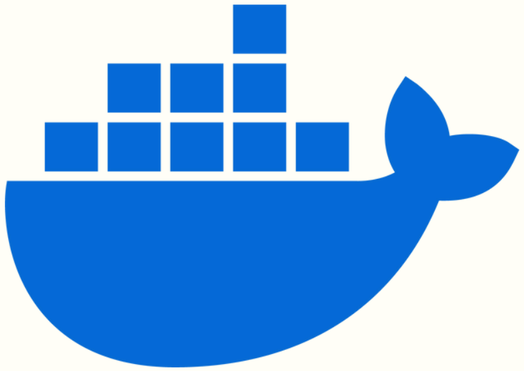
CONTENEDORES DOCKER PARA BALANCEO DE CARGA EN APLICACIONES WEB

Javier Chinchilla Lugo



¿QUE ES DOCKER?

Docker es una herramienta que facilita la creación, implementación y ejecución de aplicaciones en contenedores ligeros, lo que permite aislar las aplicaciones del entorno del sistema operativo.

DOCKER COMPOSE

Docker Compose permite definir aplicaciones con múltiples servicios (contenedores) en un solo archivo. Con docker-compose up se levantan automáticamente todos los servicios.

HAPROXY

HAProxy es una herramienta de balanceo de carga que distribuye el tráfico entre múltiples contenedores, garantizando alta disponibilidad y rendimiento.

