Repaso de computación en la Nube y servicios AWS

Temas: Servicios de cómputo en AWS

Objetivos

El conocimiento profundo de EC2, servicios de cómputo adicionales, ECS y EKS proporciona un conjunto completo de herramientas para manejar cualquier desafío de computación que los desarrolladores puedan enfrentar, desde la infraestructura básica hasta la orquestación avanzada de contenedores.

Los objetivos de este repaso son profundizar estos temas en tres niveles: desde el punto de vista teórico, dsde el punto de vista páctico usando las herramientas de AWS y por codificación a través de conceptos rudimentarios de python.

Tareas

Prepara una presentación grupal para responder las siguientes preguntas y utiliza un ide de tu preferencia para las implementaciones solicitadas:

Preguntas

1. Introducción a Amazon EC2 y AMIs

- **Pregunta:** Explica el proceso de creación y lanzamiento de una instancia EC2 desde una Amazon Machine Image (AMI). ¿Cuáles son las diferencias entre una AMI pública y una AMI privada?
- **Ejercicio:** Diseña un script en Python usando Boto3 que liste todas las AMIs disponibles en tu cuenta y filtre las AMIs por su nombre y fecha de creación.

2. Explorando de tipos de instancias EC2

- **Pregunta:** Compara y contrasta los diferentes tipos de instancias EC2 (t2.micro, m5.large, p3.2xlarge, etc.). ¿En qué escenarios es más apropiado utilizar cada tipo de instancia?
- **Ejercicio:** Crea un script en Python que recorra y liste todos los tipos de instancias EC2 disponibles en una región específica y muestre sus características clave (CPU, memoria, almacenamiento, red).

3. Amazon EBS y instance store volumes

- **Pregunta:** Describe las diferencias entre Amazon Elastic Block Store (EBS) y los volúmenes de almacenamiento de instancias EC2. ¿Cuándo se debería usar EBS en lugar de almacenamiento de instancia y viceversa?
- **Ejercicio:** Implementa un script en Python que cree un volumen EBS, lo adjunte a una instancia EC2 en ejecución y realice una instantánea de dicho volumen.

4. Opciones de precios EC2

- **Pregunta:** Explica las diferencias entre las opciones de precios de instancias EC2: On-Demand, Reserved y Spot. ¿Cuáles son las ventajas y desventajas de cada una?
- **Ejercicio:** Diseña un simulador en Python que calcule los costos de ejecutar una instancia EC2 utilizando cada una de las opciones de precios para un periodo determinado (por ejemplo, 6 meses). Incluye variaciones en el uso de CPU y almacenamiento.

5. Shared File Storage con Amazon EFS

- **Pregunta:** ¿Qué es Amazon Elastic File System (EFS) y cómo se diferencia de Amazon EBS? Describe un escenario de uso típico para EFS.
- **Ejercicio:** Escribe un script en Python que configure un sistema de archivos EFS, lo monte en varias instancias EC2, y verifique la conectividad y consistencia de los datos entre las instancias.

6. VPS con Amazon Lightsail

- **Pregunta:** ¿Qué es Amazon Lightsail y cómo se compara con Amazon EC2 en términos de funcionalidad y uso? Describe una situación en la que sería más ventajoso usar Lightsail sobre EC2.
- **Ejercicio:** Implementa un script en Python que cree una instancia de Lightsail, configure su red y lance una aplicación web básica en dicha instancia.

7. Amazon ECS y Kubernetes

- **Pregunta:** Compara las dos opciones de despliegue de Amazon ECS (EC2 launch type y Fargate launch type). ¿Cuáles son las ventajas de utilizar Amazon EKS para la orquestación de contenedores?
- **Ejercicio:** Diseña un script en Python que implemente una aplicación simple utilizando Amazon ECS con el tipo de lanzamiento EC2. Configura los contenedores y el servicio para que escale automáticamente según la carga.

