

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

MEMORIA DE LA PRÁCTICA 1

SISTEMAS INFORMÁTICOS

Sergio Larriba Moreno
Javier Valero Velázquez

ÍNDICE

PARTE I.....	2
MICRO-SERVICIO EN QUART.....	3
DESPLIEGUE CON HYPERCORN.....	6
CONTENEDORES DOCKER.....	7
CREACIÓN DE UNA IMAGEN DE CONTENEDOR CON UN DOCKERFILE.....	14
DOCKER REGISTRY.....	18
PARTE II.....	20
AWS 33.....	21
AWS LAMBDA.....	21
API GATEWAY.....	25
DYNAMODB.....	33

PARTE I

MICRO-SERVICIO EN QUART

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

La anotación @app.route es un decorador que mapea la URL a una función específica. Por ejemplo en el archivo user_rest.py proporcionado se mapea la dirección raíz con la función "hello()", es decir, define una ruta para dicha función cuando accedas a la URL del sitio, como por ejemplo "http://paginaweb.com/" o "http://localhost:5000/" si lo estás ejecutando localmente. Además, debido al segundo argumento "methods=["GET"]", cuando accedas a la URL y realices una solicitud GET, la función "hello()" se ejecutará.

La anotación @app.get es un decorador que indica a "Quart" que la función que está justo debajo, en este caso "user_get", es la encargada de gestionar las peticiones. Este decorador especifica la ruta, lo que crea un método get en la ruta del sitio, el cuál es devuelto por la función envuelta. Por ejemplo, si accedes a la URL "http://paginaweb.com/user/javi", el valor de "javi" se capturará como <username> y se ejecutará la función user_get. Dicha función abrirá el archivo "data.json" correspondiente al nombre de usuario "javi", leerá su contenido, lo devolverá y se mostrará en el navegador del usuario.

El decorador @app.put vincula la función "user_put" con solicitudes HTTP PUT en la ruta "/user/<username>", de modo que esta función se ejecuta cuando se accede a esa ruta con el método PUT. La ruta "/user/<username>" corresponde a las solicitudes PUT y <username> es una variable de ruta que permite capturar el valor del ID del elemento que se debe actualizar. La función "user_put" se utiliza para crear un directorio de usuario y almacenar datos JSON relacionados con ese usuario en un archivo en el directorio.

2 - ¿Qué ventajas o inconvenientes crees que pueden tener?

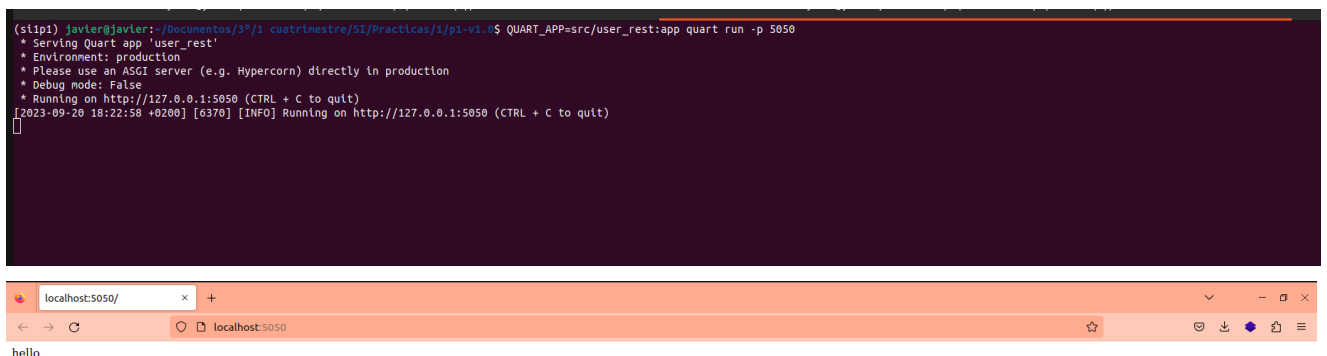
Las anotaciones anteriores pueden tener las siguientes **desventajas**:

- **Complejidad acumulativa** - Se puede dar el caso en el que en aplicaciones web grandes y complejas, es posible tener múltiples decoradores en diferentes partes del código, lo que puede traer como consecuencia una complejidad acumulativa, dificultar la comprensión de cómo se comporta la aplicación en su conjunto.
- **Posible sobrecarga** - En sistemas muy grandes el uso excesivo de decoradores puede llevar a una sobrecarga, lo que dificulta la comprensión del flujo de la aplicación y la identificación de problemas.
- **Incompatibilidad entre frameworks** - Los decoradores específicos de un framework pueden no ser compatibles con otros frameworks web. Esto puede hacer que sea más difícil migrar o reutilizar código en diferentes entornos.
- **Falta de flexibilidad** - Algunos decoradores pueden ser inflexibles en cuanto a cómo se manejan las rutas y las solicitudes. Supongamos que creamos una aplicación web utilizando un framework específico para manejar solicitudes HTTP GET, el decorador funciona muy bien para muchas de las rutas pero ahora tenemos una ruta especial en la que necesitamos hacer algunas validaciones adicionales antes de procesar la solicitud. Aquí nos encontramos con un problema, el no poder extender fácilmente el comportamiento predeterminado del decorador GET para esta ruta específica.

Sin embargo, los decoradores también tienen sus **ventajas**, como por ejemplo:

- **Escalabilidad** - Permiten agregar nuevas rutas y controladores de manera sencilla a medida que la aplicación web crece.
- **Sintaxis clara y expresiva** - Permiten definir las rutas y los métodos HTTP de una manera legible y natural, lo que facilita la comprensión del enrutamiento y el manejo de solicitudes HTTP en el código.
- **Abstracción de tareas comunes** - Los decoradores encapsulan tareas comunes relacionadas con el manejo de solicitudes HTTP, como el análisis de parámetros de URL, el procesamiento de formularios y la generación de respuestas. Esto ahorra tiempo y reduce la cantidad de código repetitivo.
- **Facilita el enrutamiento dinámico** - Pueden manejar rutas con parámetros variables, lo que permite la creación de rutas dinámicas que capturen valores de la URL como identificadores de usuarios o nombres de archivos y los pasan a los controladores.

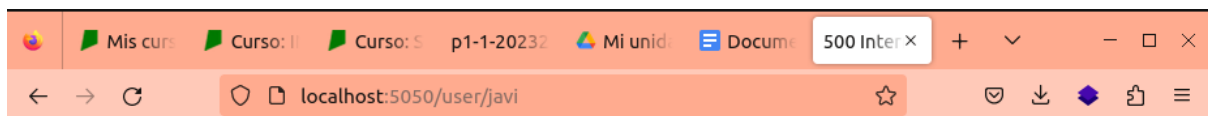
Ejecución de user_rest.py ¿Qué crees que ha sucedido?



The image shows a terminal window and a web browser. The terminal window displays the command `QUART_APP=src/user_rest:app quart run -p 5050` and its output, which includes status messages like "Serving Quart app 'user_rest'", "Environment: production", and "Running on http://127.0.0.1:5050". The web browser shows the address `localhost:5050/` and the response "hello".

El primer comando activa el entorno virtual, el segundo comprueba que build/users existe y en caso contrario los crea, por último se establece a QUART_APP el valor src/user_rest:app e inicia la aplicación en el puerto 5050.

Una vez se está ejecutando, si accedes a localhost:5050 aparece el mensaje "hello" como se ha explicado en el apartado 1 y si accedes a <http://localhost:5050/user/<username>> aparece un error tanto en consola como en la pantalla por que todavia no hemos ejecutado `curl -X PUT`.



Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

La respuesta es OK.

```
javier@javier: ~/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0
(slip1) javier@javier:~/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0$ QUART_APP=src/user_rest:app quart run -p 5050
* Serving Quart app 'user_rest'
* Environment: production
* Please use an ASGI server (e.g. Hypercorn) directly in production
* Debug mode: False
* Running on http://127.0.0.1:5050 (CTRL + C to quit)
[2023-09-20 18:43:26 +0200] [8780] [INFO] Running on http://127.0.0.1:5050 (CTRL + C to quit)
[2023-09-20 18:43:34 +0200] [8780] [INFO] 127.0.0.1:47338 GET / 1.1 200 5 1516
[2023-09-20 18:43:48 +0200] [8780] [INFO] 127.0.0.1:35394 GET /user/javi 1.1 500 265 1664
[2023-09-20 18:43:48.141] ERROR in app: Exception on request GET /user/javi
Traceback (most recent call last):
  File "/home/javier/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0/venv/slip1/lib/python3.10/site-packages/quart/app.py", line 1650, in handle_request
    return await self.full_dispatch_request(request_context)
  File "/home/javier/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0/venv/slip1/lib/python3.10/site-packages/quart/app.py", line 1675, in full_dispatch_request
    result = await self.handle_user_exception(error)
  File "/home/javier/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0/venv/slip1/lib/python3.10/site-packages/quart/app.py", line 1107, in handle_user_exception
    raise error
  File "/home/javier/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0/venv/slip1/lib/python3.10/site-packages/quart/app.py", line 1673, in full_dispatch_request
    result = await self.dispatch_request(request_context)
  File "/home/javier/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0/venv/slip1/lib/python3.10/site-packages/quart/app.py", line 1718, in dispatch_request
    return await self.ensure_async(handler)(**request.view_args)
  File "/home/javier/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0/src/user_rest.py", line 23, in user_get
    with open(f"{USERDIR}/{username}/data.json", "r") as ff:
FileNotFoundError: [Errno 2] No such file or directory: './build/users/javi/data.json'
```

Petición a user_rest.py

```
javier@javier:~/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0$ curl -X PUT http://localhost:5050/user/myusername -H 'Content-Type: application/json' -d '{"firstName": "myFirstName"}'
{"status": "OK"}javier@javier:~/Documentos/3º/1 cuatrimestre/SI/Practicas/1/p1-v1.0$
```

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

El comando curl es una herramienta para realizar solicitudes HTTP. En este caso, estamos realizando una solicitud HTTP PUT. Al ejecutar el mandato curl, se crea el archivo data.json en la carpeta build/users.



