

# Práctica 1: Micro-servicios

## Parte 1: REST API en contenedor

DANIEL AQUINO SANTIAGO y ENRIQUE GÓMEZ  
FERNÁNDEZ

P10

### Tarea: entorno python

```
e460502@6A-7-6-7:~$ mkdir -p venv/siip1
e460502@6A-7-6-7:~$ python3 -m venv venv/siip1
e460502@6A-7-6-7:~$ source ./venv/siip1/bin/activate
(siip1) e460502@6A-7-6-7:~$ pip install hypercorn quart localstack awscli awscli-local
Collecting hypercorn
  Downloading hypercorn-0.14.4-py3-none-any.whl (58 kB)
    ━━━━━━━━━━━━━━━━ 58.3/58.3 kB 3.1 MB/s eta 0:00:00
Collecting quart
  Downloading quart-0.19.3-py3-none-any.whl (77 kB)
    ━━━━━━━━━━━━━━━━ 77.9/77.9 kB 3.5 MB/s eta 0:00:00
Collecting localstack
  Downloading localstack-2.3.2.tar.gz (5.3 kB)
  Preparing metadata (setup.py) ... done
Collecting awscli
  Downloading awscli-1.29.66-py3-none-any.whl (4.3 MB)
    ━━━━━━━━━━━━━━ 4.3/4.3 kB 25.1 MB/s eta 0:00:00
Collecting awscli-local
  Downloading awscli-local-0.21.tar.gz (11 kB)
  Preparing metadata (setup.py) ... done
Collecting wsproto>=0.14.0
  Downloading wsproto-1.2.0-py3-none-any.whl (24 kB)
Collecting priority
  Downloading priority-2.0.0-py3-none-any.whl (8.9 kB)
Collecting h2>=3.1.0
  Downloading h2-4.1.0-py3-none-any.whl (57 kB)
    ━━━━━━━━━━━━━━ 57.5/57.5 kB 3.7 MB/s eta 0:00:00
Collecting h11
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
    ━━━━━━━━━━━━━━ 58.3/58.3 kB 3.2 MB/s eta 0:00:00
Collecting httpcore
```

### Tarea: revisión de user\_rest.py

#### Pregunta: user\_rest.py

1. ¿Qué crees que hacen las anotaciones `@app.route`, `@app.get`, `@app.put`?

`@app.route` se usa para definir una ruta y asociar funciones con métodos HTTP en esa ruta.

`@app.get` se usa para definir la ruta y asociar funciones al método GET.

`@app.put` se usa para definir la ruta y asociar funciones al el método PUT

2. ¿Qué ventajas o inconvenientes crees que pueden tener?  
Proporciona organización al código y un control específico sobre rutas y métodos HTTP. Sin embargo, en códigos grandes, con muchas rutas, se puede complicar el seguimiento de la lógica de la aplicación.

### Tarea: ejecución de user\_rest.py

## Pregunta: ejecución de user\_rest.py

```
(si1p1) e460502@4-31-4-31:~$ source venv/si1p1/bin/activate
(si1p1) e460502@4-31-4-31:~$ [[ -d build/users ]] || mkdir -p build/users
(si1p1) e460502@4-31-4-31:~$ QUART_APP=src.user_rest:app quart run -p 5050
  * Serving Quart app 'src.user_rest'
  * Debug mode: False
  * Please use an ASGI server (e.g. Hypercorn) directly in production
  * Running on http://127.0.0.1:5050 (CTRL + C to quit)
[2023-10-19 14:25:46 +0200] [5562] [INFO] Running on http://127.0.0.1:5050 (CTRL + C to quit)
```

1. ¿Qué crees que ha sucedido?

Al ejecutar los comandos, se detecta un error de sintaxis en el -p por lo que no funciona. Sin embargo, si lo cambiamos por un guión(-p) se ejecuta correctamente. En el primer comando, estamos activando el entorno virtual; en el segundo, se crea el directorio “build/users” si este no está creado; finalmente, el tercer comando bien escrito inicia el programa “user\_rest.py” como una aplicación quart en el puerto 5050.

## Tarea: petición a user\_rest.py

### Pregunta: petición a user\_rest.py

```
e460502@4-31-4-31:~$ curl -X PUT \
http://localhost:5050/user/myusername \
-H 'Content-Type: application/json' \
-d '{"firstName": "myFirstName"}'
{"status": "OK"}
```

Revisa la ayuda de curl (man curl) y contesta:

1. ¿Qué crees que ha sucedido al ejecutar el mandato curl?

Según el manual de curl, la opción de -X, especifica un método de solicitud personalizado que se utilizará cuando se comunique. Por ello, creemos que se está comunicando a la aplicación de user\_rest.py que se encuentra en ejecución en el puerto 5050, y que gracias a esto, se sigan los pasos de la instrucción PUT del código.

2. ¿Cuál es la respuesta del microservicio?

```
(si1p1) e460502@4-31-4-31:~$ QUART_APP=src.user_rest:app quart run -p 5050
  * Serving Quart app 'src.user_rest'
  * Debug mode: False
  * Please use an ASGI server (e.g. Hypercorn) directly in production
  * Running on http://127.0.0.1:5050 (CTRL + C to quit)
[2023-10-19 14:25:46 +0200] [5562] [INFO] Running on http://127.0.0.1:5050 (CTRL + C to quit)
[2023-10-19 14:27:13 +0200] [5562] [INFO] 127.0.0.1:46098 PUT /user/myusername 1.1 200 16 4665
```

Aquí podemos observar la respuesta del microservicio, la cual imprime que se ha realizado un PUT, la ruta en la que se hace y la fecha.

3. ¿Se ha generado algún archivo? ¿Cuál es su ruta y contenido?

Sí, se ha generado un archivo llamado data.json en una ruta como esta:

./build/users/myusername/data.json y al abrir el archivo vimos en su contenido se encontraba la siguiente cadena: {"firstName": "myFirstName", "username": "myusername"}

## Tarea: petición HTTP

### Pregunta: petición HTTP

1. ¿Cuál es la petición HTTP que llega al programa nc?

Para este ejercicio, primero miramos el manual de nc, en el cual vimos que para ponerlo en modo escucha en el puerto 8888 debíamos ejecutar el comando: “nc -l 8888” ya que la etiqueta -l es la que pone en modo escucha al programa nc. Tras esto, ejecutamos desde otra terminal el mismo comando, pero indicando en este que el puerto es el 8888:

```
enrique@enrique-VirtualBox:~$ curl -X PUT http://localhost:8888/users/myusername  
-H 'Content-Type: application/json' -d '{"firstName": "enrique"}'
```

La petición que llega al programa nc es la siguiente:

```
enrique@enrique-VirtualBox:~$ nc -l 8888  
PUT /users/myusername HTTP/1.1  
Host: localhost:8888  
User-Agent: curl/8.1.2  
Accept: */*  
Content-Type: application/json  
Content-Length: 24  
  
{"firstName": "enrique"}■
```

2. ¿En qué campo de la cabecera HTTP se indica el tipo de dato del cuerpo (body) de la petición y cuál es ese tipo?

El tipo de dato del cuerpo de la petición se indica en el campo de cabecera llamado Content-type. En este caso, se puede ver que es application/json, lo que indica que el cuerpo de la petición contiene datos en formato JSON, dado que ha sido indicado en el propio curl.

3. ¿Qué separa la cabecera del cuerpo?

Como se puede apreciar en la imagen de la pregunta 1, vemos que lo que separa la cabecera del cuerpo es una línea completamente vacía después de la última línea de la cabecera (“Content-Length”). Es en la siguiente donde empieza el cuerpo.

## Tarea: prototipo user

1. Implementa el método **DELETE** para eliminar un usuario

```

@app.delete('/user/<username>')
async def user_delete(username):
    resp={}
    try:
        os.remove(f"{USERDIR}/{username}/data.json")
        os.rmdir(f"{USERDIR}/{username}")
    except Exception as e:
        resp["status"] = "KO"
        resp["error"] = f"Error borrando user dir: {str(e)}"
        return await make_response(jsonify(resp), 400)
    resp["status"] = "OK"
    return await make_response(jsonify(resp), 200)

```

Esta función primero borra el archivo “.json” correspondiente al usuario indicado, y además se borra el directorio de dicho usuario, borrando por completo dicho usuario. Además se ha implementado un control de errores, la parte de except.

## 2. Implementa el método PATCH para actualizar algún campo del usuario

```

@app.patch('/user/<username>')
async def user_patch(username):
    resp = {}
    try:
        with open(f'{USERDIR}/{username}/data.json', "r") as ff:
            user_data = json.loads(ff.read())
        patch_data = await request.get_json()
        for key, value in patch_data.items():
            user_data[key] = value
        with open(f'{USERDIR}/{username}/data.json', "w") as ff:
            ff.write(json.dumps(user_data))
        resp["status"] = "OK"
    except Exception as e:
        resp["status"] = "KO"
        resp["error"] = f"Error actualizando usuario: {str(e)}"
        return await make_response(jsonify(resp), 400)
    return await make_response(jsonify(resp), 200)

```

En esta función actualizamos el contenido de los campos modificables del archivo .json de un usuario, para ello obtenemos el valor de su data.json, y leyendo la clave valor del campo modificable para después asignar a dicho campo el valor que nosotros deseemos.

**Tarea: ejecución hypercorn**

**Pregunta: ejecución hypercorn**

```
e460502@4-31-4-31:~$ hypercorn --bind 0.0.0.0:8080 --workers 2 src.user_rest:app
[2023-10-19 14:31:46 +0200] [5808] [INFO] Running on http://0.0.0.0:8080 (CTRL + C to quit)
[2023-10-19 14:31:46 +0200] [5809] [INFO] Running on http://0.0.0.0:8080 (CTRL + C to quit)
```

1. ¿Qué crees que ha sucedido?

Se ejecuta Hypercorn

2. ¿Qué petición curl debes hacer ahora para hacer un GET de un usuario? ¿Qué ha cambiado respecto de la ejecución sin hypercorn?

"curl -X GET \  
http://localhost:8080/user/myusername"

```
e460502@4-31-4-31:~$ curl -X GET \
http://localhost:8080/user/myusername
{"firstName": "myFirstName", "username": "myusername"}•
```

Lo que ha variado con respecto a la ejecución sin hypercorn, es que las solicitudes al servidor ahora se encuentran encriptadas, a diferencia de ejecutar QUART, donde se podían observar a tiempo real las solicitudes que el servidor estaba recibiendo.

## Tarea: arranque de docker rootless

```
e460502@4-31-4-31:~$ man subuid
e460502@4-31-4-31:~$ sudo /usr/local/bin/slZsetuid.sh 3000000:100000 $USER
[sudo] password for e460502:
Setting /etc/subuid for e460502
Setting /etc/resolv.conf
e460502@4-31-4-31:~$ dockerd-rootless-setuptool.sh install
[INFO] Creating /home/alumnos/e460502/.config/systemd/user/docker.service
[INFO] starting systemd service docker.service
+ systemctl --user start docker.service
+ sleep 3
+ systemctl --user --no-pager --full status docker.service
● docker.service - Docker Application Container Engine (Rootless)
   Loaded: loaded (/home/alumnos/e460502/.config/systemd/user/docker.service; disabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-10-19 14:38:12 CEST; 3s ago
       Docs: https://docs.docker.com/go/rootless/
 Main PID: 6119 (rootlesskit)
   Tasks: 1
    Memory: 130.8M
      CPU: 947ms
     CGroup: /user.slice/user-16168.slice/user@16168.service/app.slice/docker.service
           ├─6119 rootlesskit --net=slirp4netns --mtu=65520 --slirp4netns-sandbox=auto --slirp4netns-seccomp=auto --disable-host-loopback --port-driver=builtin --copy-up=/etc --copy-up=/run --propagation=slave /usr/bin/dockerd-rootless.sh
           ├─6130 /proc/self/exe --net=slirp4netns --mtu=65520 --slirp4netns-sandbox=auto --slirp4netns-seccomp=auto --disable-host-loopback --port-driver=builtin --copy-up=/etc --copy-up=/run --propagation=slave /usr/bin/dockerd-rootless.sh
           ├─6148 slirp4netns --mtu 65520 -r 3 --disable-host-loopback --enable-sandbox --enable-seccomp 6130 tap0
           └─6155 dockerd
           ├─6174 containerd --config /run/user/16168/docker/containerd.toml
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.424345516+02:00" level=warning msg="Running modprobe bridge br_netfilter failed with message: modprobe: E
rror: Could not insert 'bridge' into /lib/modules/5.15.0-46-generic/kernel/net/bridge/br_netfilter.ko \n, error: exit status 1"
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.424420104+02:00" level=info msg="skipping fwrulewd management for rootless mode"
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.783784838+02:00" level=info msg="Loading containers: done."
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.876423185+02:00" level=warning msg="Not using native diff for overlay2, this may cause degraded performance for building images: running in a user namespace" storage-driver=overlay2
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.876861800+02:00" level=warning msg="WARNING: bridge-nf-call-iptables is disabled"
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.876895289+02:00" level=warning msg="WARNING: bridge-nf-call-ip6tables is disabled"
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.876937706+02:00" level=info msg="Docker daemon" commit=659604f graphdriver=overlay2 version=24.0.2
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.877160293+02:00" level=info msg="Daemon has completed initialization"
oct 19 14:38:12 4-31-4-31 dockerd-rootless.sh[6155]: time="2023-10-19T14:38:12.9996653123+02:00" level=info msg="API listen on /run/user/16168/docker.sock"
oct 19 14:38:12 4-31-4-31 systemd[2530]: Started Docker Application Container Engine (Rootless).
+ DOCKER_HOST=unix:///run/user/16168/docker.sock /usr/bin/docker version
Client: Docker Engine - Community
Version:          24.0.2
API version:      1.49
Go version:       go1.20.4
Git commit:       c574dfc
Built:            Thu May 25 21:51:00 2023
OS/Arch:          linux/amd64
Context:          default
Server: Docker Engine - Community
Engine:
```

## Pregunta: modo rootless

1. Revisa el contenido de /etc/subuid y la documentación de docker rootless: ¿para qué crees que sirve ese archivo?  
El archivo /etc/subuid se utiliza en sistemas Linux que ejecutan Docker en modo rootless para gestionar las asignaciones de UIDs que docker utiliza en los contenedores.