8. Servicios de cálculo adicionales con AWS

- **Pregunta:** Compara AWS Lambda, AWS Batch, y AWS Outposts en términos de casos de uso y beneficios. ¿Cuándo sería más apropiado usar cada servicio?
- **Ejercicio:** Implementa una función Lambda que procese eventos de S3 (por ejemplo, cuando se sube un archivo) y escriba los resultados en una base de datos DynamoDB.

9. Opciona de almacenamiento adicional en AWS

- **Pregunta:** Describe las características y casos de uso de Amazon FSx for Lustre y Amazon FSx for Windows File Server. ¿Cómo se diferencian de Amazon EFS y EBS?
- **Ejercicio:** Implementa un script en Python que configure Amazon FSx for Lustre, lo monte en una instancia EC2 y ejecute un trabajo de procesamiento intensivo de datos.

10. Asegurando tu VPC con Bastion Hosts

- **Pregunta:** Explica el concepto de un bastion host y cómo se usa para asegurar el acceso a instancias en una VPC privada. ¿Cuáles son las mejores prácticas para configurar un bastion host?
- **Ejercicio:** Escribe un script en Python que cree una VPC con subredes públicas y privadas, configure un bastion host en la subred pública, y asegure el acceso SSH a las instancias en la subred privada a través del bastion host.

Ejercicios prácticos en Python:

1. Gestión de AMIs

Ejercicio: Implementa un script en Python usando Boto3 que realice las siguientes tareas:

- Lista todas las AMIs disponibles en tu cuenta.
- Filtra las AMIs por un criterio específico (por ejemplo, nombre o fecha de creación).
- Crea una nueva AMI a partir de una instancia EC2 en ejecución.
- Comparte una AMI específica con otra cuenta de AWS.

2. Gestión de volúmenes EBS

Ejercicio: Implementa un script en Python que realice las siguientes tareas:

- Crea un volumen EBS.
- Adjunta el volumen a una instancia EC2 en ejecución.
- Realiza una instantánea del volumen EBS

- Lista todas las instantáneas disponibles en tu cuenta.

Código base:

```
import boto3

ec2 = boto3.client('ec2')

# Crear un volumen EBS
def create_ebs_volume(size, availability_zone):
    response = ec2.create_volume(Size=size, AvailabilityZone=availability_zone, VolumeType='gp2')
    volume_id = response['VolumeId']
    print(f"Created EBS Volume: {volume_id}")
    return volume_id

# Adjuntar un volumen EBS a una instancia EC2
def attach_ebs_volume(volume_id, instance_id, device):
    ec2.attach_volume(VolumeId=volume_id, InstanceId=instance_id, Device=device)
    print(f"Attached Volume {volume_id} to Instance {instance_id}")

# Realizar una instantánea de un volumen EBS
def create_snapshot(volume_id, description):
    response = ec2.create_snapshot(VolumeId=volume_id, Description=description)
    snapshot_id = response['SnapshotId']
    print(f"Created Snapshot: {snapshot_id}")
    return snapshot_id

# Listar todas las instantáneas disponibles
def list_snapshots():
    response = ec2.describe_snapshots(OwnerIds=['self'])
    for snapshot in response['Snapshots']:
        print(f"ID: {snapshot['SnapshotId']}, Volume ID: {snapshot['VolumeId']}, Description:
{snapshot['Description']}")

# Ejecución de funciones
volume_id = create_ebs_volume(10, 'us-west-2a')
attach_ebs_volume(volume_id, 'i-1234567890abcdef0', '/dev/sdf')
create_snapshot(volume_id, 'Snapshot of volume ' + volume_id)
list_snapshots()
```

3. Gestión de políticas de precios EC2

Ejercicio: Implementa un simulador en Python que calcule los costos de ejecutar una instancia EC2 utilizando las opciones de precios On-Demand, Reserved y Spot para un periodo de 6 meses. Incluye variaciones en el uso de CPU y almacenamiento.

Usando AWS Lab Learner

Ejercicio 1: Configuración y gestión de Amazon EC2 y AMIs

Objetivo: Crear, gestionar y lanzar instancias EC2 utilizando Amazon Machine Images (AMIs) y explorar diferentes tipos de instancias.