2 - ¿Cuál es la respuesta del microservicio?

La respuesta del microservicio ha sido “{“status”:“OK”}”. Lo que nos indica que se ha ejecutado correctamente mediante el comando curl la solicitud HTTP PUT.

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

Si, se ha generado el archivo “data.json” en la ruta “...p1-v1.0\build\users\myusername” cuyo contenido es el ejecutado al realizar la petición HTTP PUT: '{"firstName": "myFirstName"}'.

Pregunta: petición HTTP

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

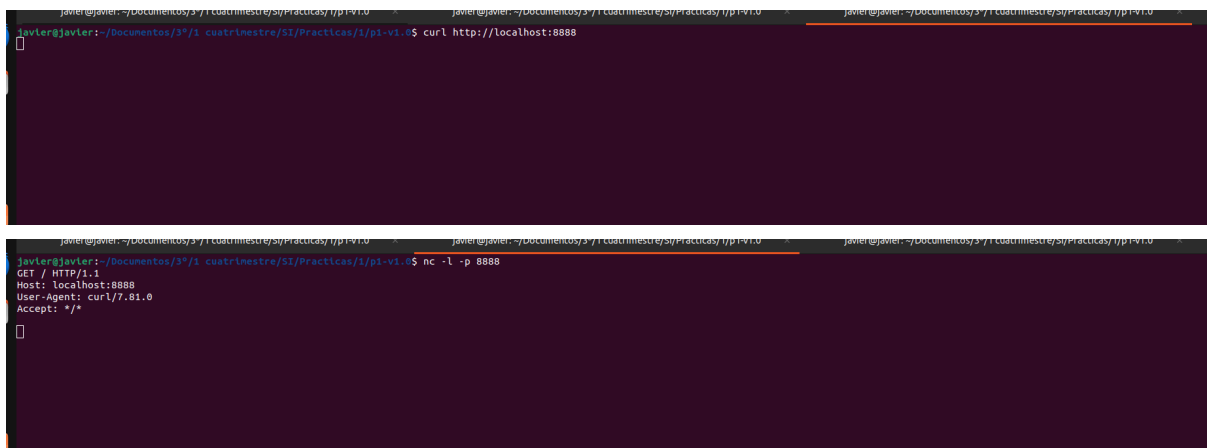
GET / HTTP/1.1

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 “Content-Type” de la cabecera HTTP. Este campo especifica el tipo de medio (por ejemplo, texto, imagen, aplicación json, etc) del contenido que se encuentra en el campo de la solicitud. En este caso el campo “Content-Type” contiene “application/json”.

3 - ¿Qué separa la cabecera del cuerpo?

La cabecera y el cuerpo de una solicitud o respuesta HTTP están separados por una línea en blanco. Esta línea en blanco es un indicador de que la cabecera ha terminado y que el cuerpo de la solicitud o respuesta comienza a continuación.

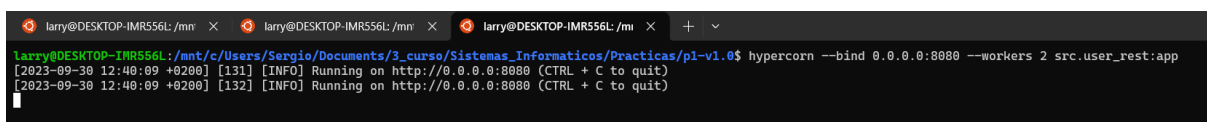


```
javier@javier:~/Documentos/3º/1 cuatrimestre/SI/Practicas/pl-v1.0$ curl http://localhost:8888
[]

javier@javier:~/Documentos/3º/1 cuatrimestre/SI/Practicas/pl-v1.0$ nc -l -p 8888
GET / HTTP/1.1
Host: localhost:8888
User-Agent: curl/7.81.0
Accept: */*
[]
```

DESPLIEGUE CON HYPERCORN

Ejecución hypercorn

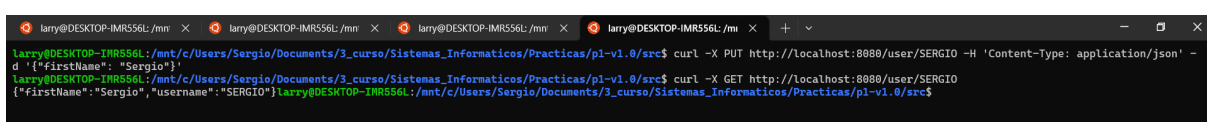


```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ hypercorn --bind 0.0.0.0:8080 --workers 2 src.user_rest:app
[2023-09-30 12:40:09 +0200] [131] [INFO] Running on http://0.0.0.0:8080 (CTRL + C to quit)
[2023-09-30 12:40:09 +0200] [132] [INFO] Running on http://0.0.0.0:8080 (CTRL + C to quit)
```

1 - ¿Qué crees que ha sucedido?

Lo que ha sucedido es que hemos iniciado un servidor web usando Hypercorn en el puerto 8080 y configurado para manejar dos trabajadores “workers” simultáneamente. Los mensajes que se muestran en la salida de Hypercorn indican que se ha iniciado correctamente. El servidor está escuchando en las direcciones IP ‘0.0.0.0’ en el puerto ‘8080’, y los números entre corchetes [131] y [132] son los identificadores de los trabajadores. Todo ello nos lleva a la conclusión de que el servidor está listo para recibir solicitudes en la dirección ‘<http://0.0.0.0:8080>’.

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



```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0/src$ curl -X PUT http://localhost:8080/user/SERGIO -H 'Content-Type: application/json' -d '{"firstName": "Sergio"}'
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0/src$ curl -X GET http://localhost:8080/user/SERGIO
```

En primer lugar hacemos un PUT para insertar un .json nuevo con datos diferentes para poder así comprobar la correcta ejecución de la petición GET en hypercorn. Podemos apreciar que el puerto ha pasado a ser 8080 en vez de 5050. Esto es debido a que al ejecutar “hypercorn ... :8080 ...” hemos iniciado un servidor en el puerto 8080, por lo que si queremos extraer información de dicho servidor, habrá que especificarle el puerto como “8080”. Pero, a excepción del puerto, la petición GET es la misma respecto a la ejecución sin hypercorn.

CONTENEDORES DOCKER

Ejecución docker info

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker info
Client: Docker Engine - Community
Version: 24.0.6
Context: default
Debug Mode: false
Plugins:
buildx: Docker Buildx (Docker Inc.)
  Version: v0.11.2-desktop.5
  Path: /usr/local/lib/docker/cli-plugins/docker-buildx
compose: Docker Compose (Docker Inc.)
  Version: v2.22.0-desktop.2
  Path: /usr/local/lib/docker/cli-plugins/docker-compose
dev: Docker Dev Environments (Docker Inc.)
  Version: v0.1.0
  Path: /usr/local/lib/docker/cli-plugins/docker-dev
extension: Manages Docker extensions (Docker Inc.)
  Version: v0.2.20
  Path: /usr/local/lib/docker/cli-plugins/docker-extension
init: Creates Docker-related starter files for your project (Docker Inc.)
  Version: v0.1.0-beta.8
  Path: /usr/local/lib/docker/cli-plugins/docker-init
sbom: View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc.)
  Version: 0.6.0
  Path: /usr/local/lib/docker/cli-plugins/docker-sbom
scan: Docker Scan (Docker Inc.)
  Version: v0.26.0
  Path: /usr/local/lib/docker/cli-plugins/docker-scan
scout: Docker Scout (Docker Inc.)
  Version: v1.0.7
  Path: /usr/local/lib/docker/cli-plugins/docker-scout

Server:
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 24.0.6
Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
Logging Driver: json-file
Cgroup Driver: cgroupfs
Cgroup Version: 1
Plugins:
```



```

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: 8165feabfdfe38c65b599c4993d227328c231fca
runc version: v1.1.8-0-g82f18fe
init version: de40ad0
Security Options:
  seccomp
    Profile: unconfined
Kernel Version: 5.15.90.1-microsoft-standard-WSL2
Operating System: Docker Desktop
OSType: linux
Architecture: x86_64
CPUs: 20
Total Memory: 7.623GiB
Name: docker-desktop
ID: a79e67dc-a8b6-47ff-bef2-6c9c16157ca0
Docker Root Dir: /var/lib/docker
Debug Mode: false
HTTP Proxy: http.docker.internal:3128
HTTPS Proxy: http.docker.internal:3128
No Proxy: hubproxy.docker.internal
Experimental: false
Insecure Registries:
  hubproxy.docker.internal:5555
  127.0.0.0/8
Live Restore Enabled: false

```

```

WARNING: No blkio throttle.read_bps_device support
WARNING: No blkio throttle.write_bps_device support
WARNING: No blkio throttle.read_iops_device support
WARNING: No blkio throttle.write_iops_device support
WARNING: daemon is not using the default seccomp profile

```