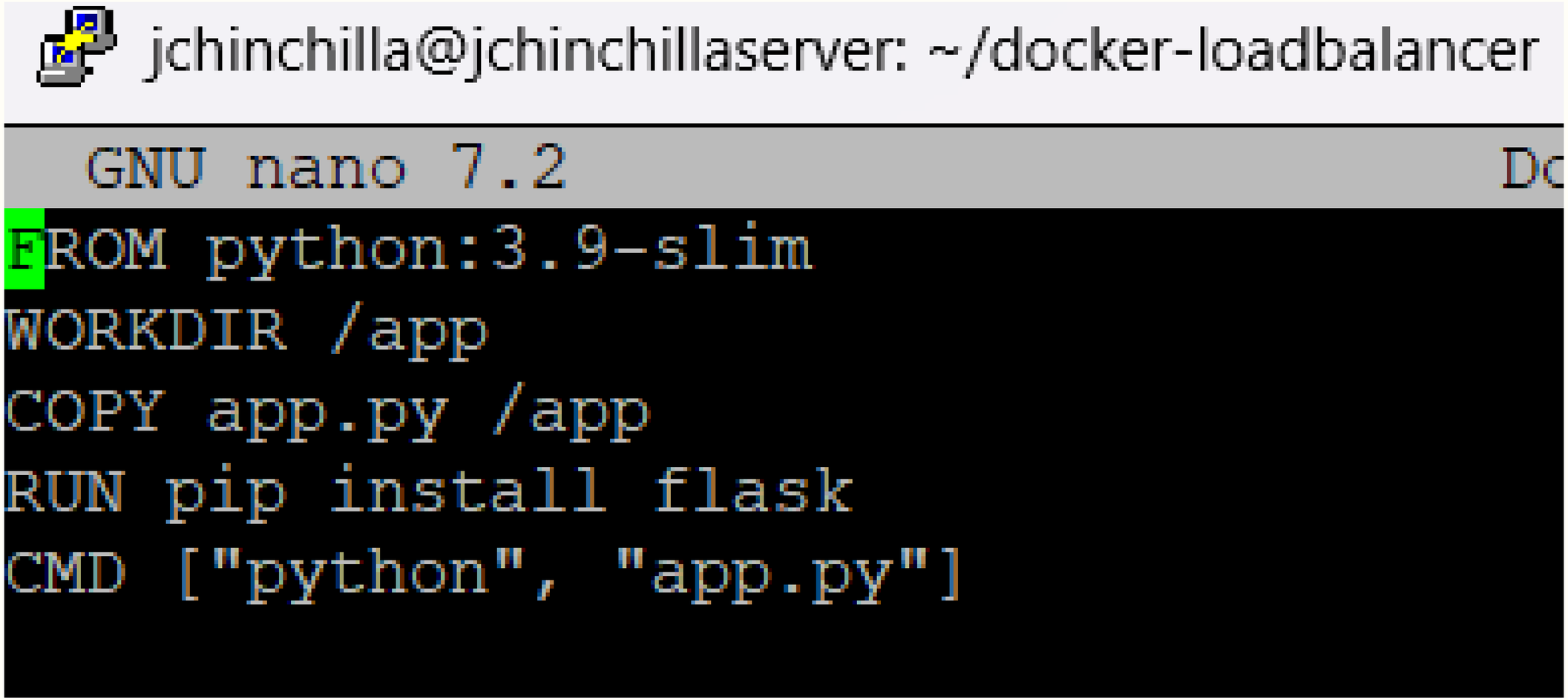
Aplicación Web

jchinchilla@jchinchillaserver: ~/docker-loadbalancer

GNU nano 7.2 app.py

```
from flask import Flask
import os

app = Flask(__name__)

@app.route("/")
def hello():
    return f"Hello from container {os.getenv('HOSTNAME')}!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Dockerfile


jchinchilla@jchinchillaserver: ~/docker-loadbalancer
GNU nano 7.2
FROM python:3.9-slim
WORKDIR /app
COPY app.py /app
RUN pip install flask
CMD ["python", "app.py"]

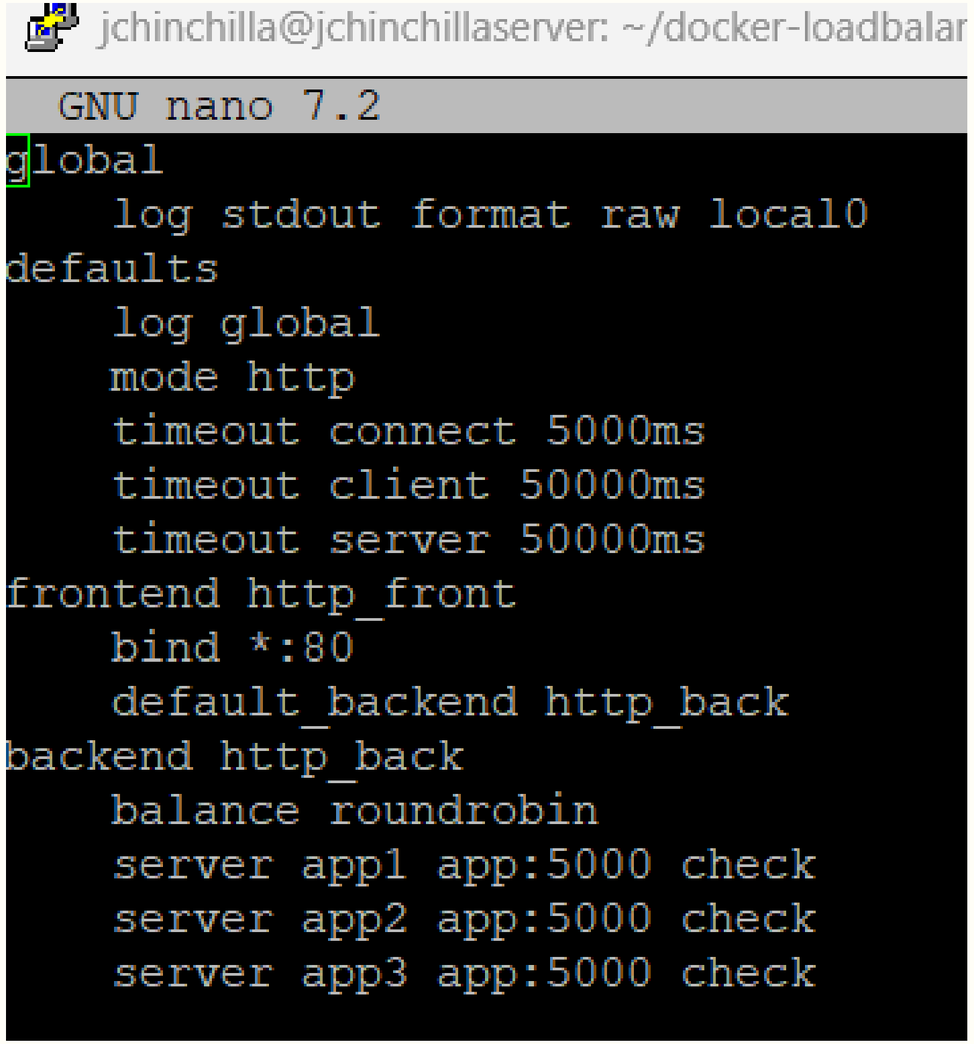
Docker Compose

jchinchilla@jchinchillaserver: ~/docker-loadbalancer

GNU nano 7.2 docker-compose.yml

```
version: '3.8'
services:
  app:
    image: myapp
    build:
      context: .
    deploy:
      replicas: 3
    environment:
      - HOSTNAME=${HOSTNAME}
  haproxy:
    image: haproxy:latest
    ports:
      - "80:80"
    volumes:
      - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
```

Configuración de HAProxy



docker-compose up --build