Pasos:

Crear una instancia EC2:

- Utiliza el AWS Management Console para lanzar una nueva instancia EC2 utilizando una AMI existente.

Explorar tipos de instancias:

- Cambia el tipo de instancia de una instancia EC2 en ejecución.
- Documenta las diferencias de rendimiento entre diferentes tipos de instancias (t2.micro, m5.large, p3.2xlarge).

Crear una AMI:

- Configura una instancia EC2 con un servidor web básico.
- Crea una AMI a partir de esta instancia.

Lanzar instancias desde una AMI:

- Utiliza la AMI creada para lanzar nuevas instancias EC2.
- Verifica que el servidor web esté funcionando en las nuevas instancias.

Validación:

- Asegúrate de que las nuevas instancias lanzadas desde la AMI funcionen correctamente.
- Documenta las diferencias observadas en el rendimiento entre los diferentes tipos de instancias.

Ejercicio 2: Configuración de Amazon EBS y almacenamiento de instancias

Objetivo: Crear y gestionar volúmenes de Amazon EBS y volúmenes de almacenamiento de instancias, y entender sus diferencias.

Pasos:

Crear un volumen EBS:

- Utiliza el AWS Management Console para crear un volumen EBS.
- Adjunta el volumen a una instancia EC2 en ejecución.

Gestionar volúmenes de almacenamiento de instancias:

- Configura una instancia EC2 con volúmenes de almacenamiento de instancias.
- Comprueba las diferencias de rendimiento entre EBS y el almacenamiento de instancias.

Realizar instantáneas de EBS:

- Crea una instantánea de un volumen EBS en uso.
- Restaura un volumen EBS desde una instantánea y adjúntalo a una instancia EC2.

Validación:

- Verifica que los volúmenes EBS y de almacenamiento de instancias estén correctamente configurados y en uso.
- Comprueba que las instantáneas se crean y restauran correctamente.

Ejercicio 3: Configuración de Amazon EFS y montaje en instancias EC2

Objetivo: Configurar Amazon Elastic File System (EFS) y montarlo en varias instancias EC2.

Pasos:

Crear un Sistema de Archivos EFS:

- Utiliza el AWS Management Console para crear un sistema de archivos EFS.

Configurar puntos de montaje:

- Configura puntos de montaje en varias instancias EC2.
- Monta el sistema de archivos EFS en cada instancia.

Probar la consistencia de datos:

- Crea y modifica archivos en el sistema de archivos EFS desde una instancia.
- Verifica que los cambios sean visibles y consistentes en todas las instancias.

Validación:

- Asegúrate de que el sistema de archivos EFS esté correctamente montado en todas las instancias.
- Verifica la consistencia de los datos entre las instancias.

Ejercicio 4: Exploración de opciones de precios EC2

Objetivo: Explorar las diferentes opciones de precios de instancias EC2 (On-Demand, Reserved, Spot) y entender sus costos y beneficios.

Pasos:

Crear instancias On-Demand:

- Utiliza el AWS Management Console para lanzar una instancia EC2 On-Demand.
- Monitorea los costos durante un periodo de tiempo corto (por ejemplo, 1 día).

Reservar instancias:

- Reserva una instancia EC2 por un periodo de 1 año.
- Documenta los costos y beneficios en comparación con las instancias On-Demand.

Utilizar instancias Spot:

- Lanza una instancia EC2 Spot.
- Monitorea los costos y la disponibilidad de las instancias Spot.

Validación:

- Compara los costos y beneficios de las diferentes opciones de precios.
- Documenta los escenarios en los que cada opción de precio sería más ventajosa.

Ejercicio 5: Implementación de almacenamiento compartido con Amazon EFS

Objetivo: Configurar un sistema de archivos EFS y montarlo en varias instancias EC2, asegurando el acceso compartido y la consistencia de datos.

Pasos:

Crear y configurar Amazon EFS:

- Utiliza el AWS Management Console para crear un sistema de archivos EFS.
- Configura puntos de acceso y políticas de montado.

