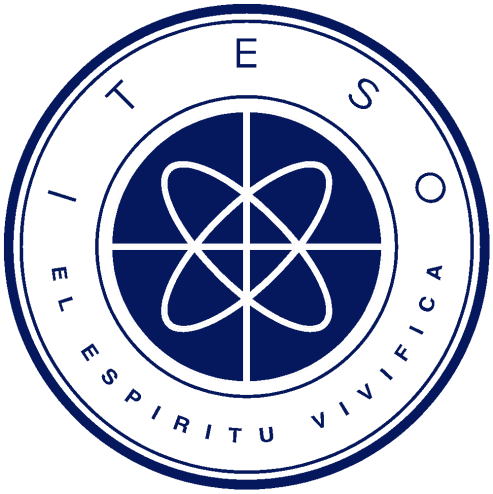
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

CÓMPUTO EN LA NUBE



REPORTE DE PRACTICA DE LABORATORIO

“Práctica 2”

I. Introducción

El objetivo es poner en práctica la automatización de la nube utilizando esquemas serverless.

Cuando una imagen se coloca en un bucket de S3, una función serverless actúa para enviar la imagen a un cluster de Docker y que la imagen sea cambiada de tamaño mediante una aplicación hecha en python.

Además de todo esto, las imágenes tratadas en el cluster, se pueden descargar y cargar directamente en S3.

II. Marco Teórico

A continuación se indican los componentes investigados que afectaron en el proceso de desarrollo de la práctica.

Docker: Docker proporciona una manera estándar de ejecutar código. Docker es un sistema operativo para contenedores. De manera similar a cómo una máquina virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores.[1]

ECS: Amazon Elastic Container Service (Amazon ECS) es un servicio de organización de contenedores de alta escalabilidad y rendimiento compatible con los contenedores Docker que le permite ejecutar y ajustar la escala de aplicaciones en contenedores en AWS con facilidad. [4]

S3: Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos,

seguridad y rendimiento líderes en el sector. Esto significa que se puede utilizar para almacenar y proteger cualquier cantidad de datos para diversos casos de uso, como sitios web, aplicaciones móviles, procesos de copia de seguridad y restauración, operaciones de archivado, aplicaciones empresariales, dispositivos IoT y análisis de big data. Amazon S3 proporciona características de administración fáciles de utilizar que le permiten organizar los datos y configurar sofisticados controles de acceso con objeto de satisfacer sus requisitos empresariales, organizativos y de conformidad. [2]

Lambda: AWS Lambda es un servicio de informática sin servidor que ejecuta código en respuesta a eventos y administra automáticamente los recursos informáticos subyacentes. Se puede usar Lambda para ampliar la funcionalidad de otros productos de AWS con lógica personalizada o bien crear servicios back-end propios que funcionen con el nivel de seguridad, rendimiento y escala de AWS. AWS Lambda puede ejecutar código automáticamente en respuesta a varios eventos, como solicitudes HTTP a través de Amazon API Gateway, modificaciones realizadas en objetos en buckets de Amazon S3, actualizaciones de tablas en Amazon DynamoDB y transiciones de estado en AWS Step Functions.[3]

Boto3: El kit de desarrollo de software (SDK) de Amazon Web Services (AWS) para Python, que permite escribir software que haga uso de servicios como Amazon S3 y Amazon EC2. [5]

Fargate: AWS Fargate es un motor informático para Amazon ECS que le permite ejecutar contenedores sin tener que administrar servidores o clústeres, se elimina la necesidad de elegir tipos de servidores, decidir cuándo escalar los clústeres u optimizar conjuntos de clústeres. Fargate permite concentrarse en el diseño y la creación de aplicaciones en lugar de centrarse en la administración de la infraestructura que las ejecuta.[6]

[1]"Contenedores de Docker | ¿Qué es Docker? | AWS", *Amazon Web Services, Inc.*, 2019. [Online]. Available: https://aws.amazon.com/es/docker/. [Accessed: 09- Apr- 2019].

[2]"Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service", *Amazon Web Services, Inc.*, 2019. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 09- Apr- 2019].

[3]"¿Qué es AWS Lambda? - AWS Lambda", *Docs.aws.amazon.com*, 2019. [Online]. Available: https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html. [Accessed: 09- Apr- 2019].