```
Successfully built 86ceba9ba471
Successfully tagged myapp:latest
Stopping and removing docker-loadbalancer_app_4 ... done
Stopping and removing docker-loadbalancer_app_5 ... done
Starting docker-loadbalancer_haproxy_1          ... done
Starting docker-loadbalancer_app_1              ... done
Starting docker-loadbalancer_app_2              ... done
Starting docker-loadbalancer_app_3              ... done
Attaching to docker-loadbalancer_haproxy_1, docker-loadbalancer_app_1, docker-l
er-loadbalancer_app_3
haproxy_1  | [NOTICE]   (1) : Initializing new worker (7)
haproxy_1  | [NOTICE]   (1) : Loading success.
app_1      |  * Serving Flask app 'app'
app_1      |  * Debug mode: off
app_1      | WARNING: This is a development server. Do not use it in a producti
oduction WSGI server instead.
app_1      |  * Running on all addresses (0.0.0.0)
app_1      |  * Running on http://127.0.0.1:5000
app_1      |  * Running on http://172.20.0.3:5000
app_1      | Press CTRL+C to quit
app_3      |  * Serving Flask app 'app'
app_3      |  * Debug mode: off
app_3      | WARNING: This is a development server. Do not use it in a producti
oduction WSGI server instead.
app_3      |  * Running on all addresses (0.0.0.0)
app_3      |  * Running on http://127.0.0.1:5000
app_3      |  * Running on http://172.20.0.4:5000
app_3      | Press CTRL+C to quit
app_2      |  * Serving Flask app 'app'
app_2      |  * Debug mode: off
app_2      | WARNING: This is a development server. Do not use it in a producti
oduction WSGI server instead.
app_2      |  * Running on all addresses (0.0.0.0)
app_2      |  * Running on http://127.0.0.1:5000
app_2      |  * Running on http://172.20.0.5:5000
app_2      | Press CTRL+C to quit
```

curl http://localhost

```
jchinchilla@jchinchillaserver:~/docker-lo
Hello from container !jchinchilla@jchinch

oadbalancer$ curl http://localhost
illaserver:~/docker-loadbalancer$
```

PRUEBAS DE RENDIMIENTO

SIEGE -C10 -R10 HTTP://LOCALHOST/

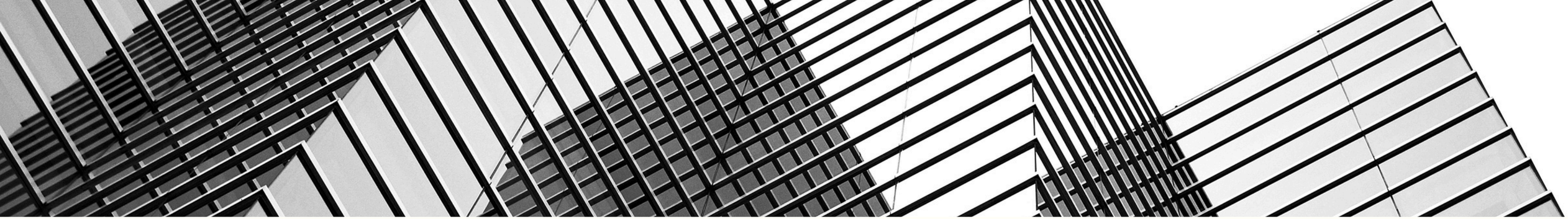
SIMULA 10 USUARIOS CON 10 SOLICITUDES CONCURRENTES.

```
New configuration template added to /home/jchinchilla/.siege
Run siege -C to view the current settings in that file

{       "transactions":                        100,
        "availability":                     100.00,
        "elapsed_time":                       0.34,
        "data_transferred":                   0.00,
        "response_time":                      0.03,
        "transaction_rate":                 294.12,
        "throughput":                         0.01,
        "concurrency":                        9.44,
        "successful_transactions":             100,
        "failed_transactions":                   0,
        "longest_transaction":                0.20,
        "shortest_transaction":               0.00
}
jchinchilla@jchinchillaserver:~/docker-loadbalancer$
```

SIEGE -C50 -R10 HTTP://LOCALHOST/

SIMULA 50 USUARIOS CON 10 SOLICITUDES CONCURRENTES.

```
jchinchilla@jchinchillaserver:~/docker-loadbalancer$ siege -c50 -r10 http://localhost/

{       "transactions":                        500,
        "availability":                      100.00,
        "elapsed_time":                        0.78,
        "data_transferred":                    0.01,
        "response_time":                       0.07,
        "transaction_rate":                  641.03,
        "throughput":                          0.01,
        "concurrency":                        44.38,
        "successful_transactions":             500,
        "failed_transactions":                   0,
        "longest_transaction":                 0.14,
        "shortest_transaction":                0.00
}
```



PROPUESTAS DE MEJORA

1. Servidor WSGI en Producción
   - Implementar Gunicorn o uWSGI para reemplazar el servidor de desarrollo Flask.
2. Seguridad en HAProxy
   - Configurar HTTPS con certificados SSL.
   - Restringir accesos no autorizados con reglas ACL.
3. Monitoreo y Logging
   - Implementar Prometheus y Grafana para monitorear rendimiento.
   - Configurar logs detallados en HAProxy.
4. Pruebas de Carga Adicionales
   - Realizar pruebas con Apache JMeter para analizar rendimiento bajo alta concurrencia.

GRACIAS