## Tarea: docker info

```
e460502@4-31-4-31:~$ docker info
Client: Docker Engine - Community
Version: 24.0.2
Context: rootless
Debug Mode: false
Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version: v0.10.5
    Path: /usr/libexec/docker/cli-plugins/docker-buildx
compose: Docker Compose (Docker Inc.)
  Version: v2.18.1
  Path: /usr/libexec/docker/cli-plugins/docker-compose

Server:
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 24.0.2
Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: false
  userxattr: true
Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 2
Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
    Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 3dce8eb055cbb6872793272b4f20ed16117344f8
runc version: v1.1.7-0-g860f061
init version: de40ad0
Security Options:
  seccomp
    Profile: builtin
  rootless
  cgroups
Kernel Version: 5.19.0-46-generic
Operating System: Ubuntu 22.04.2 LTS
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 7.186GiB
Name: 4-31-4-31
```

## Pregunta: docker info

1. ¿Dónde se guarda la configuración del demonio arrancado?

La configuración del demonio se guarda en el PATH :  
 "/home/alumnos/e462883/.config/systemd/user"

2. ¿Dónde se almacenan los contenedores?

Los contenedores se almacenan en :  
 "/run/user/16253/docker/containerd/daemon"

```
e462883@6A-13-6-13:/run/user/16253/docker/containerd/daemon$ ls
io.containerd.runtime.v1.linux  io.containerd.runtime.v2.task
e462883@6A-13-6-13:/run/user/16253/docker/containerd/daemon$ █
```

3. ¿Qué tecnología de almacenamiento se usa para los contenedores?

¿Qué es copy-on-write? ¿Lo usa docker?

- La tecnología utilizada es extfs, que es un sistema de archivos extendidos para el sistema operativo linux  
 Un extent es un conjunto de bloques físicos contiguos, mejorando el rendimiento al trabajar con ficheros de gran tamaño y reduciendo la fragmentación.
- técnica que retrasa la duplicación de datos hasta que sea necesario, en lugar de copiar datos inmediatamente se comparten referencias a los mismos datos entre procesos lo que ahorra recursos de memoria y tiempo de copia.

- Si, lo usa en su funcionalidad , cuando se crea un contenedor a partir de una imagen, para minimizar el uso de recursos.

## Tarea: docker info

### Pregunta: run whoami

```
e462883@6A-13-6-13:~$ docker run alpine whoami
root
e462883@6A-13-6-13:~$
```

1. ¿Qué crees que ha sucedido?

Esta función inicia un contenedor, basado en una imagen del Linux Alpine y ejecuta el comando `whoami` que responde diciendo el cual es el usuario ejecutando la orden en dicho contenedor.

2. ¿Crees que el usuario root del contenedor es el mismo que el del host?  
No, el usuario root dentro de un contenedor no es el mismo ya que los contenedores están diseñados para ser aislados y compartimentados, lo que significa que tienen su propio espacio de nombres para usuarios y procesos. Esto es una característica fundamental de Docker ya que permite que múltiples contenedores se ejecuten en el mismo sistema host sin interferir entre sí.

## Tarea: list images

```
e462883@6A-13-6-13:~$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
alpine          latest        8ca4688f4f35  7 days ago    7.33MB
ubuntu          22.04        3565a89d9e81  11 days ago   77.8MB
ubuntu          latest        3565a89d9e81  11 days ago   77.8MB
```

### Pregunta: list images

1. ¿Qué crees que tiene que ver con el mandato `docker pull`?

`Docker images`, funciona como un listado de imágenes que están almacenadas en tu sistema, `docker pull`, se utiliza para obtener imágenes desde un registro remoto. Por lo tanto, se relacionan en que `docker images` listará las imágenes cargadas con `docker pull`.

## Tarea: docker ps

```
e462883@6A-13-6-13: $ docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED        STATUS          PORTS     NAMES
efa40b58834f   alpine    "whoami"     10 minutes ago  Exited (0) 10 minutes ago  hopeful_maxwell
6795ee88e84b   ubuntu    "/bin/bash"   55 minutes ago  Exited (0) 55 minutes ago  u22.04
74ca992cb168   alpine    "whoami"     56 minutes ago  Exited (0) 56 minutes ago  confident_wright
e462883@6A-13-6-13: $
```

### Pregunta: docker ps

1. ¿Qué crees que muestra la salida?

La salida muestra la actividad en las distintas imágenes guardadas en el sistema, incluyendo de esta manera como se puede observar los comandos realizados en cada una de ellas. También podemos observar toda la información de estos contenedores, así como la imagen, el comando ejecutado, hace cuánto se creó, el estado del comando, los puertos y los nombres de dicho contenedor.

## Tarea: docker pull y run -ti

```
e462883@12-21-12-21:~$ docker pull ubuntu:22.04
22.04: Pulling from library/ubuntu
37aaaf24cf781: Pull complete
Digest: sha256:9b8dec3bf938bc80fbe758d856e96fd fab5f56c39d44b0cff351e847bb1b01ea
Status: Downloaded newer image for ubuntu:22.04
docker.io/library/ubuntu:22.04

e462883@12-21-12-21:~$ docker run -ti --name u22.04 ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
Digest: sha256:9b8dec3bf938bc80fbe758d856e96fd fab5f56c39d44b0cff351e847bb1b01ea
Status: Downloaded newer image for ubuntu:latest
```

## Pregunta: docker pull y run -ti

1. ¿Qué hacen las opciones -ti?

-t: permite una interacción directa con el contenedor como si estuvieras en una sesión de terminal  
-i: permite la interacción continua con el contenedor manteniendo la entrada del terminal abierta

2. ¿Qué ha sucedido?

Docker primero intenta buscar la imagen “ubuntu:latest” localmente pero no la encuentra por lo que la descarga desde Docker Hub, e inicia el contenedor con el nombre especificado con –name que en este caso es u22.04.

## Tarea: docker inspect

```
e462883@12-21-12-21:~$ docker inspect u22.04 | jq '.[].Config.Cmd[]'
"/bin/bash"
```

## Pregunta: docker inspect

1. ¿Qué hace el mandato jq?

Es una herramienta que permite realizar consultas y transformaciones en documentos JSON, en nuestro caso se utiliza para extraer la propiedad .Config.Cmd del resultado de docker inspect que contiene el comando que se ejecutó cuando se creó el contenedor.

2. ¿Qué tiene que ver el resultado del mandato con la tarea anterior?

Está extrayendo y mostrando el comando que se ejecutó cuando se creó el contenedor u22.04 el resultado que hemos obtenido hace referencia al comando anterior que indica que se ha ejecutado una shell interactiva en el contenedor.

## Tarea: docker star

## Pregunta: docker start

1. ¿Qué ha sucedido?

```
e462883@12-21-12-21:~$ docker start u22.04
u22.04
```

```
e462883@12-21-12-21:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS
S      NAMES
ab1074e9b1e5      ubuntu      "/bin/bash"   16 minutes ago   Up About a minute
u22.04
```

Como podemos comprobar el contenedor está en ejecución.

La salida del comando muestra que la ejecución se realizó con éxito, mostrando el nombre del contenedor iniciado.

## 2. ¿Cómo se borra el contenedor u22.04?

Para eliminar el contenedor primero deberemos detener dicho contenedor realizando: docker stop u22.04

```
e462883@12-21-12-21:~$ docker stop u22.04
u22.04
e462883@12-21-12-21:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS
S      NAMES
```

Como podemos observar hemos detenido la ejecución del docker, y al listar los contenedores en ejecución ya no nos aparece

Y a continuación, borrar dicho contenedor con: docker rm u22.04

```
e462883@12-21-12-21:~$ docker rm u22.04
u22.04
```

## Tarea: docker run -d

```
e462883@12-21-12-21:~$ docker run -ti -d --name u22.04d ubuntu
699517a5a1b834689f7208cb6c172ee4c23834fd2f1b0baeec2f0618dc917279
e462883@12-21-12-21:~$
```

## Pregunta: docker run -d

### 1. ¿Qué hace la opción -d?

Esta opción ejecuta el contenedor en modo daemon, el cual se ejecuta en segundo plano lo que significa que no muestra la salida en la terminal, y la libera.

## Tarea: docker exec

```
e462883@12-21-12-21:~$ docker exec -ti u22.04d /bin/bash
root@699517a5a1b8:/# 
```

```
e462883@12-21-12-21:~$ docker exec -ti u22.04d /bin/bash
root@699517a5a1b8:/# mkdir -p /si1/users
root@699517a5a1b8:/# touch /si1/users/test
```

## Pregunta: docker exec

### 1. ¿Qué ha sucedido?

Se ha ejecutado una shell interactiva dentro del contenedor y a posteriori en dicha shell se ha creado un directorio con ruta /si1/users, en el cual se ha creado después un archivo test. Todo ello dentro del sistema de archivos de dicho contenedor.

### 2. Sal del contenedor

```
root@699517a5a1b8:/# exit  
exit  
e462883@12-21-12-21:~$
```

## Tarea: docker volume

```
e462883@12-21-12-21:~$ [[ -d si1/users/testv ]] || mkdir -p si1/users/testv  
e462883@12-21-12-21:~$ docker run -ti -v ${PWD}/si1/users/testv:/si1/users/testv  
alpine \  
touch /si1/users/testv/afile  
e462883@12-21-12-21:~$ docker run -ti -v ${PWD}/si1/users/testv:/si1/users/testv  
alpine \  
ls -al /si1/users/testv/afile  
-rw-r--r-- 1 root root 0 Oct 11 08:59 /si1/users/testv/afile  
e462883@12-21-12-21:~$ ls -al si1/users/testv/afile  
-rw-r--r-- 1 e462883 alumnos 0 oct 11 10:59 si1/users/testv/afile  
e462883@12-21-12-21:~$ echo "a test" >>si1/users/testv/afile  
e462883@12-21-12-21:~$ docker run -ti -v ${PWD}/si1/users/testv:/si1/users/testv  
alpine \  
cat /si1/users/testv/afile  
a test
```

## Pregunta: docker volume

### 1. ¿Qué hace la opción -v?

Se utiliza para montar un volumen en un contenedor docker. Permite vincular una ubicación en el sistema host a una ubicación en el sistema de archivos del contenedor lo que afecta positivamente a la persistencia de datos e intercambios de datos entre el sistema host y el contenedor

### 2. ¿Qué ha sucedido al ejecutar los mandatos anteriores?

Primero se verifica que el directorio si1/users/testv esté presente en el sistema host

Después se ejecuta un contenedor alpine con la opción -v para montar el directorio del sistema host en el contenedor

A continuación, se crea un archivo llamado afiche en la ubicación anterior mencionada dentro del contenedor

Se verifica la existencia de dicho archivo creado en el directorio compartido entre el sistema host y el contenedor

Se agrega el texto a fula dentro del mismo archivo

Se ejecuta nuevamente el contenedor y se muestra el contenido del archivo afiche, mostrando por pantalla el mensaje "a file"

Demostrando todo esto el correcto funcionamiento de la conexión entre el sistema host y el contenedor creando un archivo desde el sistema host, y observando su contenido desde el propio contenedor.

## Tarea: limpieza

### Pregunta: limpieza

1. ¿Qué mandato has usado para eliminar los contenedores?

```
e462883@12-21-12-21:~$ docker rm -f $(docker ps -aq)
7de7945ccbd8
f54b85c84c7b
1fa6f445ca80
699517a5a1b8
d42efc9ea144
6f6eadc004ed
```

Este comando borra los todos los contenedores listados con docker ps -aq, los cuales son todos los existentes, en ejecución o no en ejecución.

2. ¿Qué mandato has usado para eliminar las imágenes?

```
e462883@12-21-12-21:~$ docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: alpine:latest
untagged: alpine@sha256:eece025e432126ce23f223450a0326fbebe39cdf496a85d8c016293
fc851978
deleted: sha256:8ca4688f4f356596b5ae539337c9941abc78eda10021d35cbc52659c74d9b443
deleted: sha256:cc2447e1835a40530975ab80bb1f872fbab0f2a0faecf2ab16fbbb89b3589438
untagged: ubuntu:22.04
untagged: ubuntu:latest
untagged: ubuntu@sha256:9b8dec3bf938bc80fbe758d856e96fdfab5f56c39d44b0cff351e847
bb1b01ea
deleted: sha256:3565a89d9e81a4cb4cb2b0d947c7c11227a3f358dc216d19fc54bfd77cd5b542
deleted: sha256:01d4e4b4f381ac5a9964a14a650d7c074a2aa6e0789985d843f8eb3070b58f7d

Total reclaimed space: 85.15MB
```

Esto borra las imágenes que ya no están en uso (imágenes huérfanas) sin contenedor, los cuales borramos previamente.

