 'README.md' (initial commit 5 days ago). The README content is visible, showing the repository title 'Integracion-Continua' and its purpose: 'Repositorio de integracion continua poli'. The right sidebar contains sections for 'Acercas de' (About), 'Lanzamientos' (Releases), and 'Paquetes' (Packages).

- Actualización del archivo .txt con los nombres de los integrantes del grupo:

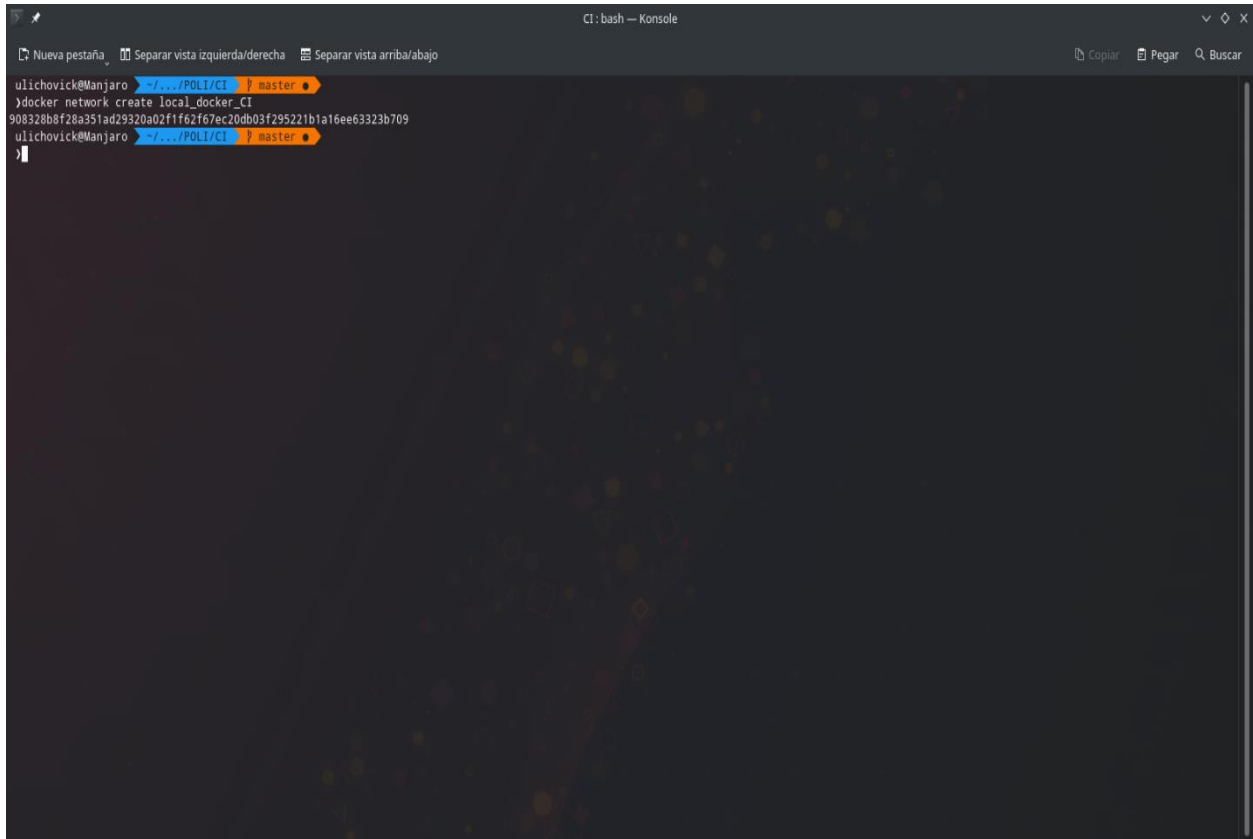
The screenshot shows the content of the 'Integrantes.txt' file. It lists 5 contributors with their names and GitHub usernames in parentheses:

```
1  Andrés Felipe Rojas Parra (usuario ulichovick).
2  Nelson Lainelec Santisteban Toloza (usuario nelaisanty9)
3  Yohanny Murillo Urrutia (usuario YohannyM)
4  Daniel Ricardo Gómez Riveros (usuario danigomr)
5  Cesar Augusto Salamanca Panqueva (usuario cesarsalamanca)
```

