



PROYECTO VIRTUALIZACIÓN

Mar Rosado, Paola Mérida

FASE 1

Creación de un monolito

```
app.py ×
dockerize > flask > app > app.py > ...
1  from flask import Flask
2  import socket
3  app = Flask(__name__)
4
5  @app.route('/')
6  def hello():
7      return f"Proyecto \n Contenedor ID: {socket.gethostname()}"
8
9  if __name__ == '__main__':
10     app.run(host='0.0.0.0', port=8000)
11
```

Framework



FLASK

Lenguaje



PYTHON

FASE 2

Crear una imagen y un contenedor en Docker para la aplicación

 Dockerfile X

dockerize > flask > app >  Dockerfile > ...

```
1 FROM python:3.10-alpine
2
3 WORKDIR /app
4
5 COPY requirements.txt .
6 RUN pip3 install -r requirements.txt
7
8 COPY . .
9
10 EXPOSE 8000
11
12 CMD gunicorn --bind 0.0.0.0:8000 app:app
13
```

Tecnología de
contenedores



DOCKER

FASE 3

Escalar horizontalmente el monolito por medio de un balanceador de carga

nginx.conf X


dockerize > flask > nginx.conf

```
1  events {
2      worker_connections 1000;
3  }
4
5  http {
6      server{
7          listen 9999;
8
9          location / {
10             proxy_pass http://flask_app:8000;
11         }
12     }
13 }
```

Balanceador



NGNIX

 docker-compose.yml X

dockerize > flask >  docker-compose.yml

3 services:

28

29 flask_app:

30

image: awesome-flask

31

ports:

32

- "8000"

33

nginx:

34

image: nginx:latest

35

volumes:

36

- ./logs:/var/log/nginx

37

- "./nginx.conf:/etc/nginx/nginx.conf"

38

depends_on:

39

- flask_app

40

ports:

41

8000:8000

FASE 4

Creación de logs

```
docker-compose.yml X
dockerize > flask > docker-compose.yml
1  version: '3.3'
2
3  services:
4    grafana:
5      image: grafana/grafana-oss
6      container_name: grafana
7      restart: unless-stopped
8      ports:
9        - '3000:3000'
10     volumes:
11       - grafana_data:/var/lib/grafana
12    loki:
13      image: grafana/loki:2.9.0
14      container_name: grafana-loki
15      command: -config.file=/etc/loki/local-config.yaml
16      volumes:
17        - ./loki-config.yaml:/etc/loki/local-config.yaml
18      ports:
19        - "3100:3100"
20    promtail:
21      image: grafana/promtail:2.9.0
22      container_name: promtail
23      volumes:
24        - ./logs:/var/log/nginx
25        - ./promtail-config.yaml:/etc/promtail/config.yaml
```

Herramienta de
obtención de datos



GRAFANA

Almacenamiento de
logs



LOKI

access.log

dockerize > flask > logs > access.log

```
1 172.23.0.1 - - [04/Apr/2024:18:27:15 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
2 172.23.0.1 - - [04/Apr/2024:18:27:16 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
3 172.23.0.1 - - [04/Apr/2024:18:27:17 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
4 172.23.0.1 - - [04/Apr/2024:18:27:17 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
5 172.23.0.1 - - [04/Apr/2024:18:27:19 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
6 172.23.0.1 - - [04/Apr/2024:18:28:45 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
7 172.23.0.1 - - [04/Apr/2024:18:31:42 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
8 172.23.0.1 - - [04/Apr/2024:18:31:44 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
9 172.23.0.1 - - [04/Apr/2024:18:31:55 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
10 172.23.0.1 - - [04/Apr/2024:18:31:58 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
11 172.23.0.1 - - [04/Apr/2024:18:32:16 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
12 172.23.0.1 - - [04/Apr/2024:19:24:58 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
13 172.23.0.1 - - [04/Apr/2024:19:24:59 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
14 172.23.0.1 - - [04/Apr/2024:19:24:59 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
15 172.23.0.1 - - [04/Apr/2024:19:24:59 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
16 172.23.0.1 - - [04/Apr/2024:19:24:59 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
17 172.23.0.1 - - [04/Apr/2024:19:25:00 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
18 172.23.0.1 - - [04/Apr/2024:19:25:00 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
19 172.19.0.1 - - [05/Apr/2024:02:30:13 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
20 172.19.0.1 - - [05/Apr/2024:02:30:13 +0000] "GET /favicon.ico HTTP/1.1" 404 207 "http://localhost:9999/
21 172.19.0.1 - - [05/Apr/2024:02:31:06 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
22 172.19.0.1 - - [05/Apr/2024:02:31:07 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
23 172.19.0.1 - - [05/Apr/2024:02:31:08 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
24 172.19.0.1 - - [05/Apr/2024:02:31:09 +0000] "GET / HTTP/1.1" 200 38 "-" "Mozilla/5.0 (Windows NT 10.0;
```