## Tarea: Dockerfile

### Pregunta: Dockerfile

1. ¿Qué hace la sentencia FROM?

La instrucción FROM inicializa una nueva etapa de construcción y configura la imagen base para instrucciones posteriores. Un Dockerfile válido debe comenzar con una instrucción FROM.

2. ¿Qué hace la sentencia ENV?

La instrucción ENV establece la variable de entorno <key> en el valor <value> (**ENV <key>=<value>**).

3. ¿Qué diferencia presenta ENV respecto de ARG?

ENV se utiliza para definir variables de entorno que afectan al entorno de ejecución del contenedor mientras que ARG, define argumentos que no están disponibles en tiempo de ejecución y no persisten en la imagen final.

**4. ¿Qué hace la sentencia RUN?**

Ejecuta cualquier comando en una nueva capa encima de la imagen actual y confirma los resultados.

**5. ¿Qué hace la sentencia COPY?**

Copia archivos desde el sistema de archivos del host al del contenedor donde permanecen independientes en su propia capa y no se invalidan cuando se cambian los comandos de las capas anteriores.

**6. ¿Qué otra sentencia del Dockerfile tiene un cometido similar a COPY?**

La sentencia ADD también se utiliza para copiar archivos, sin embargo este puede descomprimir los archivos y descargarlos desde una URL.

**7. ¿Qué hace la sentencia EXPOSE?**

Informa a Docker que el contenedor escucha en los puertos de red especificados en tiempo de ejecución.

**8. ¿Qué hace la sentencia CMD?**

Proporcionar valores predeterminados para un contenedor en ejecución.

**9. ¿Qué otras sentencias del Dockerfile tienen un cometido similar a CMD?**

La sentencia RUN ejecuta un comando y confirma el resultado; CMD no ejecuta nada en el momento de la compilación, pero especifica el comando previsto para la imagen.

Al igual que la sentencia ENTRYPOINT, que se utiliza para definir comandos predeterminados, sin embargo, este no se puede hacer en tiempo de ejecución.

## Tarea: docker build

```
(si1p1) e462883@12-21-12-21:~/Downloads/src$ docker build --tag si1p1:latest .
[+] Building 60.3s (12/12) FINISHED
=> [internal] load .dockerignore                                         0.1s
=> => transferring context: 2B                                         0.0s
=> [internal] load build definition from Dockerfile                  0.1s
=> => transferring dockerfile: 353B                                    0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest      1.3s
=> [internal] load build context                                      0.1s
=> => transferring context: 1.13kB                                    0.0s
=> [1/7] FROM docker.io/library/ubuntu@sha256:9b8dec3bf938bc80fbe758d856 3.5s
=> => resolve docker.io/library/ubuntu@sha256:9b8dec3bf938bc80fbe758d856 0.1s
=> => sha256:b4b521bfcecc90b11d2869e00fe1f2380c21cbfc799ee35 424B / 424B 0.0s
=> => sha256:3565a89d9e81a4cb4cb2b0d947c7c11227a3f358dc2 2.30kB / 2.30kB 0.0s
=> => sha256:37aaaf24cf781dcc5b9a4f8aa5a99a40b60ae45d64 29.54MB / 29.54MB 0.9s
=> => sha256:9b8dec3bf938bc80fbe758d856e96fd fab5f56c39d4 1.13kB / 1.13kB 0.0s
=> => extracting sha256:37aaaf24cf781dcc5b9a4f8aa5a99a40b60ae45d64dcb4f6d 0.7s
=> [2/7] RUN apt update                                              4.2s
=> [3/7] RUN apt install -y python3-pip                            45.0s
=> [4/7] RUN pip install hypercorn quart                           3.0s
=> [5/7] RUN mkdir -p /si1/build/users                           0.4s
=> [6/7] WORKDIR /si1                                           0.1s
=> [7/7] COPY user_rest.py ./                                       0.1s
=> exporting to image                                              2.5s
=> => exporting layers                                            2.5s
=> => writing image sha256:325b1e70e0d22cc29c6c8695a24d43722679e753b211f 0.0s
=> => naming to docker.io/library/si1p1:latest                   0.0s
```

```
(si1p1) e462883@12-21-12-21:~/Downloads/src$ docker build --tag si1p1:1.0 .
[+] Building 0.5s (12/12) FINISHED
=> [internal] load .dockerignore                                         0.0s
=> => transferring context: 2B                                         0.0s
=> [internal] load build definition from Dockerfile                  0.0s
=> => transferring dockerfile: 353B                                    0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest      0.3s
=> [internal] load build context                                      0.0s
=> => transferring context: 34B                                     0.0s
=> [1/7] FROM docker.io/library/ubuntu@sha256:9b8dec3bf938bc80fbe758d856 0.0s
=> CACHED [2/7] RUN apt update                                         0.0s
=> CACHED [3/7] RUN apt install -y python3-pip                         0.0s
=> CACHED [4/7] RUN pip install hypercorn quart                          0.0s
=> CACHED [5/7] RUN mkdir -p /si1/build/users                           0.0s
=> CACHED [6/7] WORKDIR /si1                                         0.0s
=> CACHED [7/7] COPY user_rest.py ./                                     0.0s
=> exporting to image                                              0.0s
=> => exporting layers                                            0.0s
=> => writing image sha256:325b1e70e0d22cc29c6c8695a24d43722679e753b211f 0.0s
=> => naming to docker.io/library/si1p1:1.0                        0.0s
```

```
(si1p1) e462883@12-21-12-21:~/Downloads/src$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
si1p1           1.0          325b1e70e0d2  2 minutes ago  492MB
si1p1           latest        325b1e70e0d2  2 minutes ago  492MB
```

## Pregunta: docker build

1. ¿Por qué es mucho más rápido la creación de la segunda imagen?

Se debe a la reutilización de capas de caché, la optimización del sistema de archivos y otros factores que hacen que la segunda construcción sea más eficiente en términos de recursos y tiempo

2. Las imágenes constan de distintas capas: ¿Qué quiere decir esto?

Si cada una de estas imágenes está compuesta por una serie de capas que contienen los archivos y configuraciones necesarios para que el contenedor funcione. Estas capas se apilan una encima de otra para formar la imagen final del contenedor haciendo que se puedan utilizar eficientemente los recursos y la administración de versiones de imágenes de contenedor.

## Tarea: docker build with

```
FROM ubuntu

ENV NUMWORKERS 2

RUN apt update
RUN apt install -y python3-pip
RUN pip install hypercorn quart

RUN mkdir -p /sil/build/users

EXPOSE 8000

RUN useradd sil
RUN chown -R sil /sil
USER sil

WORKDIR /sil

COPY user_rest.py ./

CMD hypercorn --bind 0.0.0.0:8000 --workers ${NUMWORKERS} user_rest:app
```

user

```
(si1p1) e462883@12-21-12-21:~/Downloads/src$ docker build --tag si1p1:1.0u .
[+] Building 3.0s (14/14) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 350B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                           0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.7s
=> [internal] load build context                          0.0s
=> => transferring context: 34B                         0.0s
=> [1/9] FROM docker.io/library/ubuntu@sha256:9b8dec3bf938bc80fbe758d856 0.0s
=> CACHED [2/9] RUN apt update                           0.0s
=> CACHED [3/9] RUN apt install -y python3-pip           0.0s
=> CACHED [4/9] RUN pip install hypercorn quart            0.0s
=> CACHED [5/9] RUN mkdir -p /si1/build/users             0.0s
=> [6/9] RUN useradd si1                                0.5s
=> [7/9] RUN chown -R si1 /si1                           0.5s
=> [8/9] WORKDIR /si1                                 0.2s
=> [9/9] COPY user_rest.py ./                           0.1s
=> exporting to image                                  0.8s
=> => exporting layers                               0.7s
=> => writing image sha256:a8e637eed480065458af9827330bc23e361c3fe1b2f2e 0.0s
=> => naming to docker.io/library/si1p1:1.0u           0.0s
```

## Pregunta: docker build with user

1. ¿Por qué en los últimos pasos no utiliza la caché?

En ese paso no se muestra la etiqueta CACHED, esto implica la creación de un nuevo usuario en el sistema y la información sobre este usuario no se encuentra en las capas de caché existente por lo que no pueden ser reutilizadas

```
Dockerfile user_lambda_dynamo.py user_rest.py
(si1p1) e462883@12-21-12-21:~/Downloads/src$ docker run --name si1p1 -e NUMWORKERS=5 -d -p 8080:8000 si1p1
536a4bb3ea6ff16e47b8f8a7ff4409e5c08158fe3112fcf33ab45e66662cbfbab
(si1p1) e462883@12-21-12-21:~/Downloads/src$
```

  

```
(si1p1) e462883@12-21-12-21:~/Downloads/src$ docker exec -ti si1p1 /bin/bash
root@536a4bb3ea6f:/si1#
```

```
root@536a4bb3ea6f:/si1# ps --forest -aef
UID      PID  PPID  C STIME TTY          TIME CMD
root      14      0  0 09:45 pts/0    00:00:00 /bin/bash
root      23     14  0 09:46 pts/0    00:00:00 \_ ps --forest -aef
root      1      0  0 09:45 ?        00:00:00 /bin/sh -c hypercorn --bind
root      7      1  0 09:45 ?        00:00:00 /usr/bin/python3 /usr/local/
root      8      7  0 09:45 ?        00:00:00 \_ /usr/bin/python3 -c from
root      9      7  0 09:45 ?        00:00:00 \_ /usr/bin/python3 -c from
root     10      7  0 09:45 ?        00:00:00 \_ /usr/bin/python3 -c from
root     11      7  0 09:45 ?        00:00:00 \_ /usr/bin/python3 -c from
root     12      7  0 09:45 ?        00:00:00 \_ /usr/bin/python3 -c from
root     13      7  0 09:45 ?        00:00:00 \_ /usr/bin/python3 -c from
root@536a4bb3ea6f:/si1#
```

## Tarea: docker run myimage

### Pregunta: docker run my image

1. ¿Cuántos subprocessos de hypercorn aparecen? ¿Por qué?

En la salida podemos observar subprocessos con el PID 7, 8, 9, 10, 11, 12, 13 en total 7, los cuales se encargan de manejar solicitudes y conexiones entrantes para la aplicación en ejecución.

2. ¿Qué hace la opción '-p' del comando docker run?

Se utiliza para mapear puertos entre el sistema anfitrión y el contenedor que se está iniciando, en este caso se mapea en el puerto 8080:8000, en nuestro caso cualquier solicitud que llegue al puerto 8080 de la máquina anfitrión será redirigida al puerto 8000 del contenedor.

3. Si deseamos ejecutar otra réplica (contenedor) del microservicio, ¿qué deberíamos cambiar en la sentencia docker run?

Por ejemplo: docker run --name si1p2 -e NUMWORKERS=5 -d -p 8081:8000 si1p1

Con este comando creamos una réplica pero en este caso mapeamos en el puerto 8081 del sistema host redirigiendo las solicitudes a nuestro puerto de contenedor 8000.

Esto es posible repetirlo tantas veces como se quiera para ejecutar múltiples réplicas.

## Tarea: docker run myimage on ./si1

```
(si1p1) e462883@12-21-12-21:~$ docker run --name si1p1-r -e NUMWORKERS=5 -d -p 8082:8000 -v ~/si1:/si1 si1p1  
c51eeefaf357655ee453d7896a23e736e664100e55838c9d1f33c248657a6616b
```

### Pregunta: docker run myimage on ./si1

1. ¿Qué mandato has ejecutado?

Este comando crea un contenedor nuevo llamado si1p1\_r y montará el directorio local en el directorio si1 del contenedor de esta manera cualquier dato o archivo en el directorio local estará disponible para la aplicación dentro del contenedor en el directorio si1

2. De cara a la realización de backups, ¿cuál crees que es la utilidad de montar volúmenes externos respecto de usar los internos de docker?

Permiten persistencia de datos más allá del ciclo de vida de un contenedor. Si un contenedor se elimina o se detiene, los datos en un volumen externo no se pierden. Facilita copias de seguridad y restauración, y además los volúmenes externos se pueden compartir entre varios contenedores, lo que permite que varios contenedores accedan a los mismos datos. Los volúmenes externos pueden estar respaldados por sistemas de almacenamiento en red lo que permite un mayor grado de escalabilidad y disponibilidad de datos.

Además los volúmenes externos facilitan el aislamiento de datos entre aplicaciones. Estos volúmenes también permiten la portabilidad de datos, se pueden mover y compartir volúmenes entre entornos de desarrollo, pruebas y producción de manera más sencilla.