2. Creación de los dos Contenedores

Paso 1. crear red local en docker

docker network create local_docker_CI



```
CI: bash — Konsole
Nueva pestaña Separar vista izquierda/derecha Separar vista arriba/abajo Copiar Pegar Buscar
ulichovick@Manjaro > /.../POLI/CI > master
)docker network create local_docker_CI
908328b8f28a351ad29320a02ff62f67ec20db03f295221b1a16ee63323b709
ulichovick@Manjaro > /.../POLI/CI > master
)
```

Paso 1. Crear Red

#Paso 2. Construir imágenes:

crear imagen de python en docker

docker build -t my_docker_python my_docker_python/

crear imagen de mariadb en docker

docker build -t my_docker_mariadb my_docker_mariadb/

```
CI: bash — Konsole
Nueva pestaña Separar vista izquierda/derecha Separar vista arriba/abajo
ulichovick@Manjaro ~/.../POLI/CI master
>docker network create local_docker_CI
908328b8f28a351ad29320a02f1f62f67ec20db03f295221b1a16ee63323b709
ulichovick@Manjaro ~/.../POLI/CI master
```

Paso 2. Construir Imágenes

#Paso 3. Verificar imágenes:

```
CI: bash — Konsole
Nueva pestaña Separar vista izquierda/derecha Separar vista arriba/abajo
ulichovick@Manjaro ~/.../POLI/CI master
>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
my_docker_python latest 8e08c0eab36c 5 seconds ago 937MB
my_docker_mariadb latest fcf428d896d7 21 seconds ago 498MB
mariadb latest eff629089685 7 days ago 408MB
python 3 5b3b4504ff1f 8 days ago 886MB
ulichovick@Manjaro ~/.../POLI/CI master

CI: bash — Konsole
Nueva pestaña Separar vista izquierda/derecha Separar vista arriba/abajo
ulichovick@Manjaro ~/.../POLI/CI master
>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
my_docker_python latest 8e08c0eab36c 11 seconds ago 937MB
my_docker_mariadb latest fcf428d896d7 27 seconds ago 498MB
mariadb latest eff629089685 7 days ago 408MB
python 3 5b3b4504ff1f 8 days ago 886MB
ulichovick@Manjaro ~/.../POLI/CI master
```