```

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$

```

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

La configuración del demonio arrancado se guarda en el directorio “/var/lib/docker”.

2 - ¿Dónde se almacenan los contenedores?

Los contenedores se almacenan en un repositorio para contenedores, en este caso, dicho repositorio está ubicado en el directorio “/var/lib/docker”.

3 - ¿Qué tecnología de almacenamiento se usa para los contenedores? ¿Qué es copy-on-write? ¿Lo usa docker?

Docker utiliza una tecnología de almacenamiento por capas (layered storage) para los contenedores. Copy-on-write (copiar al escribir) es una estrategia de almacenamiento que se usa para optimizar el uso de recursos y mejorar la velocidad de creación de contenedores e imágenes. Docker la usa de la siguiente manera: cuando se crea un contenedor a partir de una imagen en Docker, en lugar de copiar todos los archivos de la imagen, Docker crea una capa adicional que se monta sobre la imagen base. Esta capa es de solo lectura y contiene la imagen base, así que, cuando se realizan modificaciones en archivos dentro del contenedor, Docker utiliza dicha técnica para copiar solo los archivos que se modifican, manteniendo el resto de la capa de solo lectura intacta.

Ejecución run whoami

```
larry@DESKTOP-IMR556L: /mi × + v
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker run alpine whoami
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
96526aa774ef: Pull complete
Digest: sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c016293fc851978
Status: Downloaded newer image for alpine:latest
root
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$
```

1 - ¿Qué crees que ha sucedido?

Lo que ha sucedido es que al ejecutar dicho comando, se nos ha iniciado un contenedor. En primer lugar, ha visto si se encuentra la imagen en nuestro directorio, como no la ha encontrado la descarga. Después crea un contenedor basado en la imagen “alpine” y en él ejecuta el comando whoami, que nos indicará el nombre de usuario actual dentro del contenedor, en este caso, “root”

2 - ¿Crees que el usuario root del contenedor es el mismo que el del host?

Creemos que el usuario root del contenedor es diferente al del host porque aunque ambos tengan el mismo nombre de usuario (“root”) son entidades separadas y tienen sus propios contextos y permisos. Dentro de un contenedor, el usuario root tiene control total sobre el entorno de dicho contenedor, pero sus acciones están aisladas del host. Las acciones realizadas por el usuario root dentro del contenedor no afectan al host.

Ejecución list images

```
larry@DESKTOP-IMR556L: /mi × + v
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
alpine latest 8ca4688f4f35 4 days ago 7.34MB
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$
```

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

En primer lugar, podemos observar como al ejecutar el comando “docker images” nos devuelve un listado completo de todas las imágenes que hemos descargado en nuestra máquina. En nuestro caso, al haber aplicado anteriormente el comando “docker run alpine whoami” nos ha descargado la última versión de la imagen “alpine”. Nos muestra el nombre de la imagen “alpine”, la versión (TAG), el id de la imagen, cuando fue creada y finalmente el tamaño de dicha imagen.

Al ejecutar docker pull nos va a descargar la imagen que le especifiquemos, por lo que la añadirá a la lista de imágenes descargadas en nuestra máquina y si volviésemos a ejecutar “docker images” nos aparecería la nueva imagen descargada con el comando pull. Cabe destacar que si la imagen que deseamos descargar ya existe y es igual a la que se encuentra en el registro de Docker, entonces no descargará la nueva imagen completa, sino que actualizará las capas que hayan cambiado en la nueva versión de la imagen.

Ejecución docker ps

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a3fbef75ddeb	alpine	"whoami"	22 minutes ago	Exited (0) 22 minutes ago		brave_sammet

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$
```

¿Qué crees que muestra la salida?

Nos muestra una tabla con los datos de los contenedores que tenemos, en ella se encuentra el id del contenedor, su nombre, ambos sirven para hacer referencia a dicho contenedor cuando queramos eliminarlo o mapear por ejemplo, el comando aplicado dentro del contenedor, cuando fue creado y si está en ejecución o no, y los puertos. Al ejecutarlo con la opción -a nos muestra además también aquellos que no se están ejecutando (como ocurre en la imagen). Tal y como se puede apreciar, solo tenemos 1, correspondiente a la imagen alpine.

Ejecución docker pull y run -ti

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker pull ubuntu:22.04
22.04: Pulling from library/ubuntu
37aaf24cf781: Pull complete
Digest: sha256:9b8dec3bf938bc80f8e758d856e96fd5f56c39d44b0cff351e847bb1b01ea
Status: Downloaded newer image for ubuntu:22.04
docker.io/library/ubuntu:22.04
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker run -ti --name u22.04 ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
Digest: sha256:9b8dec3bf938bc80f8e758d856e96fd5f56c39d44b0cff351e847bb1b01ea
Status: Downloaded newer image for ubuntu:latest
root@b1dfac21caa:/#
```

1 - ¿Qué hace la opción -ti?

La opción -ti significa que estamos ejecutando el contenedor en modo interactivo asignándole una terminal para él mismo. --name establece un nombre personalizado para el contenedor, en este caso, "u22.04". Finalmente, "ubuntu" es el nombre de la imagen base que se utilizará para el contenedor.

2 - ¿Qué ha sucedido?

Hemos descargado en primer lugar la imagen "ubuntu" con la versión "22.04" desde Docker Hub. Después, al ejecutar el comando run, hemos creado un contenedor interactivo basado en esa imagen, lo que nos permite trabajar en un entorno de Ubuntu 22.04 dentro del ordenador.

Ejecución docker inspect

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker inspect u22.04 | jq '.[].Config.Cmd[]'
```

```
"/bin/bash"
```

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$
```

1 - ¿Qué hace el mandato jq?

Jq es un procesador de JSON que permite filtrar y transformar datos JSON. En este caso, está extrayendo el valor de la clave "Cmd" del objeto JSON resultante de la inspección del contenedor.

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

El resultado del comando está relacionado con la tarea anterior de obtener información sobre el comando que se ejecuta cuando se inicia el contenedor u22.04, en particular "docker inspect u22.04" se utiliza para obtener información detallada sobre el contenedor u22.04, incluyendo su configuración y "jq '.[].Config.Cmd[]'" se utiliza para filtrar la salida de docker inspect y extraer la lista de comandos y argumentos que se ejecutarán cuando se inicie el contenedor. Entonces, al inspeccionar la configuración usando "docker inspect" se encuentra que el valor de la clave "Cmd" en la configuración del contenedor se establece en "/bin/bash", que es el comando que se ejecutará automáticamente cuando inicies el contenedor en modo interactivo.

Ejecución docker start

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker start u22.04
u22.04
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
b1dfacb21caa   ubuntu   "/bin/bash"             49 minutes ago Up 10 seconds   u22.04
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$
```

1 - ¿Qué ha sucedido?

Lo que ha sucedido es que hemos ejecutado el contenedor de nombre u22.04. Podemos apreciar que ha salido bien porque al ejecutar docker ps aparece, lo que significa que dicho contenedor está en ejecución.

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

Para borrar un contenedor hay que hacer 2 pasos:

1. Parar la ejecución del contenedor con el comando: "docker stop u22.04"
2. Borrar el contenedor con el comando: "docker rm u22.04"

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker stop u22.04
u22.04
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
b1dfacb21caa   ubuntu   "/bin/bash"             54 minutes ago Exited (137) 19 seconds ago   u22.04
a3fbef75ddeb   alpine    "whoami"                About an hour ago Exited (0) About an hour ago   brave_sammet
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker rm u22.04
u22.04
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
a3fbef75ddeb   alpine    "whoami"                About an hour ago Exited (0) About an hour ago   brave_sammet
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$
```

Ejecución de docker run -d

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker run -ti -d --name u22.04d ubuntu
7d26a6917e22a0d7e6af08e60bbafdb3921b9deb363f4550fbada615f222e2d
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$

larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
7d26a6917e22   ubuntu    "/bin/bash"             6 minutes ago Up 6 minutes   u22.04d
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$
```

¿Qué hace la opción -d?

La opción -d permite que el contenedor se ejecute en segundo plano y no bloqueará la terminal actual. Esto significa que después de ejecutar el comando “docker run” se devolverá el control a la terminal para que podamos seguir usando la misma.

Ejecución docker exec

```
root@7d26a6917e22: /
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker exec -ti u22.04d /bin/bash
root@7d26a6917e22: /# mkdir -p /si1/users
root@7d26a6917e22: /# touch /si1/users/test
root@7d26a6917e22: /#
```

¿Qué ha sucedido?