## Tarea: docker registry

### Pregunta: docker registry

1. ¿Qué mandato has usado para descargar la imagen del registry?

```
docker run -d -p 5050:5000 --restart=always --name si1_registry registry:latest
```

```
(si1p1) e462883@12-21-12-21:~$ docker run -d -p 5050:5000 --restart=always --name si1_registry registry:latest
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry
96526aa774ef: Pull complete
cc37b24bb099: Pull complete
1d8a1aa97222: Pull complete
e3ff0af69d79: Pull complete
17443307a4fc: Pull complete
Digest: sha256:12a6ddd56d2de5611ff0d9735ac0ac1d1e44073c7d042477329e589c46867e4e
Status: Downloaded newer image for registry:latest
1d896a3c2255f0547cc09244cc7605d1e776a8746a67a15da25f7e28841f457a
```

2. ¿Cuál es el número de versión de dicha imagen?

```
(si1p1) e462883@12-21-12-21:~$ docker logs si1_registry
time="2023-10-11T10:30:41.714143815Z" level=warning msg="No HTTP secret provided - generated random secret. This may cause problems with uploads if multiple registries are behind a load-balancer. To provide a shared secret, fill in http.secret in the configuration file or set the REGISTRY_HTTP_SECRET environment variable." go.version=go1.20.8 instance.id=2ad2c4ba-8629-415b-aa61-34e95ecb7aae service=registry version=2.8.3
time="2023-10-11T10:30:41.714190702Z" level=info msg="redis not configured" go.version=go1.20.8 instance.id=2ad2c4ba-8629-415b-aa61-34e95ecb7aae service=registry version=2.8.3
time="2023-10-11T10:30:41.714279011Z" level=info msg="using inmemory blob descriptor cache" go.version=go1.20.8 instance.id=2ad2c4ba-8629-415b-aa61-34e95ecb7aae service=registry version=2.8.3
time="2023-10-11T10:30:41.714374199Z" level=info msg="listening on [::]:5000" go.version=go1.20.8 instance.id=2ad2c4ba-8629-415b-aa61-34e95ecb7aae service=registry version=2.8.3
time="2023-10-11T10:30:41.714420466Z" level=info msg="Starting upload purge in 36m0s" go.version=go1.20.8 instance.id=2ad2c4ba-8629-415b-aa61-34e95ecb7aae service=registry version=2.8.3
time="2023-10-11T10:30:41.714420466Z" level=info msg="Starting upload purge in 36m0s" go.version=go1.20.8 instance.id=2ad2c4ba-8629-415b-aa61-34e95ecb7aae service=registry version=2.8.3
```

La versión como se puede comprobar es 2.8.3

3. ¿Qué mandato has usado para ejecutar el contenedor del registry?

```
(si1p1) e462883@12-21-12-21:~$ docker start si1_registry
si1_registry
(si1p1) e462883@12-21-12-21:~$ docker exec -it si1_registry /bin/sh
#
```

4. ¿En qué puerto escucha por defecto el contenedor del registry?

El servicio escucha en el puerto 5050 en el sistema anfitrion, como lo especifica el mapeo de puertos -p 5050:5000 y se comunicara con el contendor en el puerto 5000.

## Tarea: docker tag push

### Pregunta: docker tag push

#### 1. ¿Qué ha sucedido?

El puerto 5000 no está disponible, y tras unas pruebas llegamos a la conclusión de la imposibilidad de este puerto para esta tarea. Informándonos descubrimos que generalmente se utiliza el puerto 5000 para exponer servicios dentro del contenedor, no para el etiquetado y empuje de imágenes

```
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
aece8493d397: Already exists
Digest: sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac8176c6e496e5c1f05f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker run -d -p 5050:5000 --restart=always --name si1_registry registry:latest
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry
96526aa774ef: Pull complete
cc37b24bb099: Pull complete
1d8a1aa97222: Pull complete
e3ff0af69d79: Pull complete
17443307a4fc: Pull complete
Digest: sha256:12a6ddd56d2de5611ff0d9735ac0ac1d1e44073c7d042477329e589c46867e4e
Status: Downloaded newer image for registry:latest
e75e3d52dd73aa55e6e15154cd55dafff74c9a5e0418ce610fc219eea9ae9b3c

(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker tag ubuntu:latest localhost:5050/ubuntu:latest

(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker push localhost:5050/ubuntu:latest
The push refers to repository [localhost:5050/ubuntu]
256d88da4185: Pushed
latest: digest: sha256:f29870aec43cb049f711f7b807626214c9a97e428f93dfcdf3ba23d2c51c2fa5 size: 529
```

Para ello accedimos al puerto 5050, y como podemos comprobar se creó el tag para dicha imagen con “localhost:5050/ubuntu” y se realizó correctamente el push.

## Tarea: app from local

### Pregunta: docker app from local

1. ¿Qué mandatos has usado?

```
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker build --tag si1p1_app .
[+] Building 0.2s (12/12) FINISHED
=> [internal] load build definition from Dockerfile                                     0.1s
=> => transferring dockerfile: 368B                                                 0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                 0.0s
=> [internal] load metadata for localhost:5000/ubuntu:latest                         0.0s
=> [1/7] FROM localhost:5000/ubuntu                                                 0.0s
=> [internal] load build context                                                 0.0s
=> => transferring context: 34B                                                 0.0s
=> CACHED [2/7] RUN apt update                                                 0.0s
=> CACHED [3/7] RUN apt install -y python3-pip                                 0.0s
=> CACHED [4/7] RUN pip install hypercorn quart                                0.0s
=> CACHED [5/7] RUN mkdir -p /si1/build/users                                0.0s
=> CACHED [6/7] WORKDIR /si1                                                 0.0s
=> CACHED [7/7] COPY user_rest.py ./                                         0.0s
=> exporting to image                                                 0.0s
=> => exporting layers                                                 0.0s
=> => writing image sha256:70dac451ee3d46d78c1e85c9f413a54972d30ead8837d 0.0s
=> => naming to docker.io/library/si1p1_app                                  0.0s
=> => transferring context: 34B                                                 0.0s
=> CACHED [2/7] RUN apt update                                                 0.0s
=> CACHED [3/7] RUN apt install -y python3-pip                                 0.0s
=> CACHED [4/7] RUN pip install hypercorn quart                                0.0s
=> CACHED [5/7] RUN mkdir -p /si1/build/users                                0.0s
=> CACHED [6/7] WORKDIR /si1                                                 0.0s
=> CACHED [7/7] COPY user_rest.py ./                                         0.0s
=> exporting to image                                                 0.0s
=> => exporting layers                                                 0.0s
=> => writing image sha256:70dac451ee3d46d78c1e85c9f413a54972d30ead8837d 0.0s
=> => naming to docker.io/library/si1p1_app                                  0.0s
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ vim Dockerfile
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker tag si1p1_app:latest localhost
:5050/ubuntu
5050/ubuntu      5050/ubuntu:latest
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker tag si1p1_app:latest localhost
:5050/ubuntu
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS              PORTS
NAMES
e75e3d52dd73   registry:latest  "/entrypoint.sh /etc..."   3 minutes ago    Up 3 minutes
0.0.0.0:5050->5000/tcp, :::5050->5000/tcp   si1_registry
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED             SIZE
si1p1_app       latest       70dac451ee3d  3 minutes ago   478MB
localhost:5050/ubuntu  latest       70dac451ee3d  3 minutes ago   478MB
ubuntu          latest       e4c58958181a  13 days ago    77.8MB
localhost:5000/ubuntu  latest       e4c58958181a  13 days ago    77.8MB
localhost:5050/ubuntu  <none>     e4c58958181a  13 days ago    77.8MB
registry         latest       0ae1560ca86f  2 weeks ago    25.4MB
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker tag si1p1_app:latest localhost
:5050/si1p1_app
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker push localhost:5050/si1p1_app
Using default tag: latest
The push refers to repository [localhost:5050/si1p1_app]
1496b13c41ba: Pushed
5f70bf18a086: Pushed
ee26a3caa91: Pushed
9a847b2b4d19: Pushed
bac8c6e75d68: Pushed
55a451137c7c: Pushed
256d88da4185: Mounted from ubuntu
latest: digest: sha256:c49f2f4aa328f8f1c5d022509c2b19e056e60fc3d7278b7a2140cbc8a
c2e54c1 size: 1785
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ docker run -d --name registry si1p1_a
pp
0bee450e5549d05870401c3fad6f58a158ff7c04bcf5380980cf37f5488522b9
(si1p1) e462883@12-15-12-15:~/p1-v1.0/src$ █
```

Construimos la imagen docker a partir del contexto actual donde nos encontramos , en este caso en la carpeta src.

creamos un tag para dicha imagen “localhost:5050/sip1\_app”

y realizamos un push de dicho tag

Para finalizar hacemos un run de dicha imagen bajo el nombre de registry

El dockerfile contiene lo siguiente:

```
FROM localhost:5050/ubuntu
#FROM ubuntu

ENV NUMWORKERS 2

RUN apt update
RUN apt install -y python3-pip
RUN pip install hypercorn quart

RUN mkdir -p /si1/build/users

EXPOSE 8000

RUN useradd si1
RUN chown -R si1 /si1
USER si1

WORKDIR /si1

COPY user_rest.py .

CMD hypercorn --bind 0.0.0.0:8000 --workers ${NUMWORKERS} user_rest:app
```

## Parte 2: REST API en cloud

### Tarea: AWS S3

```
e460502@6A-7-6-7:~/pi-v1.0$ awslocal s3 mb s3://si1p1
make_bucket: si1p1
e460502@6A-7-6-7:~/pi-v1.0$ awslocal s3 cp src/user_rest.py s3://si1p1/src/user_rest.py
upload: src/user_rest.py to s3://si1p1/src/user_rest.py
e460502@6A-7-6-7:~/pi-v1.0$ awslocal s3 ls s3://si1p1
      PRE src/
e460502@6A-7-6-7:~/pi-v1.0$ awslocal s3 ls s3://si1p1/src
      PRE src/
e460502@6A-7-6-7:~/pi-v1.0$ awslocal s3 ls s3://si1p1/src/
2023-10-19 18:00:54          1088 user_rest.py
e460502@6A-7-6-7:~/pi-v1.0$
```

### Pregunta: AWS S3

1. ¿Qué hacen los distintos subcomandos de awslocal s3?

- a. awslocal s3 mb s3://si1p1: crea un nuevo bucket en Amazon S3 con el nombre si1p1
- b. awslocal s3 cp src/user\_rest.py s3://si1p1/src/user\_rest.py: Este comando copia el archivo src/user\_rest.py desde tu sistema local al bucket
- c. awslocal s3 ls s3://si1p1: Este comando lista el contenido del bucket mostrara una lista de objetos y subdirectorios en ese bucket

- d. awslocal s3 ls s3://si1p1/src: este comando lista el contenido del directorio src dentro del bucket mostrara objetos y subdirectorios que se encuentren dentro de ese directorio
  - e. awslocal s3 ls s3://si1p1/src/: lista el contenido del directorio src en el bucket incluyendo su contenido. Mostrara los objetos que se encuentren directamente dentro de ese directorio.

2. ¿Qué comando podemos usar para descargar el archivo user\_rest.py desde el bucket que hemos creado, dándole otro nombre para evitar sobreescribirlo ?

```
awslocal s3 cp s3://sj1p1/src/user rest.py /ruta/de/destino/nuevo nombre.py
```

Obviamente cambiando el destino donde queremos descargarlo y el nombre del archivo.

## Tarea: Lambdas

```
(siip1) enrique@enrique-VirtualBox:~$ LAMBDA_NAME=user_lambda
(siip1) enrique@enrique-VirtualBox:~$ [[ -d build ]] || mkdir build ; \
cd src ; zip ..//build/${LAMBDA_NAME}.zip ${LAMBDA_NAME}.py; cd -
updating: user_lambda.py (deflated 46%)
/home/enrique
```

```
e462883@4-32-4-32:~/p2$ echo "create: newlambda"; \
awslocal lambda create-function \
--function-name ${LAMBDA_NAME} \
--runtime python3.9 \
--zip-file file:///build/${LAMBDA_NAME}.zip \
--handler ${LAMBDA_NAME}.handler \
--role arn:aws:iam::000000000000:role/si1 ; \
awslocal lambda create-function-url-config \
--function-name ${LAMBDA_NAME} \
--auth-type NONE ;
create: newlambda
{
    "FunctionName": "user_lambda",
    "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda",
    "Runtime": "python3.9",
    "Role": "arn:aws:iam::000000000000:role/si1",
    "Handler": "user_lambda.handler",
    "CodeSize": 799,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2023-10-19T12:06:51.993871+0000",
    "CodeSha256": "F/V6rC02SVH6TIo1dXs0emHnT6S/pfll8B9Znvascjs=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "6982ec35-3adb-45c5-9ff6-c5beae7a1154",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    },
    "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
    },
    "RuntimeVersionConfig": {
        "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:8eff65f6809a3ce81507fe733fe09b835899b99481ba22fd75b5a7338290ec1"
    }
}
{
    "FunctionUrl": "http://sbxh0vfj1cpnkjbvm0vgd0atl7alvful.lambda-url.us-east-1.localstack.cloud:4566",
    "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda",
    "AuthType": "NONE",
    "CreationTime": "2023-10-19T12:06:52.449264+0000"
}
```

```
e462883@4-32-4-32:~/p2$ awslocal lambda wait function-active-v2 --function-name ${LAMBDA_NAME}
e462883@4-32-4-32:~/p2$ awslocal lambda list-functions
{
    "Functions": [
        {
            "FunctionName": "user_lambda",
            "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda",
            "Runtime": "python3.9",
            "Role": "arn:aws:iam::000000000000:role/si1",
            "Handler": "user_lambda.handler",
            "CodeSize": 799,
            "Description": "",
            "Timeout": 3,
            "MemorySize": 128,
            "LastModified": "2023-10-19T12:06:51.993871+0000",
            "CodeSha256": "F/V6rC02SVH6TIo1dXs0emHnT6S/pfll8B9Znvascjs=",
            "Version": "$LATEST",
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "6d7c06c5-fe9b-4479-a2c0-e84282de05d5",
            "PackageType": "Zip",
            "Architectures": [
                "x86_64"
            ],
            "EphemeralStorage": {
                "Size": 512
            },
            "SnapStart": {
                "ApplyOn": "None",
                "OptimizationStatus": "Off"
            }
        }
    ]
}
e462883@4-32-4-32:~/p2$ curl -X POST \
$(awslocal lambda get-function-url-config --function-name ${LAMBDA_NAME} | \
jq -r '.FunctionUrl') \
-H 'Content-Type: application/json' \
-d '{"username": "pepe", "data": "10"}'
"f[test_object_key] placed into S3"e462883@4-32-4-32:~/p2$
```