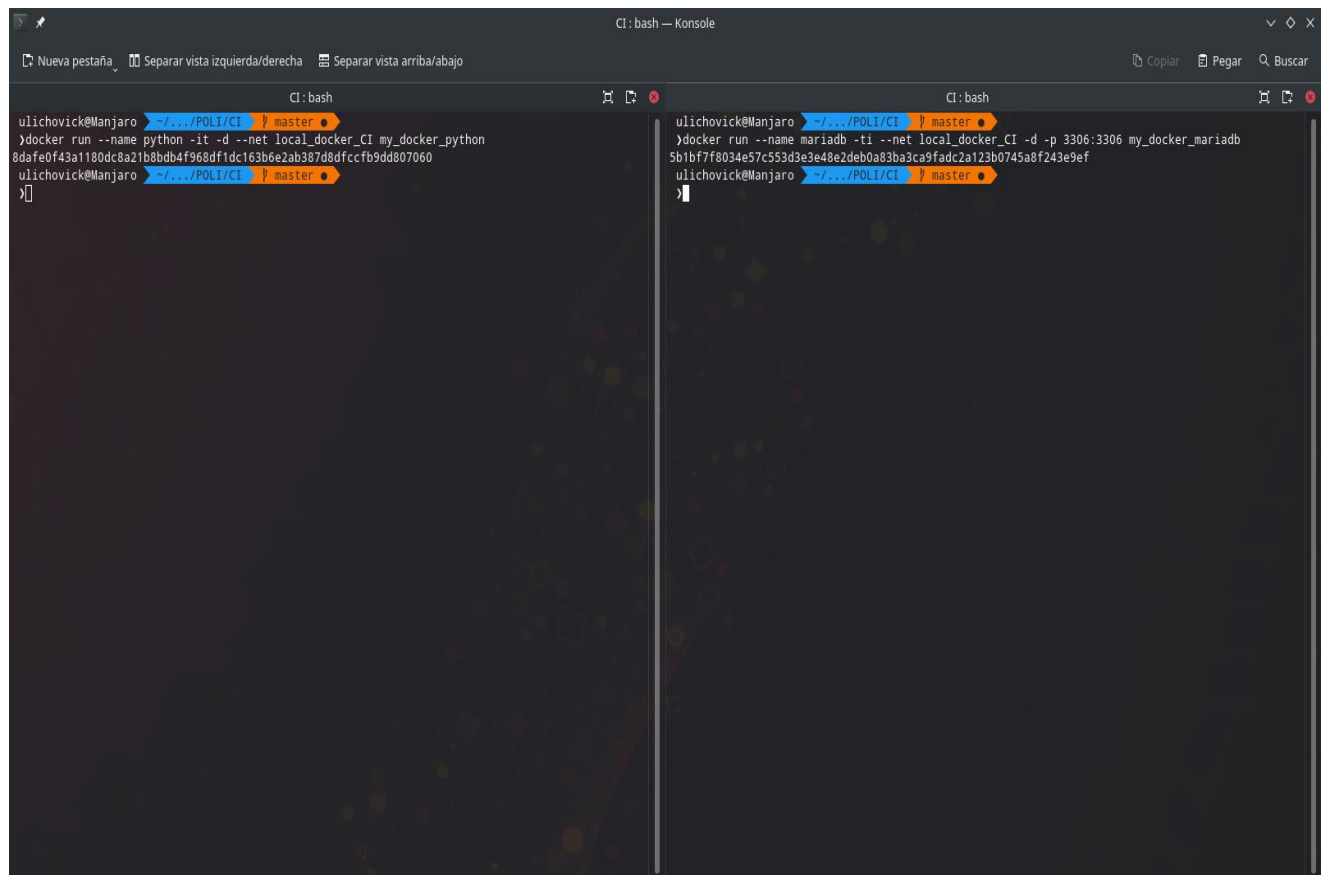
Paso 3. Verificar Imágenes

#Paso 4. Construir Contenedores:

crear contenedor de docker en modo interactivo, que corra en segundo plano y que esté anclado a la red previamente creada

```
docker run --name python -it -d --net local_docker_CI my_docker_python
```

```
docker run --name mariadb -ti --net local_docker_CI -d -p 3306:3306  
my_docker_mariadb
```