Montar EFS en instancias EC2:

- Lanza varias instancias EC2.
- Monta el sistema de archivos EFS en cada instancia.

Verificar consistencia de datos:

- Crea archivos en el sistema de archivos EFS desde una instancia.
- Verifica que los archivos sean accesibles y modificables desde todas las instancias.

Validación:

- Verifica que el sistema de archivos EFS esté correctamente montado y accesible desde todas las instancias.
- Asegúrate de que los datos sean consistentes entre todas las instancias.

Ejercicio 6: Configuración de una arquitectura de contenedores con Amazon ECS y Kubernetes

Objetivo: Configurar una aplicación simple utilizando Amazon ECS y explorar la integración con Kubernetes (Amazon EKS).

Pasos:

Configurar Amazon ECS:

- Crea un clúster de Amazon ECS.
- Define una tarea y servicio para una aplicación simple (por ejemplo, una aplicación web).

Escalar automáticamente el servicio ECS:

- Configura políticas de escalado automático basadas en la carga.

Integra con Amazon EKS:

- Configura un clúster de Amazon EKS.
- Despliega la misma aplicación utilizando Kubernetes.

Validación:

- Verifica que la aplicación esté desplegada y funcionando en Amazon ECS.
- Asegúrate de que el escalado automático funcione correctamente.
- Comprueba que la aplicación esté desplegada y funcionando en Amazon EKS.

Ejercicio 7: Implementación de funciones lambda para procesamiento de eventos

Objetivo: Implementar una función Lambda que procese eventos de S3 y escriba resultados en una base de datos DynamoDB.

Pasos:

Configurar un bucket de S3:

- Crea un bucket de S3 para almacenar archivos de eventos.

Crear una función Lambda:

- Define una función Lambda que procese archivos subidos al bucket de S3.

Configurar rrigger de S3:

- Configura el bucket de S3 para que dispare la función Lambda en cada carga de archivo.

Escribir resultados en DynamoDB:

- Configura la función Lambda para escribir resultados en una tabla DynamoDB.

Validación:

- Verifica que la función Lambda se ejecute correctamente en respuesta a los eventos de S3.
- Asegúrate de que los resultados se escriban correctamente en DynamoDB.

Ejercicio 8: Configuración de amazon FSx para Lustre

Objetivo: Configurar Amazon FSx for Lustre y montar el sistema de archivos en una instancia EC2 para ejecutar un trabajo de procesamiento intensivo de datos.

Pasos:

Configurar Amazon FSx for Lustre:

- Utiliza el AWS Management Console para crear un sistema de archivos FSx for Lustre.

Montar FSx en Instancia EC2:

- Lanza una instancia EC2 y monta el sistema de archivos FSx for Lustre.

Ejecutar trabajo de procesamiento de datos:

- Ejecuta un trabajo de procesamiento intensivo de datos (por ejemplo, análisis de big data) utilizando el sistema de archivos FSx for Lustre.

Validación:

- Verifica que el sistema de archivos FSx for Lustre esté montado correctamente.

- Asegúrate de que el trabajo de procesamiento se ejecute correctamente y utilice el sistema de archivos FSx for Lustre.

Código

Ejercicio 1: Simulación completa de Amazon EC2 y gestión de AMIs

Objetivo: Diseñar y simular un sistema completo para la creación, gestión y despliegue de instancias EC2 utilizando AMIs. Este sistema debe permitir a los usuarios crear AMIs a partir de instancias existentes, listar todas las AMIs disponibles y lanzar nuevas instancias a partir de una AMI seleccionada.

Instrucciones:

Diseñar clases para AMI y EC2Instance:

- Crea clases que representen una AMI y una instancia EC2.
- Implementa métodos para gestionar el estado de la instancia (start, stop, terminate).

Gestión de AMIs:

- Implementa funciones para crear una AMI a partir de una instancia EC2.
- Lista todas las AMIs disponibles y permitir la selección de una AMI para lanzar nuevas instancias.