## Pregunta: Lambdas

1. ¿Qué runtime podríamos usar si por ejemplo queremos implementar una función Lambda en JavaScript ?  
El entorno de ejecución Node.js.
2. ¿Cómo recibe la función lambda el usuario los datos que hemos enviado con el comando curl?  
La función Lambda recibe los datos como datos JSON.
3. Comprueba que ha funcionado descargando del bucket el fichero creado por la función. ¿Qué comandos has utilizado para ello?

```
awslocal s3 cp s3://si1p1/users/pepe ruta
```

```
e462883@4-32-4-32:~/p2$ awslocal s3 cp s3://si1p1/users/pepe pepe2  
download: s3://si1p1/users/pepe to ./pepe2  
e462883@4-32-4-32:~/p2$ ls  
build pepe2 src  
e462883@4-32-4-32:~/p2$
```

## Tarea: depuración lambdas

### Pregunta: depuración lambdas

1. ¿Describe a qué corresponde la salida de cada uno de los comandos anteriores?

```
e462883@4-32-4-32:~/p2$ docker logs localstack_main  
  
LocalStack version: 2.3.3.dev  
LocalStack Docker container id: 893ff8043346  
LocalStack build date: 2023-10-19  
LocalStack build git hash: e36975cf  
  
2023-10-19T12:03:01.924 INFO --- [-functhread6] hypercorn.error : Running on https://0.0.0.0:  
4566 (CTRL + C to quit)  
2023-10-19T12:03:01.924 INFO --- [-functhread6] hypercorn.error : Running on https://0.0.0.0:  
4566 (CTRL + C to quit)  
Ready.  
2023-10-19T12:06:51.994 INFO --- [ asgi_gw_0] localstack.request.aws : AWS lambda.CreateFunction =  
> 201  
2023-10-19T12:06:52.449 INFO --- [ asgi_gw_0] localstack.request.aws : AWS lambda.CreateFunctionUr  
lConfig => 201  
2023-10-19T12:09:24.938 INFO --- [ asgi_gw_0] localstack.request.aws : AWS lambda.GetFunction => 2  
00  
2023-10-19T12:09:33.389 INFO --- [ asgi_gw_1] localstack.request.aws : AWS lambda.ListFunctions =>  
200  
2023-10-19T12:09:40.989 INFO --- [ asgi_gw_1] localstack.request.aws : AWS lambda.GetFunctionUrlCo  
nfig => 200  
2023-10-19T12:09:41.192 INFO --- [ asgi_gw_1] l.u.container_networking : Determined main container n  
etwork: bridge  
2023-10-19T12:09:41.197 INFO --- [ asgi_gw_1] l.u.container_networking : Determined main container t  
arget IP: 172.17.0.2  
2023-10-19T12:09:41.922 INFO --- [ asgi_gw_2] localstack.request.aws : AWS s3.CreateBucket => 200  
2023-10-19T12:09:41.927 INFO --- [ asgi_gw_2] localstack.request.http : POST /_localstack_lambda/63  
c4f90b4fdd68a57e6d4266ebb2a703/status/63c4f90b4fdd68a57e6d4266ebb2a703/ready => 202  
2023-10-19T12:09:42.029 INFO --- [ asgi_gw_2] localstack.request.aws : AWS s3.PutObject => 200  
2023-10-19T12:09:42.034 INFO --- [ asgi_gw_2] localstack.request.http : POST /_localstack_lambda/63  
c4f90b4fdd68a57e6d4266ebb2a703/invocations/5b420662-9f32-47c3-866f-5ab8e061073b/logs => 202  
2023-10-19T12:09:42.036 INFO --- [ asgi_gw_2] localstack.request.http : POST /_localstack_lambda/63  
c4f90b4fdd68a57e6d4266ebb2a703/invocations/5b420662-9f32-47c3-866f-5ab8e061073b/response => 202  
2023-10-19T12:09:42.039 INFO --- [ asgi_gw_1] localstack.request.http : POST / => 200  
2023-10-19T12:11:02.457 INFO --- [ asgi_gw_2] localstack.request.aws : AWS s3.HeadObject => 200  
2023-10-19T12:11:02.466 INFO --- [ asgi_gw_0] localstack.request.aws : AWS s3.GetObject => 200
```

Esto coincide con la salida del local start.

```
e462883@4-32-4-32:~/p2$ awslocal logs describe-log-groups
{
    "logGroups": [
        {
            "logGroupName": "/aws/lambda/user_lambda",
            "creationTime": 1697717382091,
            "metricFilterCount": 0,
            "arn": "arn:aws:logs:us-east-1:000000000000:log-group:/aws/lambda/user_lambda:*",
            "storedBytes": 1567
        }
    ]
}
```

Lista los grupos de registro disponibles en la configuración de AWS Local. Aparecen el nombre, el tiempo de creación...

```
e462883@4-32-4-32:~/p2$ awslocal logs describe-log-streams --log-group-name /aws/lambda/${LAMBDA_NAME}
{
    "logStreams": [
        {
            "logStreamName": "2023/10/19[$LATEST]63c4f90b4fdd68a57e6d4266ebb2a703",
            "creationTime": 1697717382098,
            "firstEventTimestamp": 1697717382039,
            "lastEventTimestamp": 1697717382096,
            "lastIngestionTime": 1697717382105,
            "uploadSequenceToken": "1",
            "arn": "arn:aws:logs:us-east-1:000000000000:log-group:/aws/lambda/user_lambda:log-stream:2023
/10/19[$LATEST]63c4f90b4fdd68a57e6d4266ebb2a703",
            "storedBytes": 1567
        }
    ]
}
```

Información sobre los flujos de registro del grupo /aws/lambda/\${LAMBDA\_NAME}. Aparecen el nombre, el tiempo de creación, la primera marca de tiempo del evento, la última marca de tiempo del evento, el tiempo de la última ingestión, el token de secuencia, el arn y los bytes guardados.

```
e462883@4-32-4-32:~/p2$ awslocal logs describe-log-streams --log-group-name /aws/lambda/${LAMBDA_NAME} | jq '.logStreams[].firstEventTimestamp' | sort -n
1697717382039
```

La salida muestra el tiempo del primer evento en los flujos de registro del grupo que por el comando “sort -n” debería aparecer en orden de menor a mayor, pero solo existe una.

```
e462883@4-32-4-32:~/p2$ LLOGSTREAMS=$(awslocal logs describe-log-streams \
--log-group-name /aws/lambda/${LAMBDA_NAME} | jq -r '.logStreams[].logStreamName')
```

```

e462883@4-32-4-32:~/p2$ for LOGSTREAM in ${LLOGSTREAMS} ; do
awslocal logs get-log-events \
--log-group-name "/aws/lambda/${LAMBDA_NAME}" \
--log-stream-name "${LOGSTREAM}" ; done |
jq '.events[].message'
"START RequestId: 5b420662-9f32-47c3-866f-5ab8e061073b Version: $LATEST"
"[INFO]\t2023-10-19T12:09:41.928Z\t5b420662-9f32-47c3-866f-5ab8e061073b\tEscribiendo en S3"
"Received event: {"
"  \"version\": \"2.0\","
"  \"routeKey\": \"$default\","
"  \"rawPath\": \"/\","
"  \"rawQueryString\": \"\","
"  \"headers\": {"
"    \"host\": \"sbxh0vfjc1pnkjbvm0vgd0atl7alvful.lambda-url.us-east-1.localhost.localstack.cloud:4566\","
"    \"user-agent\": \"curl/7.87.0\","
"    \"accept\": \"/*/*\","
"    \"content-type\": \"application/json\","
"    \"content-length\": \"34\","
"    \"x-amzn-tls-cipher-suite\": \"ECDHE-RSA-AES128-GCM-SHA256\","
"    \"x-amzn-tls-version\": \"TLSV1.2\","
"    \"x-forwarded-proto\": \"http\","
"    \"x-forwarded-for\": \"\","
"    \"x-forwarded-port\": \"4566\""
"  },"
"  \"queryStringParameters\": {},"
"  \"requestContext\": {"
"    \"accountId\": \"anonymous\","
"    \"apiId\": \"sbxh0vfjc1pnkjbvm0vgd0atl7alvful\","
"    \"domainName\": \"sbxh0vfjc1pnkjbvm0vgd0atl7alvful.lambda-url.us-east-1.localhost.localstack.cloud:4566\","
"    \"domainPrefix\": \"sbxh0vfjc1pnkjbvm0vgd0atl7alvful\","
"    \"http\": {"
"      \"method\": \"POST\","
"      \"path\": \"/\","
"      \"protocol\": \"HTTP/1.1\","
"      \"sourceIp\": \"\","
"      \"userAgent\": \"curl/7.87.0\""
"    },"
"    \"requestId\": \"7903c4ed-9c08-47b0-988e-b6ba9f55e1ab\","
"    \"routeKey\": \"$default\","
"    \"stage\": \"$default\","
"    \"time\": \"19/Oct/2023:12:09:41 +0000\","
"    \"timeEpoch\": 1697717381176"
"  },"
"  \"body\": \"{\\"username\\\": \"pepe\", \\"data\\\": \\"10\\\"}\","
"  \"isBase64Encoded\": false"
"}"
"END RequestId: 5b420662-9f32-47c3-866f-5ab8e061073b"
"REPORT RequestId: 5b420662-9f32-47c3-866f-5ab8e061073b\tDuration: 104.03 ms\tBilled Duration: 105 ms\tMemory Size: 128 MB\tMax Memory Used: 128 MB"

```

En estos comandos: en el primero se asigna la lista de flujos de registro al valor de la variable LLOGSTREAM. Luego, se inicia un bucle que recorre cada uno de los flujos de registro en el grupo y ejecuta comandos para obtener los eventos de registro de cada flujo. Finalmente, se utiliza jq para filtrar y mostrar el contenido de los mensajes en los eventos de registro.

## Tarea: Borrado de funciones

```

e462883@4-32-4-32:~/p2$ echo "clean: dellambda" ;
clean: dellambda
e462883@4-32-4-32:~/p2$ awslocal lambda delete-function-url-config \
--function-name user_lambda ;
e462883@4-32-4-32:~/p2$ awslocal lambda delete-function --function-name user_lambda
e462883@4-32-4-32:~/p2$ echo "clean: md zip" \
awslocal s3 rb --force s3://si1p1 ; \
rm build/user_lambda.* \
clean: md zip awslocal s3 rb --force s3://si1p1

```

## Pregunta: Borrado de funciones

1. ¿Qué ocurre si en lugar de borrar las funciones y archivos en S3, simplemente reiniciamos localStack? ¿Cómo podríamos evitarlo?

Los recursos y configuraciones creados previamente en localstack seguirán estando disponibles después del reinicio. Se debe realizar una limpieza antes de reiniciar localstack, igual que hemos realizado con los comandos realizados.

## 5. API Gateway

### Tarea: Creación del API

```
(si1p1) e462883@2-7-2-7:~/p_2$ export REGION="us-east-1" ; export API_NAME="user_lambda"
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway create-rest-api \
--region ${REGION} \
--name ${API_NAME}
{
    "id": "g2qmouu7in",
    "name": "user_lambda",
    "createdDate": 1697705268.0,
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "EDGE"
        ]
    },
    "disableExecuteApiEndpoint": false
}

(si1p1) e462883@2-7-2-7:~/p_2$ [ $? == 0 ] || echo "Failed: AWS / apigateway / create-rest-api"
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-rest-apis
{
    "items": [
        {
            "id": "g2qmouu7in",
            "name": "user_lambda",
            "createdDate": 1697705268.0,
            "apiKeySource": "HEADER",
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "disableExecuteApiEndpoint": false
        }
    ]
}
```

### Pregunta: Creación del API

1. ¿Qué ID se ha asignado al API ?

Introduciendo el comando “awslocal apigateway get-rest-apis” podemos comprobar que el id asignado es g2qmouu7in

2. ¿Qué ocurre si volvemos a crear otro API con el mismo nombre ?