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker exec -ti u22.04d /bin/bash
root@65524d76d97e: /# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@65524d76d97e: /# mkdir -p /si1/users
root@65524d76d97e: /# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin si1 srv sys tmp usr var
root@65524d76d97e: /# touch /si1/users/test
root@65524d76d97e: /# cd /si1/users
root@65524d76d97e: /si1/users# ls
test
root@65524d76d97e: /si1/users#
```

Lo que ha sucedido es que hemos creado dentro del contenedor u22.04d hemos creado un directorio llamado “/si1/users” y un archivo llamado “test” en ese directorio.

Pregunta: docker volume

1 - ¿Qué hace la opción -v?

La opción -v se utiliza para montar un volumen en un contenedor Docker. Esto permite que un directorio o archivo en el sistema de archivos del host esté disponible dentro del contenedor.

2 - ¿Qué ha sucedido al ejecutar los mandatos anteriores?

- “[[-d si1/users/testv]] || mkdir -p si1/users/testv ” -> Verifica si el directorio “si1/users/testv” existe en el sistema de archivos del host. Si no existe, crea el directorio.

- `"docker run -ti -v ${PWD}/si1/users/testv:/si1/users/testv alpine \ touch /si1/users/testv/afile"` -> Este comando ejecuta un contenedor Alpine y monta el directorio `"si1/users/testv"` del sistema de archivos del host en el directorio `"/si1/users/testv"` dentro del contenedor. Luego, el comando `touch`, crea dentro del contenedor un archivo llamado `"afile"` dentro de ese directorio.
- `"docker run -ti -v ${PWD}/si1/users/testv:/si1/users/testv alpine \ ls -al /si1/users/testv/afile"` -> Este comando ejecuta nuevamente un contenedor Alpine y monta el directorio `"si1/users/testv"` del sistema de archivos del host en el directorio `"/si1/users/testv"` dentro del contenedor. Después, el comando `ls -al` muestra información detallada sobre el archivo `"afile"` en ese directorio del contenedor.
- `"ls -al si1/users/testv/afile"` -> Muestra información detallada sobre el archivo `"afile"` en el directorio `"si1/users/testv"`.
- `"echo "a test" >>si1/users/testv/afile"` -> Agrega la línea `"a test"` al archivo `"afile"` en el directorio `"si1/users/testv"`.
- `"docker run -ti -v ${PWD}/si1/users/testv:/si1/users/testv alpine \ cat /si1/users/testv/afile"` -> Ejecuta un contenedor Alpine y monta el directorio `"si1/users/testv"` del sistema de archivos del host en el directorio `"/si1/users/testv"` dentro del contenedor. Después, ejecuta el comando `cat` en el contenedor, que muestra el contenido del archivo `"afile"` en ese directorio del contenedor.

Pregunta: limpieza

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

Para eliminar los contenedores hemos usado el siguiente comando: `"docker rm nombreContenedor"`. Como ya no estaban en ejecución, no ha sido necesario pararlos.

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

El comando que hemos usado para eliminar las imágenes es el siguiente: `"docker image rm nombrelimagen:version"`. Si al descargar la imagen no hemos especificado versión, al eliminarla tampoco y el sistema dará por hecho que es la última versión (latest).

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
65524d76d97e   ubuntu   "/bin/bash"   4 days ago   Exited (255) 18 seconds ago   u22.04d
a3fbef75ddeb   alpine   "whoami"     4 days ago   Exited (0) 4 days ago       brave_sammet
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker rm u22.04d
u22.04d
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker rm brave_sammet
brave_sammet
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
alpine         latest   8ca4688f4f35   9 days ago    7.34MB
ubuntu        22.04    3565a89d9e81   13 days ago   77.8MB
ubuntu        latest   3565a89d9e81   13 days ago   77.8MB
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker image rm alpine
Untagged: alpine:latest
Untagged: alpine@sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c016293fc851978
Deleted: sha256:8ca4688f4f356596b5ae539337c9941abc78eda10021d35cbc52659c74d9b443
Deleted: sha256:cc2447e1835a40530975ab80bb1f872fbab0f2a0faecf2ab16fbbb89b3589438
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker image rm ubuntu:22.04
Untagged: ubuntu:22.04
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker image rm ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:9b8dec3bf938bc80f7e758d856e96fdab5f56c39d44b0cff351e847bb1b01ea
Deleted: sha256:3565a89d9e81a4cb4cb2b0d947c7c11227a3f358dc216d19fc54bfd77cd5b542
Deleted: sha256:01d4eb4f381ac5a9964a14a650d7c074a2aa6e0789985d843f8eb3070b58f7d
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

CREACIÓN DE UNA IMAGEN DE CONTENEDOR CON UN DOCKERFILE

Pregunta: Dockerfile

1 - ¿Qué hace la sentencia FROM?

La sentencia FROM especifica la imagen base que se utilizará como punto de partida para construir la imagen actual. Es la primera instrucción en el Dockerfile

2 - ¿Qué hace la sentencia ENV?

La sentencia ENV se utiliza para definir variables de entorno dentro del contenedor Docker. En este caso define la variable NUMWORKERS con el valor de 2

3 - ¿Qué diferencia presenta ENV respecto de ARG?

La principal diferencia entre ENV y ARG es que las variables definidas con ENV son variables de entorno en tiempo de ejecución, mientras que las definidas con ARG son variables de construcción en tiempo de creación de la imagen.

4 - ¿Qué hace la sentencia RUN?

La sentencia RUN se utiliza para ejecutar comandos en el sistema de archivos de la imagen durante la construcción de la imagen. En este caso, la sentencia RUN del Dockerfile crea un directorio (siempre y cuando no esté creado ya previamente gracias a -p) "users", en el directorio "build", que a su vez se encuentra en el directorio "si1"

5 - ¿Qué hace la sentencia COPY?

La sentencia COPY se utiliza para agregar archivos locales al contenedor. En este caso, estamos copiando el archivo user_rest.py al contenedor.

6 - ¿Qué otra sentencia del Dockerfile tiene un contenido similar a COPY?

Otra sentencia similar a COPY es ADD, la cuál, además de copiar archivos y directorios, puede extraer archivos comprimidos y acepta URLs como fuente.

7 - ¿Qué hace la sentencia EXPOSE?

La sentencia EXPOSE se utiliza para indicar qué puertos expone el contenedor al exterior, sin embargo, no abre automáticamente los puertos, solo indica la intención de exponerlos. En el archivo Dockerfile se especifica que el puerto a exponer es el 8000.

8 - ¿Qué hace la sentencia CMD?

La sentencia CMD se utiliza para proporcionar un comando que se ejecutará cuando se inicie el contenedor a partir de la imagen. Cabe destacar que sólo puede haber una instrucción CMD en el Dockerfile, si hay varias, se ejecutará sólo el último.

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

Otras sentencias similares a CMD son RUN y ENTRYPOINT, la cuál también define un comando, pero permite que los argumentos se pasen cuando se inicia el contenedor.

Ejecución docker build

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ cd src
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker build --tag silp1:latest .
[+] Building 54.5s (13/13) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 353B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [1/7] FROM docker.io/library/ubuntu@sha256:9b8dec3bf938bc80f8e758d856e96fdab5f56c39d44b0cff351e847bb1b01ea
=> => resolve docker.io/library/ubuntu@sha256:9b8dec3bf938bc80f8e758d856e96fdab5f56c39d44b0cff351e847bb1b01ea 1.13kB / 1.13kB
=> => sha256:9b8dec3bf938bc80f8e758d856e96fdab5f56c39d44b0cff351e847bb1b01ea 1.13kB / 1.13kB
=> => sha256:b4b521bfcec90b11d2869e00fe1f2380c21cbfcd799ee35df8bd7ac09e6f63ea 424B / 424B
=> => sha256:3565a89d9e81a4cb2b0d947c7c11227a3f358dc216d19fc54bdf77cd5b542 2.30kB / 2.30kB
=> => sha256:37aaf24cf781dcc5b9a4f8aa5a99a40b60ae45d64dcb4f6d5a4b9e5ab7ab0894 29.54MB / 29.54MB
=> => extracting sha256:37aaf24cf781dcc5b9a4f8aa5a99a40b60ae45d64dcb4f6d5a4b9e5ab7ab0894
=> [internal] load build context
=> => transferring context: 3.37kB
=> [2/7] RUN apt update
=> [3/7] RUN apt install -y python3-pip
=> [4/7] RUN pip install hypercorn quart
=> [5/7] RUN mkdir -p /sil/build/users
=> [6/7] WORKDIR /sil
=> [7/7] COPY user_rest.py ./
=> exporting to image
=> => exporting layers
=> => writing image sha256:703f159315774bcd318c66b9e96f7b1a95291ca3fceb20d5b049ba7460395364
=> => naming to docker.io/library/silp1:latest
```

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker build --tag silp1:1.0 .
[+] Building 0.6s (12/12) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 353B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [1/7] FROM docker.io/library/ubuntu@sha256:9b8dec3bf938bc80f8e758d856e96fdab5f56c39d44b0cff351e847bb1b01ea
=> [internal] load build context
=> => transferring context: 34B
=> CACHED [2/7] RUN apt update
=> CACHED [3/7] RUN apt install -y python3-pip
=> CACHED [4/7] RUN pip install hypercorn quart
=> CACHED [5/7] RUN mkdir -p /sil/build/users
=> CACHED [6/7] WORKDIR /sil
=> CACHED [7/7] COPY user_rest.py ./
=> exporting to image
=> => exporting layers
=> => writing image sha256:703f159315774bcd318c66b9e96f7b1a95291ca3fceb20d5b049ba7460395364
=> => naming to docker.io/library/silp1:1.0
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
silp1 1.0 703f15931577 30 seconds ago 492MB
silp1 latest 703f15931577 30 seconds ago 492MB
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0/src$ cd -
/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```


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

Es mucho más rápido la creación de la segunda imagen porque Docker instala las imágenes por capas, entonces, al descargar la primera, como no teníamos ninguna de las capas ha tenido que instalar todas y cada una de ellas. Sin embargo, al volver a descargar la misma imagen pero con diferente versión, la gran mayoría de capas ya las había descargado anteriormente, por lo que solo tenía que descargar las necesarias de esa versión.

2 - Las imágenes constan de distintas capas: ¿Qué quiere decir eso?

En Docker, cada capa es una parte acumulativa e independiente de una imagen que contiene cambios en el sistema de archivos del sistema operativo. Estas capas se apilan una encima de la otra para construir la imagen final. Cada instrucción en un Dockerfile (como RUN, COPY, ...) genera una nueva capa de la imagen. Que las imágenes se construyan por capas tiene múltiples ventajas, como por ejemplo, la reutilización (como en la pregunta anterior), el versionamiento y la personalización de imágenes de manera eficiente.

Pregunta: docker build with user

¿Por qué los últimos pasos no utiliza la caché?

No utiliza la caché debido a las instrucciones comentadas:

- RUN useradd si1: crea un nuevo usuario llamado "si1" en el sistema de archivos de la imagen. Como esto cambia el sistema de archivos, se invalida la caché de capas a partir de este punto.
- RUN chown -R si1 /si1: esta instrucción cambia los permisos de los archivos y directorios en el sistema de archivos de la imagen, otorgando al usuario "si1" el control sobre ellos. Nuevamente esto modifica el sistema de archivos y hace que las capas posteriores no puedan utilizar la caché.
- USER si1: esta instrucción cambia el usuario que se usará para las instrucciones futuras. También modifica los archivos de la imagen y, por lo tanto, invalida la caché de capas.