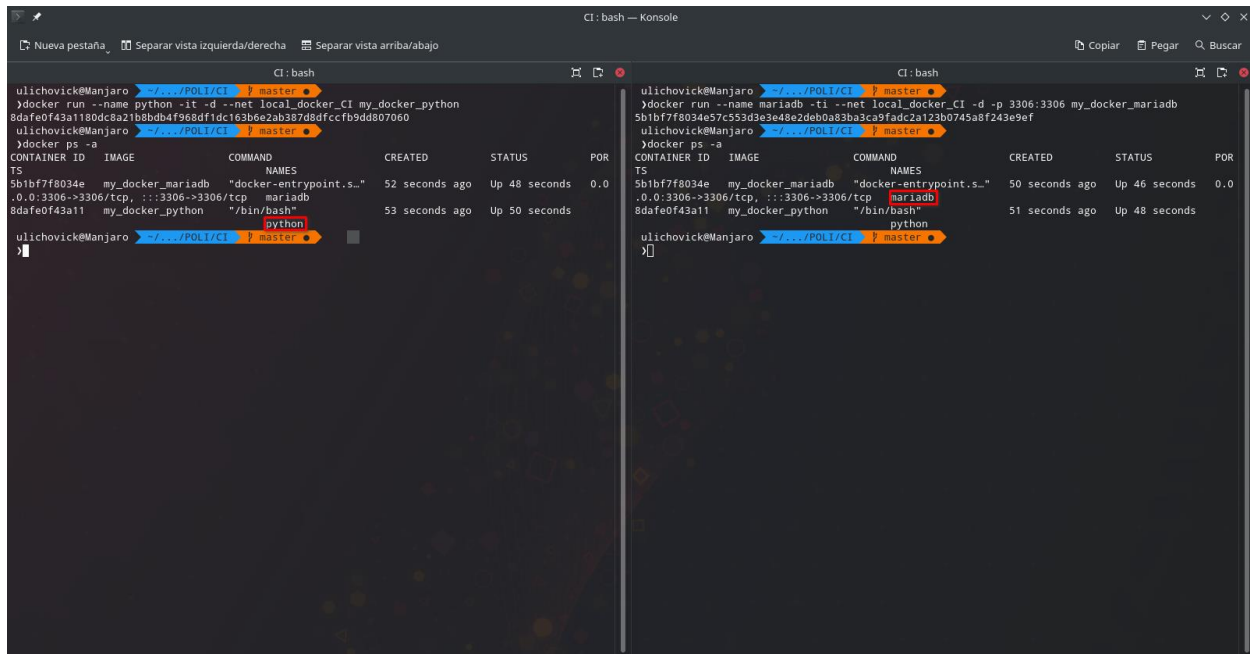
```
CI: bash — Konsole
Nueva pestaña  Separar vista izquierda/derecha  Separar vista arriba/abajo  Copiar  Pegar  Buscar

ulichovick@Manjaro ~/.../POLI/CI master
> docker run --name python -it -d --net local_docker_CI my_docker_python
8d4fe0f43a1180dc8a21b8bdb4f968df1dc163b6e2ab387d8dfccfb9dd807060
ulichovick@Manjaro ~/.../POLI/CI master
>

ulichovick@Manjaro ~/.../POLI/CI master
> docker run --name mariadb -ti --net local_docker_CI -d -p 3306:3306 my_docker_mariadb
5b1bf7f8034e57c553d3e3e48e2deb0a83ba3ca9fad2a123b0745a8f243e9ef
ulichovick@Manjaro ~/.../POLI/CI master
>
```

Paso 4. Construir Contenedores

#Paso 5. Verificar Contenedores:

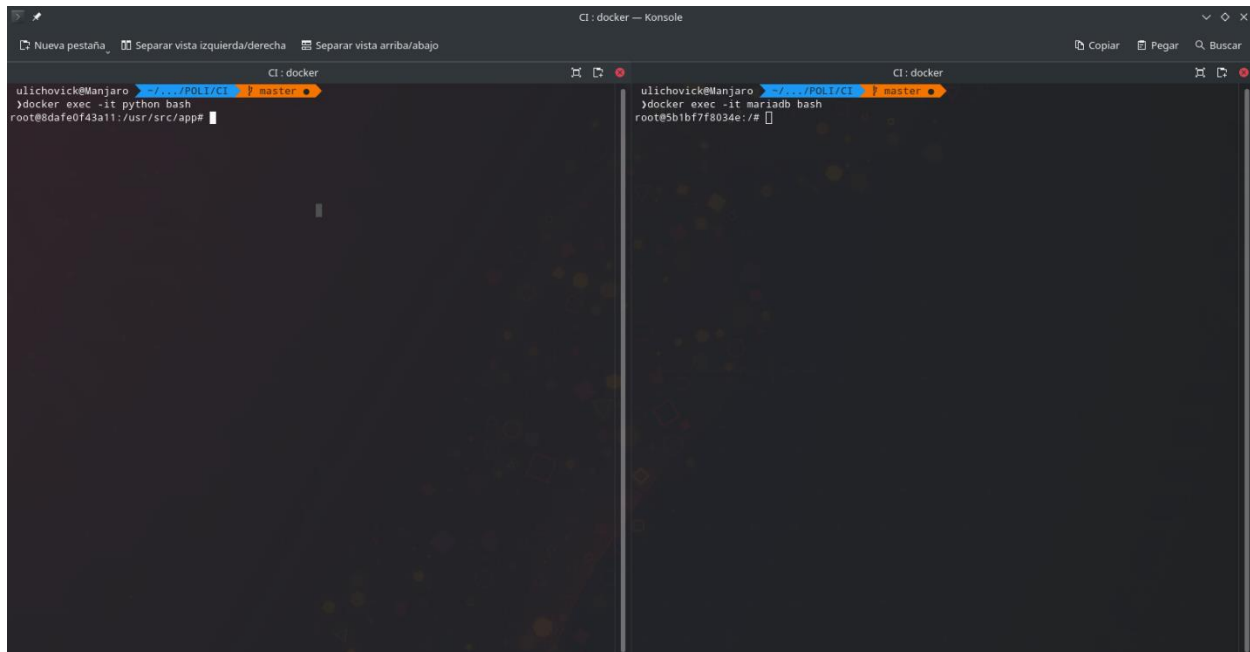


```
CI: bash
ulichovick@Manjaro ~$ docker run --name python -it -d --net local_docker_CI my_docker_python
8d4fe0f43a1180dc8a21b8b0b4f968df1dc163b6e2ab387d8dfccfb9dd807060
ulichovick@Manjaro ~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
5b1bf7f8034e   my_docker_mariadb  "docker-entrypoint.s..." 52 seconds ago Up 48 seconds 0.0
8d4fe0f43a11   my_docker_python  "/bin/bash"              53 seconds ago Up 50 seconds
ulichovick@Manjaro ~$ docker run --name mariadb -ti --net local_docker_CI -d -p 3306:3306 my_docker_mariadb
5b1bf7f8034e57c553d3e3e48e2deb0a83ba3ca9fad2a123b0745a8f243e9ef
ulichovick@Manjaro ~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
5b1bf7f8034e   my_docker_mariadb  "docker-entrypoint.s..." 50 seconds ago Up 46 seconds 0.0
8d4fe0f43a11   my_docker_python  "/bin/bash"              51 seconds ago Up 48 seconds
ulichovick@Manjaro ~$
```

Paso 5. Verificar Contenedores

#Paso 6. correr contenedor de manera interactiva

docker exec -it mariadb bash



```
CI: docker
ulichovick@Manjaro ~$ docker exec -it python bash
root@8d4fe0f43a11:/usr/src/app#
ulichovick@Manjaro ~$ docker exec -it mariadb bash
root@5b1bf7f8034e:/#
```