[4]"AWS | Gestión de contenedores (ECS) compatible con los de Docker", *Amazon Web Services, Inc.*, 2019. [Online]. Available: https://aws.amazon.com/es/ecs/. [Accessed: 09- Apr- 2019].

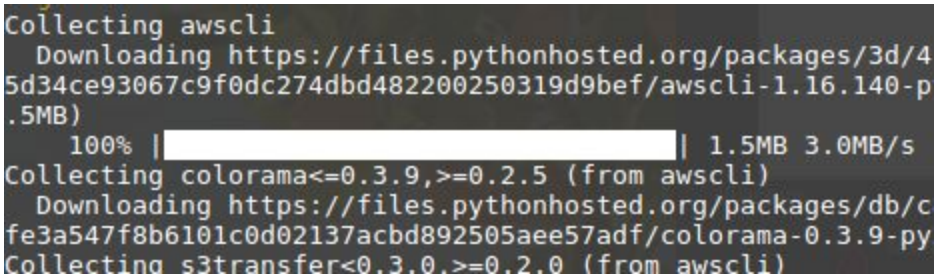
[5]"AWS SDK para Python", *Amazon Web Services, Inc.*, 2019. [Online]. Available: https://aws.amazon.com/es/sdk-for-python/. [Accessed: 09- Apr- 2019].

[6]"AWS Fargate - Run containers without having to manage servers or clusters", *Amazon Web Services, Inc.*, 2019. [Online]. Available: https://aws.amazon.com/fargate/. [Accessed: 09- Apr- 2019].

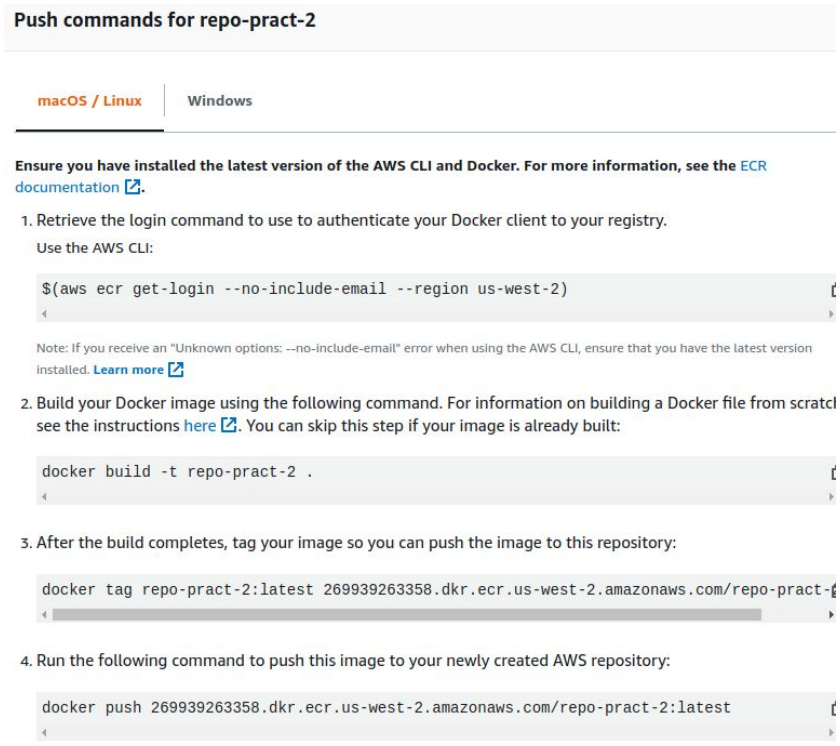
III. Desarrollo de la Práctica.

Para empezar cree el bucket para guardar las imágenes que se iban a subir y a hacer resize, además de dar permisos a mi usuario para tener acceso a S3, ECS, Task Execution.