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ cd src
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0/src$ docker build --tag silp1:1.0u .
[+] Building 2.3s (15/15) FINISHED
=> [internal] load .dockerignore
=> transferring context: 2B
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 350B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [1/9] FROM docker.io/library/ubuntu@sha256:9b8dec3bf938bc80f8e758d856e96fdab5f56c39d44b0cff351e847bb1b01ea
=> [internal] load build context
=> transferring context: 34B
=> CACHED [2/9] RUN apt update
=> CACHED [3/9] RUN apt install -y python3-pip
=> CACHED [4/9] RUN pip install hypercorn quart
=> CACHED [5/9] RUN mkdir -p /sil/build/users
=> [6/9] RUN useradd sil
=> [7/9] RUN chown -R sil /sil
=> [8/9] WORKDIR /sil
=> [9/9] COPY user_rest.py ./
=> exporting to image
=> exporting layers
=> writing image sha256:58e5cd5bdf24dacc8e85dc15f9e43ff16faa62637cd1c37ac3d5e34cc03f9f9
=> naming to docker.io/library/silp1:1.0u
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0/src$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
silp1         1.0u     58e5cd5bdf24   28 seconds ago 492MB
silp1         1.0      703f15931577   20 minutes ago 492MB
silp1         latest   703f15931577   20 minutes ago 492MB
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0/src$
```

Ejecución: docker run myimage

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker run --name silp1 -e NUMWORKERS=5 -d -p 8080:8000 silp1
26a0f4b798e897bba63ef72e1c231cd163f99e674ceaa3da851e80542e723d
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker exec -ti silp1 /bin/bash
root@26a0f4b798e8:/sil1# ps -forest -aef
UID          PID    PPID    C  STIME TTY          TIME CMD
root         14      0   0  12:06 pts/0    00:00:00 /bin/bash
root         22      0   0  12:06 pts/0    00:00:00 \_ ps --forest -aef
root          1      0   0  12:06 ?        00:00:00 /bin/sh -c hypercorn --bind 0.0.0.0:8000 --workers ${NUMWORKERS} user_rest:app
root          7      1   0  12:06 ?        00:00:00 /usr/bin/python3 /usr/local/bin/hypercorn --bind 0.0.0.0:8000 --workers 5 user_rest:app
root          8      7   0  12:06 ?        00:00:00 \_ /usr/bin/python3 -c from multiprocessing.resource_tracker import main;main(4)
root          9      7   1  12:06 ?        00:00:00 \_ /usr/bin/python3 -c from multiprocessing.spawn import spawn_main; spawn_main(tracker_fd=5, pipe_handle=7) --multiprocessing-fork
root         10      7   1  12:06 ?        00:00:00 \_ /usr/bin/python3 -c from multiprocessing.spawn import spawn_main; spawn_main(tracker_fd=5, pipe_handle=9) --multiprocessing-fork
root         11      7   1  12:06 ?        00:00:00 \_ /usr/bin/python3 -c from multiprocessing.spawn import spawn_main; spawn_main(tracker_fd=5, pipe_handle=11) --multiprocessing-fork
root         12      7   1  12:06 ?        00:00:00 \_ /usr/bin/python3 -c from multiprocessing.spawn import spawn_main; spawn_main(tracker_fd=5, pipe_handle=13) --multiprocessing-fork
root         13      7   1  12:06 ?        00:00:00 \_ /usr/bin/python3 -c from multiprocessing.spawn import spawn_main; spawn_main(tracker_fd=5, pipe_handle=15) --multiprocessing-fork
root@26a0f4b798e8:/sil1# exit
exit
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker rm -f silp1
silp1
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

1 - ¿Cuántos subprocesos de hypercorn aparecen? ¿Por qué?

Aparecen 5 subprocesos hypercorn porque se ha especificado que el número de trabajadores sea 5, ya que cada trabajador se ejecuta como un subproceso independiente. Los subprocesos de hypercorn son los siguientes: root 9, root 10, root 11, root 12, root 13. Además, podemos apreciar como el proceso root 7 es el proceso principal de hypercorn que gestiona los trabajadores.

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

La opción -p se utiliza para mapear el puerto 8080 del sistema anfitrión al puerto 8000 dentro del contenedor. Esto permite acceder a la aplicación que se ejecuta en el puerto 8000 dentro del contenedor desde el puerto 8080 en el sistema anfitrión.

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

Podríamos cambiar el nombre, el puerto y el número de trabajadores.

Pregunta: docker run myimage on ./si1

1 - ¿Qué mandato has ejecutado?

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ mkdir -p si1/volumen
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ touch si1/volumen/ficheroPrueba
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker run -ti -v ${PWD}/si1/volumen:/si1/volumen silp1 ls -al /si1/volumen/ficheroPrue
ba
-rwxrwxrwx 1 1000 1000 0 Oct  8 15:42 /si1/volumen/ficheroPrueba
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker run -d -v volumen_ejercicioo:/si1/volumen silp1
ab7308623406e71bf84ab523345d2313be5c9deb11efde1a4e36712ebab3dd0
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker volume ls
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ docker volume ls
DRIVER          VOLUME NAME
local          volumen_ejercicioo
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

Con la opción -v (usando un volumen de nombre “volumen_ejercicioo”), crea en local el directorio si1. Con el comando “docker run -d -v volumen_ejercicioo:/si1/volumen silp1” arranca otra réplica de nuestra aplicación montando el directorio creado en el directorio del contenedor.

DOCKER REGISTRY

Pregunta: docker registry

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

Hemos usado el comando “docker pull registry:latest”

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ docker pull registry:latest
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
docker.io/library/registry:latest
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
silpl                1.0             703f15931577   5 hours ago    492MB
silpl                latest          703f15931577   5 hours ago    492MB
registry             latest          0ae1560ca86f   5 days ago     25.4MB
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

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

El número de versión de dicha imagen es “latest”, ya que nos hemos descargado la última versión.

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

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ docker run -d --name sil_registry -p 5050:5000 registry:latest
09fc83476737da5114299aa70b9a322b3f97334310b6f83942359ed24937c089
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
09fc83476737  registry:late /entrypoint.sh /etc...  53 seconds ago Up 52 seconds  0.0.0.0:5050->5000/tcp  sil_registry
ab7388623406  silpl         "/bin/sh -c 'hyperco..." 42 minutes ago Up 42 minutes  8000/tcp              objective_shtern
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

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

Escucha por defecto en el puerto 5000 el contenedor del registry

Pregunta: docker tag push

¿Qué ha sucedido?

Al ejecutar los siguientes comandos, estamos etiquetando una imagen de Docker como "localhost:5000/ubuntu:latest" y la almacenamos en un registro local en el puerto 5000 de localhost utilizando el comando "docker push". De esta manera nos permitirá utilizar esta imagen en otros contextos como por ejemplo de despliegue de contenedores en nuestra máquina local o en otros hosts que tengan acceso al registro local de Docker en el puerto especificado.

Pregunta: docker app from local

¿Qué mandatos has usado?

Para generar la imagen de nuestra aplicación hemos usado el siguiente comando:

`"sudo docker build --tag ubuntu ."`

Para subir la imagen a nuestro registro hemos usado el siguiente comando:

`"docker push localhost:5000/ubuntu:latest"`

Para ejecutar un contenedor con nuestra aplicación usando la imagen de nuestro registro:

`"docker run -p 5000:5000 --name si1_registry ubuntu:latest"`

PARTE II

AWS 33

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 mb s3://silp1
make_bucket: silp1
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 cp src/user_rest.py s3://silp1/src/user_rest.py
upload: src/user_rest.py to s3://silp1/src/user_rest.py
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 ls s3://silp1
PRE src/
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 ls s3://silp1/src
PRE src/
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 ls s3://silp1/src/
2023-10-11 09:30:07          3329 user_rest.py
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

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

- *awslocal s3 mb s3://si1p1*: Crea un bucket en “s3://si1p1”, el nombre debe ser único y compatible con DNS.
- *awslocal s3 cp src/user_rest.py s3://si1p1/src/user_rest.py*: Copia el objeto de “cp src/user_rest.py” a “s3://si1p1/src/user_rest.py”
- *awslocal s3 ls s3://si1p1*: Muestra los buckets de “s3://si1p1”
- *awslocal s3 ls s3://si1p1/src*: Muestra los buckets de “s3://si1p1/src”

2. ¿Qué comando podemos usar para descargar el archivo user_rest.py desde el bucket que hemos creado, dándole otro nombre para evitar sobrescribirlo ?

Se puede usar el comando : *awslocal s3 cp src/user_rest.py s3://si1p1/src/user_rest2.py*

AWS LAMBDA

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ LAMBDA_NAME=user_lambda
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ [[ -d build ]] || mkdir build ; \
cd src ; zip ../build/${LAMBDA_NAME}.zip ${LAMBDA_NAME}.py; cd -
adding: user_lambda.py (deflated 46%)
/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0
```