Despliegue de Instancias:

- Implementa funciones para lanzar nuevas instancias EC2 utilizando una AMI seleccionada.
- Mantiene un registro de todas las instancias lanzadas, con sus estados y AMIs asociadas.

Simulación de Uso:

- Diseña una interfaz de usuario (CLI o interfaz gráfica) que permita a los usuarios interactuar con el sistema.
- Permite la creación de AMIs, el despliegue de instancias, la gestión de estados de las instancias y la visualización del estado actual del sistema.

Ejercicio 2: Implementación y gestión de almacenamiento EBS y volúmenes de instancia

Objetivo: Crear un sistema para gestionar volúmenes de almacenamiento EBS y volúmenes de instancia, simulando la creación, adjunción y gestión de estos volúmenes en instancias EC2.

Instrucciones:

Diseñar clases para EBSVolume y InstanceStoreVolume:

- Crea clases que representen un volumen EBS y un volumen de almacenamiento de instancia.
- Implementa métodos para adjuntar y desadjuntar volúmenes a instancias EC2.

Gestión de volúmenes:

- Implementa funciones para crear y eliminar volúmenes EBS.
- Simula el almacenamiento de datos en volúmenes y gestionar la integridad de los datos durante las operaciones de adjunción y desadjunción.

Integración con instancias EC2:

- Crea funciones que permitan adjuntar volúmenes a instancias EC2 y gestionar su estado.
- Mantiene un registro de todos los volúmenes creados, sus estados y las instancias a las que están adjuntos.

Simulación de escenarios:

- Diseña una interfaz de usuario que permita a los usuarios crear, gestionar y monitorizar volúmenes EBS y volúmenes de instancia.
- Implementa una funcionalidad para simular fallos en volúmenes y evaluar la recuperación de datos.

Ejercicio 3: Simulación de opciones de precios de EC2 con análisis de costos

Objetivo: Diseña un simulador que calcule los costos de diferentes opciones de precios de instancias EC2 (On-Demand, Reserved, Spot) y permita a los usuarios comparar los costos y beneficios de cada opción para diferentes escenarios de uso.

Instrucciones:

Diseña clases para EC2Instance y PricingOption:

- Crea clases que representen una instancia EC2 y una opción de precios (On-Demand, Reserved, Spot).
- Implementa métodos para calcular costos basados en horas de uso y tipo de instancia.

Simulación de costos:

- Implementa funciones para simular el costo de ejecutar instancias EC2 utilizando diferentes opciones de precios.
- Incluye variaciones en el uso de CPU, almacenamiento y tiempo de ejecución para obtener un análisis detallado de los costos.

Comparación de opciones de precios:

- Crea funciones que permitan a los usuarios comparar los costos de diferentes opciones de precios para un periodo determinado.
- Implementa gráficos y tablas para visualizar los resultados de la comparación de costos.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios ingresar datos de uso y seleccionar diferentes opciones de precios para ver los costos estimados.
- Implementa casos de estudio que simulen escenarios de uso real y muestren cómo se afectan los costos con diferentes opciones de precios.

Ejercicio 4: Implementación de almacenamiento compartido con Amazon EFS

Objetivo: Crear un sistema de archivos compartido similar a Amazon EFS y simular su montaje en varias instancias EC2, asegurando la consistencia y disponibilidad de los datos.

Instrucciones:

Diseña clases para EFS y EC2Instance:

- Crea una clase que represente un sistema de archivos EFS.
- Implementa métodos para montar y desmontar el sistema de archivos en instancias EC2.

Gestión de datos en EFS:

- Implementa funciones para crear, leer, escribir y eliminar archivos en el sistema de archivos EFS.
- Asegura la consistencia y disponibilidad de los datos en todas las instancias montadas.

Integración con instancias EC2:

- Crea funciones que permitan montar y desmontar el sistema de archivos EFS en varias instancias EC2.
- Implementa una funcionalidad para verificar la consistencia de los datos entre todas las instancias montadas.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios gestionar el sistema de archivos EFS y monitorizar su estado.
- Implementa escenarios que simulen el uso de EFS en aplicaciones reales y evalúen su rendimiento y disponibilidad.