Paso 6. Correr Contenedores



#Paso 7. Ping entre Contenedores

The screenshot shows two terminal windows side-by-side. The left window is titled 'CI: docker' and shows a user running a python script to ping a mariadb container. The output shows successful pings with varying times. The right window is also titled 'CI: docker' and shows a user running a ping command to a python container. The output shows successful pings with varying times.

```
ulichovick@Manjaro ~$ docker exec -it python bash
root@8d4fe0f43a11:/usr/src/app# ping mariadb
PING mariadb (172.18.0.3) 56(84) bytes of data:
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=1 ttl=64 time=0.050 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=2 ttl=64 time=0.099 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=3 ttl=64 time=0.148 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=4 ttl=64 time=0.155 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=5 ttl=64 time=0.102 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=6 ttl=64 time=0.093 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=7 ttl=64 time=0.098 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=8 ttl=64 time=0.097 ms
64 bytes from mariadb.local_docker_CI (172.18.0.3): icmp_seq=9 ttl=64 time=0.081 ms
^C
^C

ulichovick@Manjaro ~$ docker exec -it mariadb bash
root@5b1bf7f8034e:/# ping python
PING python (172.18.0.2) 56(84) bytes of data:
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=1 ttl=64 time=0.131 ms
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=2 ttl=64 time=0.100 ms
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=3 ttl=64 time=0.102 ms
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=4 ttl=64 time=0.119 ms
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=5 ttl=64 time=0.100 ms
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=6 ttl=64 time=0.096 ms
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=7 ttl=64 time=0.099 ms
64 bytes from python.local_docker_CI (172.18.0.2): icmp_seq=8 ttl=64 time=0.097 ms
^C
^C
```

Paso 7. Ping entre Contenedores

#Paso 8. Datos de mariadb a py

The screenshot shows two terminal windows side-by-side. The left window is titled 'CI: docker' and shows a user running a python script to connect to a mariadb database and retrieve data. The output shows a list of data rows. The right window is also titled 'CI: docker' and shows a user running a mysql command to connect to a mariadb database and retrieve data. The output shows a table with 3 rows.

```
root@8d4fe0f43a11:/usr/src/app# python test.py
id: 1
nombre: Toto
id: 2
nombre: Jack
id: 3
nombre: Titi
root@8d4fe0f43a11:/usr/src/app#

root@5b1bf7f8034e:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 6
Server version: 10.5.10-MariaDB-1:10.5.10+maria-focal.mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use testDB
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [testDB]> select * from test;
+-----+
| id | name |
+-----+
| 1 | Toto |
| 2 | Jack |
| 3 | Titi |
+-----+
3 rows in set (0.000 sec)

MariaDB [testDB]>
```

Paso 8. Datos de mariadb a py

3. Conclusiones

- Trabajamos con la herramienta GitHub y lo primero que reconocemos es la gran comunidad de desarrolladores de todo el mundo que utilizan esta plataforma web gratis, rápida y eficiente para repositorios y proyectos, para utilizarla lo único que necesitamos es instalar Git en nuestro equipo de cómputo, crear tu cuenta en GitHub y luego de realizar algunas configuraciones sencillas y ejecutar unos comandos sencillos podemos enviar documentos o archivos con algún tipo de código y luego invitar a tus colegas o grupo de trabajo para que tengan acceso a esa información y poderla modificar por parte de todos los invitados a través de una clonación, modificación, adición y los famosos commits que quedan grabados en la misma plataforma.
- GitHub es una excelente herramienta para el trabajo en equipo y facilita mucho la creación y desarrollo de proyectos, el equipo de trabajo puede trabajar al mismo tiempo desde diferentes lugares, diferentes ciudades incluso diferentes países del mundo. Además, el no tener copias de seguridad y la garantía de poder volver al estado inicial por si se cometen errores hace que sea muy versátil trabajar y muy seguro.
- También trabajamos con Docker que es una plataforma que empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute. Los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y con comandos sencillos permite crear, iniciar o detener contenedores.
- Utilizamos imágenes de Docker las cuales son plantillas de solo lectura que definen al contenedor, estas imágenes contienen un código que se va a ejecutar y así esta ejecución es la que da como resultado un contenedor.
- Nos dimos cuenta desde el inicio que Docker utiliza ciertos aspectos de Linux por lo que si no teníamos este sistema operativo era mejor trabajar en una máquina virtual Linux. Con respecto a la comunicación de los contenedores se prefirió utilizar una red virtual personalizada y no la propia red de Docker para ahorrarnos los pasos de buscar las IP de los contenedores y en vez de eso conectarlos con su alias.