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ echo "create: newlambda"; \
awslocal lambda create-function \
--function-name ${LAMBDA_NAME} \
--runtime python3.9 \
--zip-file fileb://build/${LAMBDA_NAME}.zip \
--handler ${LAMBDA_NAME}.handler \
--role arn:aws:iam::000000000000:role/sil ; \
awslocal lambda create-function-url-config \
--function-name ${LAMBDA_NAME} \
--auth-type NONE ;
create: newlambda
{
  "FunctionName": "user_lambda",
  "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda",
  "Runtime": "python3.9",
  "Role": "arn:aws:iam::000000000000:role/sil",
  "Handler": "user_lambda.handler",
  "CodeSize": 799,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2023-10-11T07:47:49.291075+0000",
  "CodeSha256": "3UmXIH+Nizdrzw1c2Qec2nZhNe9uqh8maj66bivDJz8=",
  "Version": "$LATEST",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "08c51c5f-1226-49f1-bbcf-cdd7eab0f3a6",
  "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:8eeff65f6809a3ce81507fe733fe09b835899b99481ba22fd75b5a7338290ec1"
  }
}
{
  "FunctionUrl": "http://l0uonka8w66nh1hv3fm80lti5ydr5o.lambda-url.us-east-1.localstack.cloud:4566/",
  "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:user_lambda",
  "AuthType": "NONE",
  "CreationTime": "2023-10-11T07:47:51.609409+0000"
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal lambda wait function-active-v2 --function-name ${LAMBDA_NAME}
```

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ 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/sil",
      "Handler": "user_lambda.handler",
      "CodeSize": 799,
      "Description": "",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2023-10-11T07:47:49.291075+0000",
      "CodeSha256": "3UmXIH+Nizdrzw1c2Qec2nZhNe9uqh8maj66bivDJz8=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "7a9da8ed-4d23-424c-8b8b-329105f8d2b9",
      "PackageType": "Zip",
      "Architectures": [
        "x86_64"
      ],
      "EphemeralStorage": {
        "Size": 512
      },
      "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
      }
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ curl -X POST \
$(awslocal lambda get-function-url-config --function-name ${LAMBDA_NAME} |
jq -r '.FunctionUrl') \
-H 'Content-Type: application/json'
```

Pregunta: Lambdas

1. ¿Qué runtime podríamos usar si por ejemplo queremos implementar una función Lambda en JavaScript ?

Habría que cambiar la línea “runtime python3.9” por la línea “runtime nodejs.x” siendo x la versión del mismo.

2. ¿Cómo recibe la función lambda el usuario los datos que hemos enviado con el comando curl?

La función lambda recibe los datos en forma de evento, procesará la solicitud POST y accederá a los datos del cuerpo del JSON.

3. Comprueba que ha funcionado descargando del bucket el fichero creado por la función. ¿Qué comandos has utilizado para ello?

```
$ aws s3 cp s3://nombre-de-tu-bucket/ruta/al/archivo local/archivo-de-destino
```

```
awslocal s3 cp s3://user_lambda/ruta/al/archivo local/descargaBucket
```

Tarea: depuración lamdas

```
larry@DESKTOP-1MR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ docker logs localstack_main
LocalStack version: 2.3.3.dev
LocalStack Docker container id: a9afb89772a9
LocalStack build date: 2023-10-11
LocalStack build git hash: 4ddec69f

2023-10-11T07:26:00.307 INFO --- [-function6] hypercorn.error : Running on https://0.0.0.0:4566 (CTRL + C to quit)
2023-10-11T07:26:00.307 INFO --- [-function6] hypercorn.error : Running on https://0.0.0.0:4566 (CTRL + C to quit)
Ready.
2023-10-11T07:29:37.263 INFO --- [asgi_gw_0] localstack.request.aws : AWS s3.CreateBucket => 200
2023-10-11T07:29:56.438 INFO --- [asgi_gw_0] localstack.request.aws : AWS s3.CreateBucket => 200
2023-10-11T07:30:07.416 INFO --- [asgi_gw_0] localstack.request.aws : AWS s3.PutObject => 200
2023-10-11T07:30:17.488 INFO --- [asgi_gw_0] localstack.request.aws : AWS s3.ListObjectsV2 => 200
2023-10-11T07:30:27.519 INFO --- [asgi_gw_0] localstack.request.aws : AWS s3.ListObjectsV2 => 200
2023-10-11T07:30:37.132 INFO --- [asgi_gw_0] localstack.request.aws : AWS s3.ListObjectsV2 => 200
2023-10-11T07:47:49.298 INFO --- [asgi_gw_0] localstack.request.aws : AWS Lambda.CreateFunction => 201
2023-10-11T07:47:51.609 INFO --- [asgi_gw_0] localstack.request.aws : AWS Lambda.CreateFunctionUrlConfig => 201
2023-10-11T07:48:02.797 INFO --- [asgi_gw_0] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:03.804 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:04.811 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:05.821 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:06.836 INFO --- [asgi_gw_0] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:07.845 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:08.853 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:09.859 INFO --- [asgi_gw_0] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:10.868 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:11.876 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunction => 200
2023-10-11T07:48:21.704 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.ListFunctions => 200
2023-10-11T07:48:36.762 INFO --- [asgi_gw_1] localstack.request.aws : AWS Lambda.GetFunctionUrlConfig => 200
2023-10-11T07:48:37.637 INFO --- [asgi_gw_1] l.u.container_networking : Determined main container network: bridge
2023-10-11T07:48:37.678 INFO --- [asgi_gw_1] l.u.container_networking : Determined main container target IP: 172.17.0.2
2023-10-11T07:48:37.957 INFO --- [asgi_gw_2] localstack.request.aws : AWS s3.CreateBucket => 200
2023-10-11T07:48:37.962 INFO --- [asgi_gw_2] localstack.request.http : POST /_localstack_lambda/db78c72097a29015cce0e4cc7a590b89/invocations/d31d9033-675e-486e-b6ae-2d736a0d11e6/logs => 202
2023-10-11T07:48:37.970 INFO --- [asgi_gw_2] localstack.request.aws : AWS s3.PutObject => 200
2023-10-11T07:48:37.975 INFO --- [asgi_gw_2] localstack.request.http : POST /_localstack_lambda/db78c72097a29015cce0e4cc7a590b89/invocations/d31d9033-675e-486e-b6ae-2d736a0d11e6/response => 202
2023-10-11T07:48:37.977 INFO --- [asgi_gw_2] localstack.request.http : POST /_ / => 200
larry@DESKTOP-1MR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal logs describe-log-groups
{
  "logGroups": [
    {
      "logGroupName": "/aws/lambda/user_lambda",
      "creationTime": 1697010518025,
      "metricFilterCount": 0,
      "arn": "arn:aws:logs:us-east-1:000000000000:log-group:/aws/lambda/user_lambda:*",
      "storedBytes": 1564
    }
  ]
}
```

```
larry@DESKTOP-1MR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal logs describe-log-streams --log-group-name /aws/lambda/${LAMBDA_NAME}
{
  "logStreams": [
    {
      "logStreamName": "2023/10/11/[$LATEST]db78c72097a29015cce0e4cc7a590b89",
      "creationTime": 1697010518032,
      "firstEventTimestamp": 1697010517979,
      "lastEventTimestamp": 1697010518031,
      "lastIngestionTime": 1697010518039,
      "uploadSequenceToken": "1",
      "arn": "arn:aws:logs:us-east-1:000000000000:log-group:/aws/lambda/user_lambda:log-stream:2023/10/11/[$LATEST]db78c72097a29015cce0e4cc7a590b89",
      "storedBytes": 1564
    }
  ]
}
larry@DESKTOP-1MR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal logs describe-log-streams --log-group-name /aws/lambda/${LAMBDA_NAME} |
jq '.logStreams[] | firstEventTimestamp' | sort -n
1697010517979
larry@DESKTOP-1MR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ LLOGSTREAMS=$(awslocal logs describe-log-streams \
--log-group-name /aws/lambda/${LAMBDA_NAME} |
jq -r '.logStreams[] | logStreamName')
```



```

Larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ 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: d31d9033-675e-486e-b6ae-2d736a0d11e6 Version: $LATEST"
"[INFO]t2023-10-11T07:48:37.965Z\t d31d9033-675e-486e-b6ae-2d736a0d11e6\tEscribiendo en S3""Received event: {"
  "version": "2.0",
  "routeKey": "$default",
  "rawPath": "/",
  "rawQueryString": "",
  "headers": {
    "host": "l0uonka8w66nh1hv3fmf80lti5ydpr5o.lambda-url.us-east-1.localstack.cloud:4566",
    "user-agent": "curl/7.81.0",
    "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": "l0uonka8w66nh1hv3fmf80lti5ydpr5o",
    "domainName": "l0uonka8w66nh1hv3fmf80lti5ydpr5o.lambda-url.us-east-1.localstack.cloud:4566",
    "domainPrefix": "l0uonka8w66nh1hv3fmf80lti5ydpr5o",
    "http": {
      "method": "POST",
      "path": "/",
      "protocol": "HTTP/1.1",
      "sourceIp": "",
      "userAgent": "curl/7.81.0"
    },
    "requestId": "356b3536-7580-4dc1-9d81-1c19e489f6fc",
    "routeKey": "$default",
    "stage": "$default",
    "time": "11/Oct/2023:07:48:36 +0000",
    "timeEpoch": 1697010516967
  },
  "body": "{\\\"username\\\": \\\"pepe\\\", \\\"data\\\": \\\"I0\\\"}",
  "isBase64Encoded": false
}"
"END RequestId: d31d9033-675e-486e-b6ae-2d736a0d11e6"
Larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$

```

Pregunta: depuración lambdas

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

- *docker logs localstack_main*: Muestra los logs generados por el contenedor de docker que ejecuta LocalStack, proporciona información sobre el entorno de LocalStack lo que ayuda en la depuración.
- *awslocal logs describe-log-groups*: Devuelve una lista de los grupos de registros disponibles en el entorno de LocalStack. Muestra información acerca de los grupos de registros creados.
- *awslocal logs describe-log-streams --log-group-name /aws/lambda/\${LAMBDA_NAME}*: Devuelve información sobre los registros dentro de un grupo de registros específico. En esta ocasión, devuelve información sobre los flujos de registros relacionados con la función lambda llamada *\${LAMBDA_NAME}*.
- *awslocal logs describe-log-streams --log-group-name /aws/lambda/\${LAMBDA_NAME} | jq '.logStreams[].firstEventTimestamp' | sort -n*: Busca los flujos de registros relacionados con la función llamada *\${LAMBDA_NAME}*, obtiene la marca de tiempo del primer evento y los ordena en orden ascendente. Esto sirve para saber cuando se generaron los últimos registros de cada flujo.