Se obtiene la creación de otra API con el mismo nombre pero distinto ID. Esto sucede porque estamos trabajando en entorno local, ya que si esto se lleva a cabo en un entorno de producción en la nube, encontraríamos restricciones para realizar dicha operación.

```
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-rest-apis
{
    "items": [
        {
            "id": "g2qmouu7in",
            "name": "user_lambda",
            "createdDate": 1697705268.0,
            "apiKeySource": "HEADER",
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "disableExecuteApiEndpoint": false
        },
        {
            "id": "n8d9jsi6g8",
            "name": "user_lambda",
            "createdDate": 1697705774.0,
            "apiKeySource": "HEADER",
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "disableExecuteApiEndpoint": false
        }
    ]
}
```

### Tarea: Creación de las rutas

```

(si1p1) e462883@2-7-2-7:~/p_2$ API_ID=$(awslocal apigateway get-rest-apis \
--query "items[?name=='${API_NAME}'].id" \
--output text --region ${REGION})
(si1p1) e462883@2-7-2-7:~/p_2$ PATH_PART="user" ; PARENT_PATH="/"
(si1p1) e462883@2-7-2-7:~/p_2$ PARENT_RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${PARENT_PATH}'].id" \
--output text --region ${REGION})
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway create-resource \
--region ${REGION} \
--rest-api-id ${API_ID} \
--parent-id ${PARENT_RESOURCE_ID} \
--path-part "${PATH_PART}"
{
    "id": "ttgknw4fkz",
    "parentId": "m4496av7cr",
    "pathPart": "user",
    "path": "/user"
}
(si1p1) e462883@2-7-2-7:~/p_2$ PATH_PART="{username}" ; PARENT_PATH="/user"
(si1p1) e462883@2-7-2-7:~/p_2$ PARENT_RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${PARENT_PATH}'].id" \
--output text --region ${REGION})
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway create-resource \
--region ${REGION} \
--rest-api-id ${API_ID} \
--parent-id ${PARENT_RESOURCE_ID} \
--path-part "${PATH_PART}"
{
    "id": "8q0syw6qwn",
    "parentId": "ttgknw4fkz",
    "pathPart": "{username}",
    "path": "/user/{username}"
}
(si1p1) e462883@2-7-2-7:~/p_2$ [ $? == 0 ] || echo "Failed: AWS / apigateway / create-resource ${PATH_PART}"

(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
    "items": [
        {
            "id": "m4496av7cr",
            "path": "/"
        },
        {
            "id": "ttgknw4fkz",
            "parentId": "m4496av7cr",
            "pathPart": "user",
            "path": "/user"
        },
        {
            "id": "8q0syw6qwn",
            "parentId": "ttgknw4fkz",
            "pathPart": "{username}",
            "path": "/user/{username}"
        }
    ]
}
(si1p1) e462883@2-7-2-7:~/p_2$ █

```

## Pregunta: Creación de las rutas

1. ¿Qué ID se ha asignado al recurso con la ruta /user/{username} ?

8q0syw6qwn

2. ¿Qué ocurre si creamos también la ruta /user/{name} ?

Pues se generaría una nueva ruta, y en este caso, ambas rutas (/user/{username} y /user/{name}) podrían existir bajo el mismo recurso padre y ser gestionado por la API

3. En el caso de que se pueda crear a la vez la ruta /user/{name} y /user/{username}, ¿Tiene sentido?

Tendría sentido si se quisieran implementar distintas funcionalidades para los campos de username y de name, si tendría sentido.

## Tarea: Creación de métodos

```
(si1p1) e462883@2-7-2-7:~/p_2$ PARAM=username ; METHOD_PATH="/user/{username}" ; HTTP_METHOD=POST
(si1p1) e462883@2-7-2-7:~/p_2$ RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${METHOD_PATH}'].id" \
--output text --region ${REGION})
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway put-method \
--region ${REGION} \
--rest-api-id ${API_ID} \
--resource-id ${RESOURCE_ID} \
--http-method ${HTTP_METHOD} \
--request-parameters "method.request.path.${PARAM}=true" \
--authorization-type "NONE"
{
    "httpMethod": "POST",
    "authorizationType": "NONE",
    "apiKeyRequired": false,
    "requestParameters": {
        "method.request.path.username": true
    }
}
(si1p1) e462883@2-7-2-7:~/p_2$ [ $? == 0 ] || echo "Failed: AWS / apigateway / put-method ${METHOD_PATH}"
{
  "items": [
    {
      "id": "m4496av7cr",
      "path": "/"
    },
    {
      "id": "ttgknw4fkz",
      "parentId": "m4496av7cr",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "8q0syw6qwn",
      "parentId": "ttgknw4fkz",
      "pathPart": "{username}",
      "path": "/user/{username}",
      "resourceMethods": {
        "POST": {
          "httpMethod": "POST",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        }
      }
    }
  ]
}
```

## Pregunta: Creación de métodos

1. Crea también el método GET asociado a la misma ruta, ¿Qué instrucciones has ejecutado para crearlo?

```
(s11p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway put-method \  
--region ${REGION} \  
--rest-api-id ${API_ID} \  
--resource-id ${RESOURCE_ID} \  
--http-method GET \  
--request-parameters "method.request.path.${PARAM}=true" \  
--authorization-type "NONE"  
{  
    "httpMethod": "GET",  
    "authorizationType": "NONE",  
    "apiKeyRequired": false,  
    "requestParameters": {  
        "method.request.path.username": true  
    }  
}
```

Hemos utilizado un comando similar al que utilizamos para crear el método post. Se utiliza el comando “awslocal apigateway put-method” para crear un método HTTP GET en una ruta específica de la API Gateway local. El método se configura para esperar el parámetro “username” (con la flag --request-parameters “method.request.path.\${PARAM}=true”) en las solicitudes entrantes y no requiere autenticación (--authorization-type “NONE”). También se agregan otras flags para indicar el ID de la API a la que agregamos el método, el ID del recurso al que se asocia el método y el método HTTP que queremos crear.

## 2. ¿Cómo podemos ver los métodos asociados a las rutas del API?

```
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "m4496av7cr",
      "path": "/"
    },
    {
      "id": "ttgknw4fkz",
      "parentId": "m4496av7cr",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "8q0syw6qwn",
      "parentId": "ttgknw4fkz",
      "pathPart": "{username}",
      "path": "/user/{username}",
      "resourceMethods": {
        "POST": {
          "httpMethod": "POST",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        },
        "GET": {
          "httpMethod": "GET",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        }
      }
    }
  ]
}
```

Con este comando (awslocal apigateway get-resources –rest-api-id \${API\_ID}) conseguimos ver los métodos asociados tras el parámetro “resourceMetods”: en el que podemos apreciar tanto el POST como el GET que hemos incluido anteriormente.

## Tarea: Integración

```
(si1p1) e462883@2-7-2-7:~/p_2$ HTTP_METHOD=POST ; LAMBDA_NAME=user_lambda
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway put-integration \
--region ${REGION} \
--rest-api-id ${API_ID} \
--resource-id ${RESOURCE_ID} \
--http-method ${HTTP_METHOD} \
--type AWS_PROXY \
--integration-http-method POST \
--uri arn:aws:apigateway:${REGION}:lambda:path/2015-03-
31/functions/${LAMBDA_NAME}/invocations \
--passthrough-behavior WHEN_NO_MATCH
{
    "type": "AWS_PROXY",
    "httpMethod": "POST",
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "8q0syw6qwn",
    "cacheKeyParameters": []
}
bash: 31/functions/user_lambda/invocations: No existe el archivo o el directorio
(si1p1) e462883@2-7-2-7:~/p_2$ [ $? == 0 ] || echo "Failed: AWS / apigateway / put-integration"
Failed: AWS / apigateway / put-integration
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
    "items": [
        {
            "id": "m4496av7cr",
            "path": "/"
        },
        {
            "id": "ttgknw4fkz",
            "parentId": "m4496av7cr",
            "pathPart": "user",
            "path": "/user"
        },
        {
            "id": "8q0syw6qwn",
            "parentId": "ttgknw4fkz",
            "pathPart": "{username}",
            "path": "/user/{username}",
            "resourceMethods": {
                "POST": {
                    "httpMethod": "POST",
                    "authorizationType": "NONE",
                    "apiKeyRequired": false,
                    "requestParameters": {
                        "method.request.path.username": true
                    },
                    "methodResponses": {},
                    "methodIntegration": {
                        "type": "AWS_PROXY",
                        "httpMethod": "POST",
                        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-",
                        "passthroughBehavior": "WHEN_NO_MATCH",
                        "timeoutInMillis": 29000,
                        "cacheNamespace": "8q0syw6qwn",
                        "cacheKeyParameters": []
                    }
                },
                "GET": {
                    "httpMethod": "GET",
                    "authorizationType": "NONE",
                    "apiKeyRequired": false,
                    "requestParameters": {
                        "method.request.path.username": true
                    },
                    "methodResponses": {}
                }
            }
        }
    ]
}
```

## Pregunta: Integración

1. Integra también el método GET asociado a la misma ruta con la misma función lambda, ¿qué instrucciones has ejecutado para crearlo?

```
(si1p1) e462883@2-7-2-7:~/p_2$ HTTP_METHOD=GET
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway put-integration \
--region ${REGION} \
--rest-api-id ${API_ID} \
--resource-id ${RESOURCE_ID} \
--http-method ${HTTP_METHOD} \
--type AWS_PROXY \
--integration-http-method GET \
--uri arn:aws:apigateway:${REGION}:lambda:path/2015-03-31/functions/${LAMBDA_NAME}/invocations \
--passthrough-behavior WHEN_NO_MATCH
{
    "type": "AWS_PROXY",
    "httpMethod": "GET",
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/user_lambda/invocations",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "8q0syw6qwn",
    "cacheKeyParameters": []
}
(si1p1) e462883@2-7-2-7:~/p_2$ [ $? == 0 ] || echo "Failed: AWS / apigateway / put-integration"
```

Las instrucciones que hemos utilizado son las mismas que se han utilizado para la integración de POST. Los cambios realizados son el cambio de la variable `HTTP_METHOD` a `GET` y la flag “`--integration-http-method GET`”. El resto se mantiene igual debido a que utilizamos la misma función lambda y no queremos una configuración diferente.

2. ¿Cómo podemos ver si se ha integrado cada método del API?

```
, "GET": {
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false,
    "requestParameters": {
        "method.request.path.username": true
    },
    "methodResponses": {},
    "methodIntegration": {
        "type": "AWS_PROXY",
        "httpMethod": "GET",
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/user_lambda/invocations",
        "passthroughBehavior": "WHEN_NO_MATCH",
        "timeoutInMillis": 29000,
        "cacheNamespace": "8q0syw6qwn",
        "cacheKeyParameters": []
    }
}
```

Con este comando (`awslocal apigateway get-resources --rest-api-id ${API_ID}`) podemos observar que en el apartado de GET, aparece “`methodIntegration`” con toda la información correcta de la integración.

3. Con ayuda de la documentación, por ejemplo ejecutando “`awslocal apigateway help`”, borra la integración del método GET, ¿qué instrucciones has ejecutado para lograrlo?

```
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway delete-integration \
--region ${REGION} \
--rest-api-id ${API_ID} \
--resource-id ${RESOURCE_ID} \
--http-method ${HTTP_METHOD}

(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "m4496av7cr",
      "path": "/"
    },
    {
      "id": "ttgknw4fkz",
      "parentId": "m4496av7cr",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "8q0syw6qwn",
      "parentId": "ttgknw4fkz",
      "pathPart": "{username}",
      "path": "/user/{username}",
      "resourceMethods": {
        "POST": {
          "httpMethod": "POST",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {},
          "methodIntegration": {
            "type": "AWS_PROXY",
            "httpMethod": "POST",
            "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-",
            "passthroughBehavior": "WHEN_NO_MATCH",
            "timeoutInMillis": 29000,
            "cacheNamespace": "8q0syw6qwn",
            "cacheKeyParameters": []
          }
        },
        "GET": {
          "httpMethod": "GET",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        }
      }
    }
  ]
}
```

Con el comando (awslocal apigateway delete-integration), pasando los parámetros que hemos pasado al hacer la integración del método, conseguimos eliminarlo. Tras ejecutar ese comando, volvemos a ejecutar (awslocal apigateway get-resources –rest-api-id \${API\_ID}) para comprobar que se ha borrado correctamente.

## Tarea: Despliegue

```
(si1p1) e462883@2-7-2-7:~/p_2$ STAGE=dev
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway create-deployment \
--region ${REGION} \
--rest-api-id ${API_ID} \
--stage-name ${STAGE}
{
    "id": "9vofxlzuwh",
    "createdDate": 1697708987.0
}
(si1p1) e462883@2-7-2-7:~/p_2$ [ $? == 0 ] || echo " Failed: AWS / apigateway / create-deployment"
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-deployments --rest-api-id ${API_ID}
{
    "items": [
        {
            "id": "9vofxlzuwh",
            "createdDate": 1697708987.0
        }
    ]
}
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-stages --rest-api-id ${API_ID}
{
    "item": [
        {
            "deploymentId": "9vofxlzuwh",
            "stageName": "dev",
            "cacheClusterEnabled": false,
            "cacheClusterStatus": "NOT_AVAILABLE",
            "methodSettings": {},
            "tracingEnabled": false
        }
    ]
}
```

### Pregunta: Despliegue

1. ¿Qué ocurre con la etapa “dev” (valor de STAGE) si repetimos el despliegue?

Si repetimos el despliegue, la etapa “dev” se actualizará con los cambios que hayamos realizado de la definición de la API.