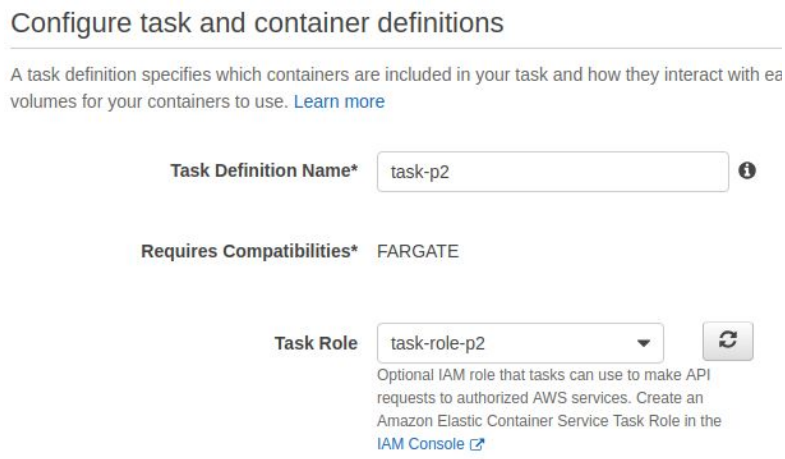
Después de esto procedí con la Instalación de aws cli para hacer push al docker.



Paso seguido de crear el repositorio para hacer push al docker.



Después de hacer push al docker , cree un clúster y una tarea



Además de esto, cree un nuevo rol con permisos para ejecutar función lambda con trigger de S3, dicha función se muestra aquí:

```python
import boto3
import json
import urllib.parse

s3 = boto3.client('s3')
client = boto3.client('ecs')

def lambda_handler(event, context):
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
    bucket = event['Records'][0]['s3']['bucket']['name']

    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        if not key.startswith( 'SMALLER' ):
            response = client.run_task(
                cluster='cluster-p2',
                taskDefinition='task-p2:1',
                overrides={
                    'containerOverrides': [
                        {
                            'name': 'container-p2-ex',
                            'command': [
                                '/app/resize.sh',
                                str(key)
                            ]
                        },
                    ]
                },

                launchType='FARGATE',
                networkConfiguration={
                    'awsvpcConfiguration': {
                        'subnets': [
                            'subnet-0674be26b3bd048d4',
                        ],
                        'assignPublicIp': 'ENABLED'
                    }
                }
            )
            return 0
        else:
            return -1
    except Exception as e:
        raise e
```

Levanta una tarea en el cluster con el ejecutable resize.sh que hace el resize de las imágenes.

Para evitar el ciclo de resize con cualquier imagen, se hace una validación del nombre de la imagen generada, un prefijo que se agregó “SMALLER”

IV. Problemas y Soluciones

A continuación se describen algunos de los problemas que surgieron en el desarrollo de la práctica.

El primer problema que me surgió en el desarrollo de la práctica se mostró en la parte de probar el contenedor y el script de python para hacer el resize de imágenes, dicho problema se soluciono modificando los volúmenes y las rutas de acuerdo al nombre de mis carpetas.

Otro problema encontrado en el desarrollo de la práctica fue la conexión del bucket S3 con una lambda para hacer el resize de imágenes lo que se soluciono despues de proporcionar permisos de acceso a la lambda para el recurso .

Además de esos problemas cabe recalcar que al guardar la imagen en S3, se activaba el trigger para hacer el proceso del diagrama siguiente:

Por lo que para solucionar dicho problema tuve que validar acerca de las imágenes existentes y las nuevas generadas, para que estas no se siguieran metiendo en el proceso de cambio de tamaño.

V. Experimentos y Resultados.

Como experimentos, utilice de manera manual un servicio que ejecutaba una tarea directamente sobre el S3 para hacer la validación y proceder con la función serverless, así me di cuenta cuando no funcionaba de que modificar los componentes, para ir adaptando la función serverless.

VI. Conclusiones

La importancia de la nube ha ido adquiriendo fuerza y relevancia a medida que las personas han tomado conciencia de la necesidad de guardar datos, compartir contenido digital de forma veloz, y al tiempo que los soportes físicos han sido menos accesible o limitados en sus capacidades de manejo de archivos.

La facilidad que proporcionan las funciones serverless para ejecutar tareas suele convertir a las funciones lambda en una herramienta poderosa para interconectar recursos de AWS.

El procesamiento y velocidad de la nube para ejecutar tareas es muchas veces una opción redituable en cuanto a costos dependiendo de las necesidades de quien lo requiera ya que en este caso de la práctica, en lugar de hacer el procesamiento de reducción de imágenes de manera local, mediante servicios de aws resulta más cómodo en caso de querer utilizar dicha funcionalidad desde cualquier dispositivo con acceso a internet.