- `LLOGSTREAMS=$(awslocal logs describe-log-streams \ --log-group-name /aws/lambda/${LAMBDA_NAME} | jq -r '.logStreams[].logStreamName')`: Guarda los nombres de los flujos de registros relacionados con la función lambda en LLOGSTREAMS.
- `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'`: Itera por los flujos almacenados en la variable LOGSTREAM almacenados en el comando anterior y en cada flujo utiliza awslocal logs get-log-events para obtener los eventos de registro, con jq extrae y muestra el contenido.

Por lo tanto, los comandos se utilizan para obtener información acerca de los eventos y registros generados por una función Lambda ejecutada por LocalStack, esto es muy útil para depurar y encontrar errores.

Pregunta: Borrado de funciones

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ echo "clean: dellambda" ;
clean: dellambda
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal lambda delete-function-url-config \
--function-name user_lambda ;
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal lambda delete-function --function-name user_lambda
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ echo "clean: md zip" \
awslocal s3 rb --force s3://silpl ; \
rm build/user_lambda.*
clean: md zip awslocal s3 rb --force s3://silpl
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

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

Si reiniciamos LocalStack en vez de borrar las funciones los recursos de las funciones seguirán en el entorno ya que reiniciar no borra ni los buckets ni las funciones. Para evitarlo hay que eliminar las funciones lambda, eliminar buckets y demás recursos, un modo de hacerlo es con los comandos anteriores.

API GATEWAY

Pregunta: Creación del API

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ export REGION="us-east-1" ; export API_NAME="user_lambda"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway create-rest-api --region ${REGION} --name ${API_NAME}
{
  "id": "noih23wply",
  "name": "user_lambda",
  "createdDate": 1697262039.0,
  "apiKeySource": "HEADER",
  "endpointConfiguration": {
    "types": [
      "EDGE"
    ]
  },
  "disableExecuteApiEndpoint": false
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / create-rest-api"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway get-rest-apis
{
  "items": [
    {
      "id": "noih23wply",
      "name": "user_lambda",
      "createdDate": 1697262039.0,
      "apiKeySource": "HEADER",
      "endpointConfiguration": {
        "types": [
          "EDGE"
        ]
      },
      "disableExecuteApiEndpoint": false
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

1. ¿Qué ID se ha asignado al API ?

Tal y como se puede observar, se ha asignado al API el siguiente ID: "id": "noih23wp1y". Este id es único y se utiliza para identificar de manera única el API en la configuración de Amazon API Gateway.

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

Si volvemos a crear otro API con el mismo nombre, AWS API Gateway nos permitirá hacerlo, pero este nuevo API tendrá un id diferente. Esto es debido a que el nombre del API no tiene que ser único a nivel global en Amazon API Gateway, pero el ID sí lo es. Por lo tanto, aunque el nombre del API sea el mismo, cada API tendrá su propio ID único para su identificación y administración en AWS.

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ export REGION="us-east-1" ; export API_NAME="user_lambda"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway create-rest-api --region ${REGION} --name ${API_NAME}
{
  "id": "burodg4owu",
  "name": "user_lambda",
  "createdDate": 1697262448.0,
  "apiKeySource": "HEADER",
  "endpointConfiguration": {
    "types": [
      "EDGE"
    ]
  },
  "disableExecuteApiEndpoint": false
}
```

Pregunta: Creación de las rutas

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ API_ID=$(awslocal apigateway get-rest-apis \
--query "items[?name=='${API_NAME}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ PATH_PART="user" ; PARENT_PATH="/"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ PARENT_RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${PARENT_PATH}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway create-resource \
--region ${REGION} \
--rest-api-id ${API_ID} \
--parent-id ${PARENT_RESOURCE_ID} \
--path-part "${PATH_PART}"
{
  "id": "p12wyek56l",
  "parentId": "07gl4856sm",
  "pathPart": "user",
  "path": "/user"
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ PATH_PART="{username}" ; PARENT_PATH="/user"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ PARENT_RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${PARENT_PATH}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway create-resource \
--region ${REGION} \
--rest-api-id ${API_ID} \
--parent-id ${PARENT_RESOURCE_ID} \
--path-part "${PATH_PART}"
{
  "id": "pgjlajsltg",
  "parentId": "p12wyek56l",
  "pathPart": "{username}",
  "path": "/user/{username}"
}
```

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / create-resource ${PATH_PART}"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "07gl4856sm",
      "path": "/"
    },
    {
      "id": "p12wyek56l",
      "parentId": "07gl4856sm",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "pgjlajsltg",
      "parentId": "p12wyek56l",
      "pathPart": "{username}",
      "path": "/user/{username}"
    }
  ]
}
```

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

El ID que se ha asignado al recurso con la ruta “/user/{username}” es: "id": "pgj1ajs1tq".

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

Si intentamos crear también la ruta “/user/{name}” ocurrirá lo siguiente:

1 - Como ya hemos creado una ruta para /user, nos dará el siguiente error

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ API_ID=$(awslocal apigateway get-rest-apis \
--query "items[?name=='${API_NAME}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ PATH_PART="{name}" ; PARENT_PATH="/user"
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ PARENT_RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${PARENT_PATH}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway create-resource --region ${REGION} --rest-api-id ${API_ID} --par
ent-id ${PARENT_RESOURCE_ID} --path-part "${PATH_PART}"
An error occurred (BadRequestException) when calling the CreateResource operation: A sibling ({name}) of this resource already has a variable path part -- only one is allowed
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

2 - Una vez que borremos la ruta en la que se nos estaba dando ese error, funcionará sin problemas la creación:

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ API_ID=$(awslocal apigateway get-rest-apis \
--query "items[?name=='${API_NAME}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ PATH_PART="{name}" ; PARENT_PATH="/user"
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ PARENT_RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${PARENT_PATH}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway create-resource \
--region ${REGION} \
--rest-api-id ${API_ID} \
--parent-id ${PARENT_RESOURCE_ID} \
--path-part "${PATH_PART}"
{
  "id": "259pj9bk2j",
  "parentId": "p12wyek56l",
  "pathPart": "{name}",
  "path": "/user/{name}"
}
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

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

No, no tiene mucho sentido crear ambas rutas a la vez ya que si ambas presentan recursos muy similares y no hay una diferencia clara entre “name” y “username” en el contexto de nuestra API podrían causar ambigüedades. Los usuarios podrían confundirse sobre qué ruta deben utilizar para qué propósito.

Pregunta: Creación de métodos

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

```
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ PARAM=username ; METHOD_PATH="/user/{username}" ; HTTP_METHOD=GET
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ RESOURCE_ID=$(awslocal apigateway get-resources \
--rest-api-id ${API_ID} --query "items[?path=='${METHOD_PATH}'].id" \
--output text --region ${REGION})
larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ 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": "GET",
  "authorizationType": "NONE",
  "apiKeyRequired": false,
  "requestParameters": {
    "method.request.path.username": true
  }
}
```

```

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / put-method ${METHOD_PATH}"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "b3f1wptyxf",
      "path": "/"
    },
    {
      "id": "r44tmxdofx",
      "parentId": "b3f1wptyxf",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "khxjx8hvre",
      "parentId": "r44tmxdofx",
      "pathPart": "{username}",
      "path": "/user/{username}",
      "resourceMethods": {
        "POST": {
          "httpMethod": "POST",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        },
        "GET": {
          "httpMethod": "GET",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        }
      }
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$

```

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

Podemos verlos con el siguiente comando:

```

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "b3f1wptyxf",
      "path": "/"
    },
    {
      "id": "r44tmxdofx",
      "parentId": "b3f1wptyxf",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "khxjx8hvre",
      "parentId": "r44tmxdofx",
      "pathPart": "{username}",
      "path": "/user/{username}",
      "resourceMethods": {
        "POST": {
          "httpMethod": "POST",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        },
        "GET": {
          "httpMethod": "GET",
          "authorizationType": "NONE",
          "apiKeyRequired": false,
          "requestParameters": {
            "method.request.path.username": true
          },
          "methodResponses": {}
        }
      }
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$

```

Podemos ver como en la ruta “/user/{username}” hay 2 métodos asociados: “POST” y “GET”

Pregunta: Integración

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

```
Larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ HTTP_METHOD=GET ; LAMBDA_NAME=user_lambda
Larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway put-integration --region $(REGION) --rest-api-id ${API_ID} --resource-id ${RESOURCE_ID} --http-method ${HTTP_METHOD} --type AWS_PROXY --integration-method POST --uri arn:aws:apigateway:${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/invocations",
  "passthroughBehavior": "WHEN_NO_MATCH",
  "timeoutInMillis": 29000,
  "cacheNamespace": "khxjx8hvre",
  "cacheKeyParameters": []
}

Larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / put-integration"
Larry@DESKTOP-IMR556L: /mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway get-resources --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "b3f1wptyxf",
      "path": "/"
    },
    {
      "id": "r44tmxdofx",
      "parentId": "b3f1wptyxf",
      "pathPart": "user",
      "path": "/user"
    },
    {
      "id": "khxjx8hvre",
      "parentId": "r44tmxdofx",
      "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/invocations",
            "passthroughBehavior": "WHEN_NO_MATCH",
            "timeoutInMillis": 29000,
            "cacheNamespace": "khxjx8hvre",
            "cacheKeyParameters": []
          }
        },
        "GET": {
          "httpMethod": "GET",
          "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/invocations",
            "passthroughBehavior": "WHEN_NO_MATCH",
            "timeoutInMillis": 29000,
            "cacheNamespace": "khxjx8hvre",
            "cacheKeyParameters": []
          }
        }
      }
    }
  ]
}
```

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

Podemos ver si se ha integrado cada método del API con el método: `awslocal apigateway get-resources --rest-api-id ${API_ID}`.

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

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway delete-integration \
--region ${REGION} \
--rest-api-id ${API_ID} \
--resource-id ${RESOURCE_ID} \
--http-method GET
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / delete-integration"
$: command not found
Failed: AWS / apigateway / delete-integration
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / delete-integration"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

Pregunta: Despliegue

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ STAGE=dev
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway create-deployment \
--region ${REGION} \
--rest-api-id ${API_ID} \
--stage-name ${STAGE}
{
  "id": "hcfts7snff",
  "createdDate": 1697266494.0
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / create-deployment"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway get-deployments --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "hcfts7snff",
      "createdDate": 1697266494.0
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway get-stages --rest-api-id ${API_ID}
{
  "item": [
    {
      "deploymentId": "hcfts7snff",
      "stageName": "dev",
      "cacheClusterEnabled": false,
      "cacheClusterStatus": "NOT_AVAILABLE",
      "methodSettings": {},
      "tracingEnabled": false
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

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

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ STAGE=dev
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway create-deployment \
--region ${REGION} \
--rest-api-id ${API_ID} \
--stage-name ${STAGE}
{
  "id": "ba2fxrot5v",
  "createdDate": 1697266598.0
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ [ $? == 0 ] || echo "Failed: AWS / apigateway / create-deployment"
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway get-deployments --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "hcfts7snff",
      "createdDate": 1697266494.0
    },
    {
      "id": "ba2fxrot5v",
      "createdDate": 1697266598.0
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal apigateway get-stages --rest-api-id ${API_ID}
{
  "item": [
    {
      "deploymentId": "ba2fxrot5v",
      "stageName": "dev",
      "cacheClusterEnabled": false,
      "cacheClusterStatus": "NOT_AVAILABLE",
      "methodSettings": {},
      "tracingEnabled": false
    }
  ]
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

Si repetimos el despliegue se genera un nuevo despliegue con un identificador diferente, lo que no debería a la etapa existente “dev”. La etapa “dev” seguirá existiendo y apuntando al despliegue más reciente.

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?

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway get-deployments --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "hcfts7snff",
      "createdDate": 1697266494.0
    },
    {
      "id": "ba2fxrot5v",
      "createdDate": 1697266598.0
    }
  ]
}

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway delete-deployment --rest-api-id ${API_ID} --deployment-id hcfts7snff
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway get-deployments --rest-api-id ${API_ID}
{
  "items": [
    {
      "id": "ba2fxrot5v",
      "createdDate": 1697266598.0
    }
  ]
}

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway get-stages --rest-api-id ${API_ID}
{
  "item": [
    {
      "deploymentId": "ba2fxrot5v",
      "stageName": "dev",
      "cacheClusterEnabled": false,
      "cacheClusterStatus": "NOT_AVAILABLE",
      "methodSettings": {},
      "tracingEnabled": false
    }
  ]
}

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

Al haber creado 2 dev, con los comandos anteriores hemos eliminado uno de ellos, para repetir el proceso, repetimos el primer comando de la segunda foto (el resto son para la comprobación de que se ha borrado de manera exitosa).

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway update-stage --rest-api-id ${API_ID} --stage-name dev --patch-operations op=replace,path=/deploymentId,value=nuevo_despliegue_id
{
  "deploymentId": "nuevo_despliegue_id",
  "stageName": "dev",
  "description": "",
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "methodSettings": {},
  "tracingEnabled": false
}

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway delete-stage --rest-api-id ${API_ID} --stage-name dev
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal apigateway get-stages --rest-api-id ${API_ID}
{
  "item": []
}

larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

Los pasos que hemos seguido para borrar el despliegue que no está asociado a la etapa “dev”.

1. Identificar la etapa “dev” en la lista y verificar a cuál de los despliegues está apuntando.
2. Si la etapa “dev” está apuntando al despliegue que queremos eliminar, hay 2 maneras de continuar: cambiarla para que apunte a otro despliegue o eliminar la etapa “dev” si no es necesaria.
3. Finalmente, ejecutamos el segundo comando de la última imagen (la tercera) y con ello conseguimos eliminar el despliegue.

Pregunta: Pruebas del API

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ STAGE=dev
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ ENDPOINT=http://localhost:4566/restapis/${API_ID}/${STAGE}/_user_request/_user
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ curl -X POST ${ENDPOINT}/pepe \
-H 'Content-Type: application/json' \
-d '{"username": "tete", "data": "100"}'
"f{test_object_key} placed into S3"larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

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?

```
(silp1) larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 ls
2023-10-17 12:32:16 silp1
(silp1) larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 ls s3://silp1
PRE src/
PRE users/
```

```
(silp1) larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ awslocal s3 cp s3://silp1/users/pepe pepe.txt
download: s3://silp1/users/pepe to ./pepe.txt
(silp1) larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$ cat pepe.txt
{"username": "pepe", "data": "10"}(silp1) larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/p1-v1.0$
```

Tal y como podemos observar, se ha actualizado son los nuevos datos insertados en la petición HTTP POST.

Pregunta: Mejora del API

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

```
def handler(event, context):
    LOGGER.info('Escribiendo en S3')
    print("Received event: " + json.dumps(event, indent=2))

    #Obtenemos el param username
    username = event['pathParameters']['username']
    #Obteneos datos del cuerpo del evento
    data = json.loads(event.get('body', '{}'))

    Clave_objeto = f"user/{username}/data.json"

    S3_CLIENT.put_object(
        Bucket=BUCKET_NAME,
        Key=Clave_objeto,
        Body=json.dumps(data)
    )

    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
```

Hemos añadido las siguientes líneas al archivo `user_lambda_param.py`:

```
#Obtenemos el param username
username = event['pathParameters']['username']
#Obteneos datos del cuerpo del evento
data = json.loads(event.get('body','{}'))

clave_objeto = f"user/{username}/data.json"
```

Y hemos eliminado las siguientes 2 líneas:

```
req=json.loads(event['body'])
test_object_key = f"users/{req['username']}"
```

DYNAMODB

Pregunta: Creación de una tabla

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

```
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas
/p1-v1.0$ awslocal dynamodb create-table \
--table-name silp1 \
--attribute-definitions AttributeName=User,AttributeType=S \
--key-schema AttributeName=User,KeyType=HASH \
--region $REGION \
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "User",
        "AttributeType": "S"
      }
    ],
    "TableName": "silp1",
    "KeySchema": [
      {
        "AttributeName": "User",
        "KeyType": "HASH"
      }
    ],
    "TableStatus": "ACTIVE",
    "CreationDateTime": 1697536576.659,
    "ProvisionedThroughput": {
      "ReadCapacityUnits": 5,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-east-1:000000000000:table/silp1",
    "TableId": "73e17328-76ea-43db-b3fe-65ccf4bd079a"
  }
}
larry@DESKTOP-IMR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas
/p1-v1.0$
```

El único campo de la tabla es User es un campo de tipo String o “S”.

Pregunta: Insertando datos manualmente

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

Si intentamos crear un ítem sin campo “User” o sin campo “Email” habrá un error, esto es porque User es la clave HASH de la tabla por lo que es obligatorio tener el campo.

Si intentamos crear un ítem sin campo “Email” no habrá errores ya que no es obligatorio por no ser una primary key o clave principal, lo que sucederá es que el ítem no tendrá campo Email.

Pregunta: Lectura de datos

```
larry@DESKTOP-INR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal dynamodb get-item \
--table-name silp1 \
--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
larry@DESKTOP-INR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$ awslocal dynamodb scan --table-name silp1 --region us-east-1
{
  "Items": [
    {
      "Email": {
        "S": "pipo@ss.com"
      },
      "User": {
        "S": "pipo"
      }
    }
  ],
  "Count": 1,
  "ScannedCount": 1,
  "ConsumedCapacity": null
}
larry@DESKTOP-INR556L:/mnt/c/Users/Sergio/Documents/3_curso/Sistemas_Informaticos/Practicas/pl-v1.0$
```

2. ¿Qué ocurre si creamos un ítem sin campo “User”?

Como hemos explicado anteriormente no se puede por ser una primary key.

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

El ítem de la tabla será cambiado por el nuevo ítem, esto sucederá porque DynamoDB sobrescribe los ítems cuando se hace uno nuevo con la misma clave principal.

Pregunta: Insertando datos manualmente

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

- 1 - Hacer una nueva función Lambda con el código necesario para interactuar con DynamoDB.
- 2 - Cambiar la implementación del método POST de “user/{username}” para que apunte a la nueva función lambda.
- 3 - Desplegar la API en API Gateway para que los cambios sean efectivos.

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

Se han añadido los siguientes datos a la tabla si1p1 tras ejecutar el comando curl:

- Un registro con “User” como “user2”.
- Un registro con “Email” como “mimail@dominio.es”

Pregunta: GET /user/{username} con dynamoDB

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

- 1 - Creamos la función Lambda, igual que user_lambda pero con: LAMBDA_NAME = user_lambda_dynamo.
- 2 - Creamos la API asociada a la función lambda y la llamamos user_lambda_dynamo.
- 3 - Creamos las rutas asociadas a la API.
- 4 - Creamos los métodos asociados a la API.
- 5 - Integramos el metodo POST en user_lambda_dynamo.

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

Podemos usar el comando: `curl -X GET ${ENDPOINT}/user2 -d '{"email": "mimail@dominio.es"}'`

Si el usuario existe la respuesta es un objeto JSON con la información del usuario. Si el usuario no existe, la respuesta es un mensaje que nos informa que el usuario no fue encontrado.