2. Con ayuda de la documentación, por ejemplo ejecutando “awslocal apigateway help”, borra el despliegue que ya no está asociado a la etapa “dev”. ¿Qué instrucciones has ejecutado para lograrlo?

```
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway get-deployments --rest-api-id ${API_ID}
{
    "items": [
        {
            "id": "9vofxlzuwh",
            "createdDate": 1697708987.0
        },
        {
            "id": "94p3ycbpmk",
            "createdDate": 1697709112.0
        }
    ]
}
```

Primero con el comando `awslocal apigateway (get-deployments –rest-api-id ${API_ID})` detectamos el deployment a eliminar.

```
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal apigateway delete-deployment --region ${REGION} --rest-api-id \
${API_ID} --deployment-id 9vofxlzuwh
(si1p1) e462883@2-7-2-7:~/p_2$
```

Tras esto, con el id del deployment que hemos obtenido anteriormente utilizamos el comando (“awslocal apigateway delete-deployment –region \${REGION} –rest-api-id \${API\_ID} –deployment-id 9vofxlzuwh” siendo este último la id encontrada)

### Tarea: Pruebas del API

```
(si1p1) e462883@2-7-2-7:~/p_2$ STAGE=dev
(si1p1) e462883@2-7-2-7:~/p_2$ ENDPOINT=http://localhost:4566/restapis/${API_ID}/${STAGE}/_user_reques
t_user
(si1p1) e462883@2-7-2-7:~/p_2$ curl -X POST ${ENDPOINT}/pepe \
-H 'Content-Type: application/json' \
-d '{"username": "tete", "data": "100"}'
>{"Type": "User", "message": "Error invoking integration for API Gateway ID 'g2qmouu7in': list index ou
t of range", "__type": "InvalidRequest"}(si1p1) e462883@2-7-2-7:~/p_2$
```

### Pregunta: Pruebas del API

1. Usando los comandos de S3, lista el contenido del bucket y descarga el archivo que se ha creado para comprobar su contenido. ¿Qué ha ocurrido? ¿Ha cambiado el valor del fichero /users/pepe o se ha creado un archivo distinto?

```
(si1p1) e462883@2-7-2-7:~/p_2$ awslocal s3 cp s3://si1p1/users/pepe -
{"username": "pepe", "data": "10"}(si1p1) e462883@2-7-2-7:~/p_2$ awslocal s3 cp s3://si1p1/users/pepe
pepe
download: s3://si1p1/users/pepe to ./pepe
(si1p1) e462883@2-7-2-7:~/p_2$ cat pepe
{"username": "pepe", "data": "10"}(si1p1) e462883@2-7-2-7:~/p_2$ awslocal s3 cp s3://si1p1/users/pepe
-
{"username": "pepe", "data": "10"}(si1p1) e462883@2-7-2-7:~/p_2$
```

Hemos descargado el archivo pepe desde el bucket de aws al sistema local y al examinar el contenido de ambos archivos hemos visto que son idénticos.

### Tarea: Mejora del API

```
e460502@4-31-4-31:~/p2$ LAMBDA_NAME=user_lambda_param

e460502@4-31-4-31:~/p2$ [[ -d build ]] || mkdir build ; cd src ; zip ..../build/${LAMBDA_NAME}.zip ${LAMBDA_NAME}.py; cd -
adding: user_lambda_param.py (deflated 46%)
/home/alumnos/e460502/p2
```

```
e460502@4-31-4-31:~/p2$ echo "create: newlambda"; \
awslocal lambda create-function \
--function-name ${LAMBDA_NAME} \
--runtime python3.9 \
--zip-file file:///build/${LAMBDA_NAME}.zip \
--handler ${LAMBDA_NAME}.handler \
--role arn:aws:iam::000000000000:role/sii ; \
awslocal lambda create-function-url-config \
--function-name ${LAMBDA_NAME} \
--auth-type NONE ;
create: newlambda
{
  "FunctionName": "user_lambda_param",
  "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda_param",
  "Runtime": "python3.9",
  "Role": "arn:aws:iam::000000000000:role/sii",
  "Handler": "user_lambda_param.handler",
  "CodeSize": 811,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2023-10-19T13:14:44.669176+0000",
  "CodeSha256": "VXm/5pEVgZsIXit/AlMSfUYUMHAGKbpojymHVxx5uAo=",
  "Version": "$LATEST",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "802ec97a-2a74-446c-b619-dbf167fa317c",
  "State": "Pending",
  "StateReason": "The Function is being created.",
  "StateReasonCode": "Creating",
  "PackageType": "Zip",
  "Architectures": [
    "x86_64"
  ],
  "EphemeralStorage": {
    "Size": 512
  },
  "Snapstart": {
    "ApplyOn": "None",
    "OptimizationStatus": "Off"
  },
  "RuntimeVersionConfig": {
    "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:8eff65f6809a3ce81507fe733fe09b835899b99481ba22fd75b5a7338290ec1"
  }
}
{
  "FunctionUrl": "http://2fe9akea9ji8qweq4fygujg62332kr45.lambda-url.us-east-1.localhost.localstack.cloud:4566",
  "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda_param",
  "AuthType": "NONE",
  "CreationTime": "2023-10-19T13:14:45.082198+0000"
}
```

```
e460502@4-31-4-31:~/p2$ awslocal lambda wait function-active-v2 --function-name ${LAMBDA_NAME}
e460502@4-31-4-31:~/p2$ awslocal lambda list-functions
```

```
{
  "Functions": [
    {
      "FunctionName": "user_lambda_param",
      "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda_param",
      "Runtime": "python3.9",
      "Role": "arn:aws:iam::000000000000:role/sii",
      "Handler": "user_lambda_param.handler",
      "CodeSize": 811,
      "Description": "",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2023-10-19T13:14:44.669176+0000",
      "CodeSha256": "VXm/5pEVgZsIXit/AlMSfUYUMHAGKbpojymHVxx5uAo=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "81088035-8dd2-4595-b22b-1dbed455caa7",
      "PackageType": "Zip",
      "Architectures": [
        "x86_64"
      ],
      "EphemeralStorage": {
        "Size": 512
      },
      "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
      }
    }
  ]
}
```

```
e460502@4-31-4-31:~/p2$ curl -X POST \
$(awslocal lambda get-function-url-config --function-name ${LAMBDA_NAME} | \
jq -r .FunctionUrl) \
-H 'Content-Type: application/json' \
-d '{"username": "pepe", "data": "10"}'
"users/pepe placed into docker logs localstack_main/docker logs localstack_main"

LocalStack version: 2.3.3-dev
LocalStack Docker container id: 5fbcc786d845
LocalStack build date: 2023-10-19
LocalStack build git hash: e36975cf

2023-10-19T13:12:21.107 INFO --- [functhread6] hypercorn.error : Running on https://0.0.0.0:4566 (CTRL + C to quit)
2023-10-19T13:12:21.107 INFO --- [-[functhread6]] hypercorn.error : Running on https://0.0.0.0:4566 (CTRL + C to quit)
Ready
2023-10-19T13:13:06.664 INFO --- [ asgi_gw_0 ] localstack.request.aws : AWS s3.CreateBucket => 200
2023-10-19T13:14:44.071 INFO --- [ asgi_gw_0 ] localstack.request.aws : AWS Lambda.CreateFunction => 201
2023-10-19T13:14:45.082 INFO --- [ asgi_gw_0 ] localstack.request.aws : AWS Lambda.CreateFunctionUrlConfig => 201
2023-10-19T13:15:04.129 INFO --- [ asgi_gw_0 ] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-19T13:15:12.129 INFO --- [ asgi_gw_0 ] localstack.request.aws : AWS Lambda.ListFunctions => 200
2023-10-19T13:15:21.362 INFO --- [ asgi_gw_0 ] localstack.request.aws : AWS Lambda.GetFunctionUrlConfig => 200
2023-10-19T13:15:21.524 INFO --- [ asgi_gw_0 ] localstack.request.aws : Determined main container network: bridge
2023-10-19T13:15:21.524 INFO --- [ asgi_gw_0 ] localstack.request.aws : Determined main container IP: 172.17.0.2
2023-10-19T13:15:22.497 INFO --- [ asgi_gw_2 ] localstack.request.aws : AWS s3.CreateBucket => 200
2023-10-19T13:15:22.492 INFO --- [ asgi_gw_2 ] localstack.request.http : POST /localstack_lambda/94074a888d843931243da8553a3b52da/status => 202
2023-10-19T13:15:22.598 INFO --- [ asgi_gw_2 ] localstack.request.aws : AWS s3.PutObject => 200
2023-10-19T13:15:22.599 INFO --- [ asgi_gw_2 ] localstack.request.http : POST /localstack_lambda/94074a888d843931243da8553a3b52da/invocations/04b3ca4e-0045-49b4-9915-cfdcc4d9ff98/logs => 202
2023-10-19T13:15:22.599 INFO --- [ asgi_gw_2 ] localstack.request.http : POST /localstack_lambda/94074a888d843931243da8553a3b52da/invocations/04b3ca4e-0045-49b4-9915-cfdcc4d9ff98/response => 202
2023-10-19T13:15:22.599 INFO --- [ asgi_gw_0 ] localstack.request.http : POST / => 200
```

**DEPURAR**

```

440502@4-31-4-31: ~ $ awslocal logs describe-log-groups
{
  "logGroups": [
    {
      "logGroupName": "/aws/Lambda/user_lambda_param",
      "creationTime": 1697721322656,
      "metricFilterCount": 0,
      "arn": "arn:aws:logs:us-east-1:000000000000:log-group:/aws/lambda/user_lambda_param:*",
      "storedBytes": 1665
    }
  ]
}
440502@4-31-4-31: ~ $ awslocal logs describe-log-streams --log-group-name /aws/lambda/S(LAMBDA_NAME)
{
  "logStreams": [
    {
      "logStreamName": "2023/10/19/[SLATEST]94074a88bd843931243da8553a3b52da",
      "creationTime": 1697721322663,
      "firstEventTimestamp": 1697721322596,
      "lastEventTimestamp": 1697721322661,
      "lastLogEvent": 1697721322678,
      "uploadSequenceToken": "1",
      "arn": "arn:aws:logs:us-east-1:000000000000:log-group:/aws/lambda/user_lambda_param:log-stream:2023/10/19/[SLATEST]94074a88bd843931243da8553a3b52da",
      "storedBytes": 1665
    }
  ]
}
440502@4-31-4-31: ~ $ ps aux | grep awslocal logs describe-log-streams --log-group-name /aws/lambda/S(LAMBDA_NAME) | jq .logStreamName[] | sort -n
1697721322596
1697721322598
440502@4-31-4-31: ~ $ ll /tmp/streams | awslocal logs describe-log-streams --log-group-name /aws/lambda/S(LAMBDA_NAME) | jq .logStreams[].logStreamName

```

## INTEGRAR

```
e460502@4-31-4-31:~/p2$ HTTP_METHOD=POST ; LAMBDA_NAME=user_lambda_param
awslocal apigateway put-integration \
--region ${REGION} \
--rest-api-id ${API_ID} \
--resource-id ${RESOURCE_ID} \
--http-method ${HTTP_METHOD} \
--type AWS_PROXY \
--integration-http-method POST \
--uri arn:aws:apigateway:${REGION}:lambda:path/2015-03-31/functions/${LAMBDA_NAME}/invocations \
--passthrough-behavior WHEN_NO_MATCH
{
    "type": "AWS_PROXY",
    "httpMethod": "POST",
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/user_lambda_param/invocations"
},
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "vzz3pv8uch",
    "cacheKeyParameters": []
}
e460502@4-31-4-31:~/p2$ HTTP_METHOD=GET ; LAMBDA_NAME=user_lambda_param
e460502@4-31-4-31:~/p2$ awslocal apigateway put-integration --region ${REGION} --rest-api-id ${API_ID} \
--resource-id ${RESOURCE_ID} --http-method ${HTTP_METHOD} --type AWS_PROXY --integration-http-method GET \
--uri arn:aws:apigateway:${REGION}:lambda:path/2015-03-31/functions/${LAMBDA_NAME}/invocations --passthrough-behavior WHEN_NO_MATCH
{
    "type": "AWS_PROXY",
    "httpMethod": "GET",
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/user_lambda_param/invocations"
},
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "vzz3pv8uch",
    "cacheKeyParameters": []
}
e460502@4-31-4-31:~/p2$ [ $? == 0 ] || echo "Failed: AWS / apigateway / put-integration"
```

```

e460502@4-31-4-31:~/p2$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "2wxgcbssne",
      "path": "/"
    },
    {
      "id": "n6ilk7kgoh",
      "parentId": "2wxgcbssne",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "vzz3pv8uch",
      "parentId": "n6ilk7kgoh",
      "pathPart": "{username}",
      "path": "/user/{username}",
      "resourceMethods": {
        "POST": {
          "httpMethod": "POST",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {},
          "methodIntegration": {
            "type": "AWS_PROXY",
            "httpMethod": "POST",
            "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/user_lambda_param/invocations"
          },
          "passthroughBehavior": "WHEN_NO_MATCH",
          "timeoutInMillis": 29000,
          "cacheNamespace": "vzz3pv8uch",
          "cacheKeyParameters": []
        }
      },
      "GET": {
        "httpMethod": "GET",
        "authorizationType": "NONE",
        "apiKeyRequired": false,
        "requestParameters": {
          "method.request.path.username": true
        },
        "methodResponses": {},
        "methodIntegration": {
          "type": "AWS_PROXY",
          "httpMethod": "GET",
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/user_lambda_param/invocations"
        },
        "passthroughBehavior": "WHEN_NO_MATCH",
        "timeoutInMillis": 29000,
        "cacheNamespace": "vzz3pv8uch",
        "cacheKeyParameters": []
      }
    }
  ]
}