FASE 5

Almacenamiento de logs
en la nube

```
app2.py X
dockerize > flask > logs > app2.py > ...

16 while True:
17     #inicializa pyrebase
18     counter=0
19     firebase = pyrebase.initialize_app(firebaseConfig)
20     db=firebase.database()
21     #verificacion
22     if not db.child('Logs').shallow().get().val():
23         #no hay nada aun en la base de datos
24         #1. Leer el archivo
25         f = open("access.log", "r")
26         lines = f.readlines()
27         #escribir archivo
28         count = 0
29         for line in lines:
30             count += 1
31             linea = line.strip()
32             data = {
33                 'LOG' : linea
34             }
35             result = Connection.post('/Logs/', data)
36             print(count)
37     else:
38         #Existen logs
39         #Consultar cantidad
40         counter = 0
41         result= db.child("Logs").get()
42         for res in result.each():
```

Base de datos



FIREBASE

Realtime Database

Datos

Reglas

Copias de seguridad

Uso

Extensions



Protege tus recursos de Realtime Database contra los abusos, como fraudes de facturación o phishing.

Configurar la Verificación de aplicaciones



<https://virtualizacion-991a2-default-rtdb.firebaseio.com>



<https://virtualizacion-991a2-default-rtdb.firebaseio.com/>

▼ — Logs

▼ — -NyM8MdZ1uou2t9oa7xN

LOG: "172.23.0.1 - - [04/Apr/2024:18:27:15 +0000] "GET / HTTP/1.1" 200

▼ — -NyM8MjWxSrzXzsqK-pV

FASE 6

Automatización de procesos

```
main.tf x
dockerize > flask > app > main.tf
1  # Specifying Docker provider
2
3  terraform {
4      required_providers {
5          docker = {
6              source = "kreuzwerker/docker"
7              version = "2.23.1"
8          }
9      }
10 }
11
12 resource "null_resource" "run_command" {}
13     triggers = {
14         always_run = "${timestamp()}"
15     }
16
17     provisioner "local-exec" {
18         command = "docker-compose up -d --scale flask_app=3"
19     }
20 }
```

Programa para
gestionar IaC



TERRAFORM

DEMOSTRACIÓN DE FUNCIONAMIENTO



PROYECTO DE VIRTUALIZACIÓN



Copy link

PROYECTO DE VIRTUALIZACIÓN



Mar Rosado, Paola Mérida

Watch on  YouTube

CONCLUSIONES



NGINX

Actúa como un balanceador de carga para distribuir el tráfico de las solicitudes y alternar entre los contenedores que están activos para hacer un mejor uso de los recursos.

DOCKER

Docker permite tener una gestión ordenada y aislada de contenedores para poder ejecutar diversas aplicaciones.

GRAFANA

Grafana es una herramienta de visualización de datos que se utiliza para monitorizar y analizar datos, es sencilla de usar, flexible, escalable y de código abierto.

LOKI

Loki es un sistema de almacenaje que puede almacenar y consultar sus logs de forma eficiente y escalable.

TERRAFORM

Terraform es una herramienta que nos permite automatizar los procesos de nuestra aplicación, utilizando comandos, docker y diversos recursos.