```

**DESPLIEGUE**

```

e460502@4-31-4-31:~/p2$ STAGE=dev
e460502@4-31-4-31:~/p2$ awslocal apigateway create-deployment \
--region ${REGION} \
--rest-api-id ${API_ID} \
--stage-name ${STAGE}
{
    "id": "nbnx9ojqc9",
    "createdDate": 1697722790.0
}
e460502@4-31-4-31:~/p2$ [ $? == 0 ] || echo " Failed: AWS / apigateway / create-deployment"
e460502@4-31-4-31:~/p2$ awslocal apigateway get-deployments --rest-api-id ${API_ID}
{
    "items": [
        {
            "id": "nbnx9ojqc9",
            "createdDate": 1697722790.0
        }
    ]
}
e460502@4-31-4-31:~/p2$ awslocal apigateway get-stages --rest-api-id ${API_ID}
{
    "item": [
        {
            "deploymentId": "nbnx9ojqc9",
            "stageName": "dev",
            "cacheClusterEnabled": false,
            "cacheClusterStatus": "NOT_AVAILABLE",
            "methodSettings": {},
            "tracingEnabled": false
        }
    ]
}

```

## PRUEBAS

```

e460502@4-31-4-31:~/p2$ STAGE=dev
e460502@4-31-4-31:~/p2$ ENDPOINT=http://localhost:4566/restapis/${API_ID}/${STAGE}/_user_request_/user
e460502@4-31-4-31:~/p2$ curl -X POST ${ENDPOINT}/pepe \
-H 'Content-Type: application/json' \
-d '{"username": "tete", "data": "100"}'

```

## Función handler de user\_lambda\_param

```

def handler(event, context):
    LOGGER.info('Escribiendo en S3')
    print("Received event: " + json.dumps(event, indent=2))
    req=json.loads(event['body'])
    test_object_key = f"users/{req['username']}"
    S3_CLIENT.put_object(
        Bucket=BUCKET_NAME,
        Key=test_object_key,
        Body=json.dumps(req)
    )
    resp_body=f'{test_object_key} placed into S3'
    # API GW compliant response
    resp = {
        "isBase64Encoded": False,
        "statusCode": 200,
        "headers": {
            "content-type": "application/json"
        },
        "body": json.dumps(resp_body)
    }
    return resp

```

### Pregunta: Mejora del API

1. ¿Qué pasos han sido necesarios para usar la nueva lambda en lugar de la anterior? Indica los comandos que has empleado.

Nosotros como se puede observar hemos realizado la creación de la nueva lambda: user\_lambda\_param, luego la creación de la API y después la inserción de la nueva lambda en la API tanto en el get como en el post. Pero crear una nueva API no hubiera sido necesario ya que podíamos haber reemplazado el anterior user\_lambda por user\_lambda\_param dentro de la API.

## 6. DynamoDB

### Tarea: Creación de una tabla

```
e460502@4-31-4-31:~/p2$ awslocal dynamodb create-table \
--table-name si1p1 \
--attribute-definitions AttributeName=User,AttributeType=S \
--key-schema AttributeName=User,KeyType=HASH \
--region $REGION \
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "User",
                "AttributeType": "S"
            }
        ],
        "TableName": "si1p1",
        "KeySchema": [
            {
                "AttributeName": "User",
                "KeyType": "HASH"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": 1697724220.005,
        "ProvisionedThroughput": [
            {
                "ReadCapacityUnits": 5,
                "WriteCapacityUnits": 5
            }
        ],
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-east-1:000000000000:table/si1p1",
        "TableId": "1cfbe01c-db91-4a17-9582-0487ae410f5b"
    }
}
```

## Pregunta: Creación de una tabla

1. ¿De qué tipo son los distintos campos de la tabla?

Como podemos comprobar los campos son de tipo "S" lo cual indica que son una cadena de caracteres.

## Tarea: Insertando datos manualmente

```
e460502@4-31-4-31:~/p2$ awslocal dynamodb put-item \
--table-name si1p1 \
--item '{"User": {"S":"pipo"}, "EMail": {"S": "pipo@ss.com"}}'
e460502@4-31-4-31:~/p2$ awslocal dynamodb scan --table-name si1p1 --region us-east-1
{
    "Items": [
        {
            "EMail": {
                "S": "pipo@ss.com"
            },
            "User": {
                "S": "pipo"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

## Pregunta: Insertando datos manualmente

1. ¿Qué ocurre si creamos un ítem sin campo "User" o sin campo "Email"?

```
e460502@4-31-4-31:~/p2$ awslocal dynamodb scan --table-name si1p1 --region us-east-1
{
    "Items": [
        {
            "EMail": {
                "S": "pipo@ss.com"
            },
            "User": {
                "S": "pipo"
            }
        },
        {
            "User": {
                "S": "dani"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```

```
e460502@4-31-4-31:~/p2$ awslocal dynamodb put-item --table-name si1p1 --item '{"EMail": {"S":"dani@gmail.com"}}'
An error occurred (ValidationException) when calling the PutItem operation: One of the required keys was not given a value
```

Como podemos comprobar, no creará un nuevo ítem, si no se especifica el campo USER, y al intentarlo nos devolverá un error ya que el User es un campo obligatorio.

Sin embargo, al crear un ítem sin campo Email se crea correctamente pero sin ese campo.

## Tarea: Lectura de datos

```
e462883@4-32-4-32:~/p2$ awslocal dynamodb get-item --table-name si1p1 --key '{"User": {"S": "pipo"}, "EMail": {"S": "pipo@ss.com"}}'

An error occurred (ValidationException) when calling the GetItem operation: The number of conditions on the keys is invalid
e462883@4-32-4-32:~/p2$ awslocal dynamodb get-item --table-name si1p1 --key '{"User": {"S": "pipo"}}'
{
    "Item": {
        "EMail": {
            "S": "pipo@ss.com"
        },
        "User": {
            "S": "pipo"
        }
    }
}
e462883@4-32-4-32:~/p2$ awslocal dynamodb scan --table-name si1p1 --region us-east-1
{
    "Items": [
        {
            "EMail": {
                "S": "pipo@ss.com"
            },
            "User": {
                "S": "pipo"
            }
        },
        {
            "User": {
                "S": "dani"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```

## Pregunta: Lectura de datos

1. ¿Qué ocurre si creamos un item sin campo “User”?

Como ya hemos respondido anteriormente, no creará un nuevo item, y al intentarlo nos devolverá un error ya que el User es un campo obligatorio.

2. ¿Qué ocurre si volvemos a insertar el mismo usuario sin el campo “Email”?

```
e462883@4-32-4-32:~/p2$ awslocal dynamodb scan --table-name si1p1 --region us-east-1
{
    "Items": [
        {
            "User": {
                "S": "pipo"
            }
        },
        {
            "User": {
                "S": "dani"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```

Lo que estaríamos consiguiendo es actualizar la json de dicho usuario, borrando en este caso el campo EMail, ya que en la creación del usuario no se especifica este campo.

## Tarea: Lambda con dynamoDB

Tras varias ejecuciones fallidas y la búsqueda de una solución nos dimos cuenta de que el comando proporcionado tenía errores. El comando es el siguiente:

```
curl -X POST ${ENDPOINT}/user2 -H 'Content-Type: application/json' \ -d '{"email": "mimail@dominio.es"}'
```

## Pregunta: Insertando datos manualmente

1. ¿Qué pasos han sido necesarios para cambiar de función lambda?

Como hemos repetido en la ocasión de la integración de la lambda relacionada con user\_lambda\_param. Hemos creado el lambda, después hemos verificado su funcionamiento. Tras ello lo hemos integrado en POST.

```
e460502@4-31-4-31:~/pz$ curl -X POST ${ENDPOINT}/user2 -H 'Content-Type: application/json' -d '{"email": "user2@dominio.es"}'
e460502@4-31-4-31:~/pz$ awslocal dynamodb scan --table-name slip1 --region us-east-1
{
    "Items": [
        {
            "EMail": {
                "S": "user2@dominio.es"
            },
            "User": {
                "S": "user2"
            }
        },
        {
            "EMail": {
                "S": "mimail@dominio.es"
            },
            "User": {
                "S": "bobt"
            }
        },
        {
            "EMail": {
                "S": "ptpo@ss.com"
            },
            "User": {
                "S": "ptpo"
            }
        },
        {
            "User": {
                "S": "dant"
            }
        }
    ],
    "Count": 4,
    "ScannedCount": 4,
    "ConsumedCapacity": null
}

e460502@4-31-4-31:~/pz$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
    "items": [
        {
            "id": "2wxgcbssne",
            "path": "/"
        },
        {
            "id": "n6ilk7kgoh",
            "parentId": "2wxgcbssne",
            "pathPart": "user",
            "path": "/user"
        },
        {
            "id": "vzz3pv8uch",
            "parentId": "n6ilk7kgoh",
            "pathPart": "{username}",
            "path": "/user/{username}",
            "resourceMethods": {
                "POST": {
                    "httpMethod": "POST",
                    "authorizationType": "NONE",
                    "apiKeyRequired": false,
                    "requestParameters": {
                        "method.request.path.username": true
                    },
                    "methodResponses": {},
                    "methodIntegration": {
                        "type": "AWS_PROXY",
                        "httpMethod": "POST",
                        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/user_lambda_dynamo/invocations"
                    },
                    "passthroughBehavior": "WHEN_NO_MATCH",
                    "timeoutInMillis": 29000,
                    "cacheNamespace": "vzz3pv8uch",
                    "cacheKeyParameters": []
                }
            }
        }
    ]
}
```

2. ¿Qué datos se han añadido a la tabla si1p1 tras ejecutar el comando curl anterior?

En nuestro caso tras las pruebas hemos añadido los usuarios bobí, con su respectivo correo y user2 con su respectivo usuario

```
e460502@4-31-4-31:~/p2$ awslocal dynamodb scan --table-name si1p1 --region us-east-1
{
    "Items": [
        {
            "EMail": {
                "S": "user2@dominio.es"
            },
            "User": {
                "S": "user2"
            }
        },
        {
            "EMail": {
                "S": "mimail@dominio.es"
            },
            "User": {
                "S": "bobi"
            }
        },
        {
            "EMail": {
                "S": "pipo@ss.com"
            },
            "User": {
                "S": "pipo"
            }
        },
        {
            "User": {
                "S": "dani"
            }
        }
    ],
    "Count": 4,
    "ScannedCount": 4,
    "ConsumedCapacity": null
}
```

## Tarea: GET /user/{username} con dynamoDB

```
e460502@4-31-4-31:~/p2$ curl -X GET "${ENDPOINT}/user?email=mimail@dominio.es" -H 'Content-Type: application/json' {"EMail": "user2@dominio.es", "User": "user2"}e460502@4-31-4-31:~/p2$
```

Como podemos comprobar hemos obtenido la información tanto el email como el usuario de un usuario dado.

```
def handler(event, context):
    LOGGER.info('Buscando usuario en DynamoDB')
    try:
        username = event['pathParameters']['username']

        # Realizar una operación de lectura en DynamoDB
        table = DYNAMODB_RESOURCE.Table(TABLE_NAME)
        response = table.get_item(
            Key={
                'User': username
            }
        )
        # Comprobar si se encontró el usuario en la tabla
        if 'Item' in response:
            user_data = response['Item']
        else:
            user_data = {} # Devolver un objeto vacío si no se encuentra el usuario
    except Exception:
        print("Error....")
        print(f"error trace {traceback.format_exc()}")
        LOGGER.info(f"error trace {traceback.format_exc()}")
        resp_body = {'msg': f"error trace {traceback.format_exc()}"}
    else:
        print("OK...")
        print(f"{username} encontrado en DynamoDB")
        LOGGER.info(f"{username} encontrado en DynamoDB")
        resp_body = user_data

    # API GW compliant response
    resp = {
        "isBase64Encoded": False,
        "statusCode": 200,
        "headers": {
            "content-type": "application/json"
        },
        "body": json.dumps(resp_body)
    }
    return resp
```

Tras una realización de búsquedas sobre como podríamos abordar dicho problema, encontramos una manera de realizar una operación de lectura en DynamoDB. Leemos la tabla entera y capturamos en la variable response el valor que queremos buscar. Si dicho elemento ha sido encontrado en la tabla devolveremos que ha sido encontrado e insertamos en resp\_body , desde user\_data.

## Pregunta: GET /user/{username} con dynamoDB

1. ¿Qué pasos han sido necesarios para crear e integrar la nueva función lambda?

Como hemos mencionado en otras partes hemos generado el lambda y lo hemos integrado en la funcionalidad de GET.

2. ¿Qué comando curl podemos usar para probar el método GET /user/{username}?. Da ejemplos también con la respuesta tanto si existe como si no existe el usuario.

```
e460502@4-31-4-31:~/p2$ curl -X GET "${ENDPOINT}/user?email=mimail@dominio.es" -H 'Content-Type: application/json' {"Email": "user2@dominio.es", "User": "user2"}e460502@4-31-4-31:~/p2$
```

como hemos podido comprobar devuelve el contenido de dicho usuario correctamente, y según está implementado el código, si no se encuentra un usuario que coincida con el definido en la llamada, la función no devolverá ningún ítem.

## Listado de archivos

- **user\_lambda\_param.py**
- **user\_get\_lambda\_dynamo.py**
- **user\_rest.py (modificado)**
- **Dockerfile (modificado)**