Ejercicio 5: Simulación de orquestación de contenedores con Amazon ECS y Kubernetes

Objetivo: Diseñar y simular un sistema de orquestación de contenedores utilizando Amazon ECS y Kubernetes, permitiendo a los usuarios desplegar, escalar y gestionar aplicaciones en contenedores.

Instrucciones:

Diseñar clases para ECS y Kubernetes:

- Crea clases que representen un clúster de Amazon ECS y un clúster de Kubernetes.
- Implementa métodos para definir tareas, servicios y políticas de escalado automático.

Gestión de contenedores:

- Implementa funciones para desplegar, escalar y eliminar contenedores en los clústeres de ECS y Kubernetes.
- Simular la gestión de recursos y la distribución de carga en los clústeres.

Integración con instancias EC2:

- Crea funciones que permitan gestionar la infraestructura subyacente para los clústeres de ECS y Kubernetes, incluyendo la creación y eliminación de instancias EC2.
- Implementa una funcionalidad para monitorizar el estado y el rendimiento de los clústeres.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios interactuar con los clústeres de ECS y Kubernetes y gestionar aplicaciones en contenedores.
- Implementa casos de estudio que simulen el despliegue de aplicaciones reales y evalúen el rendimiento y la escalabilidad de los clústeres.

Ejercicio 6: Simulación de funciones serverless con AWS Lambda

Objetivo: Diseñar y simular un sistema de funciones serverless similar a AWS Lambda, que permita a los usuarios definir, desplegar y ejecutar funciones en respuesta a eventos.

Instrucciones:

Diseñar clases para LambdaFunction y Event:

- Crea una clase que represente una función Lambda, con métodos para definir y ejecutar la función.
- Crea una clase que represente un evento, que desencadene la ejecución de una función Lambda.

Gestión de funciones Lambda:

- Implementa funciones para crear, actualizar y eliminar funciones Lambda.
- Simula la ejecución de funciones Lambda en respuesta a diferentes tipos de eventos (por ejemplo, eventos de S3, eventos de DynamoDB).

Integración con eventos:

- Crea funciones que permitan asociar funciones Lambda con diferentes tipos de eventos y gestionar sus desencadenadores.
- Implementa una funcionalidad para monitorizar y registrar la ejecución de funciones Lambda.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios definir, desplegar y gestionar funciones Lambda.
- Implementa escenarios que simulen el uso de funciones Lambda en aplicaciones reales y evalúen su rendimiento y escalabilidad.

Ejercicio 7: Simulación de procesamiento por lotes con AWS Batch

Objetivo: Crear un sistema de procesamiento por lotes similar a AWS Batch, que permita a los usuarios definir y ejecutar trabajos en un entorno gestionado.

Instrucciones:

Diseñar clases para BatchJob y JobQueue:

- Crea una clase que represente un trabajo por lotes, con métodos para definir y ejecutar el trabajo.
- Crea una clase que represente una cola de trabajos, que gestione la ejecución de trabajos en orden.

Gestión de trabajos por lotes:

- Implementa funciones para crear, actualizar y eliminar trabajos por lotes.
- Simula la gestión de recursos y la distribución de carga para la ejecución de trabajos.

Integración con instancias EC2:

- Crea funciones que permitan gestionar la infraestructura subyacente para la ejecución de trabajos por lotes, incluyendo la creación y eliminación de instancias EC2.
- Implementa una funcionalidad para monitorizar el estado y el rendimiento de los trabajos.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios definir, desplegar y gestionar trabajos por lotes.
- Implementa casos de estudio que simulen el procesamiento por lotes en aplicaciones reales y evalúen su rendimiento y escalabilidad.

Ejercicio 8: Simulación de amazon Lightsail para VPS

Objetivo: Crear una simulación de VPS utilizando Amazon Lightsail, que permita a los usuarios crear, gestionar y monitorizar instancias de VPS.

Instrucciones:

Diseñar clases para LightsailInstance y LightsailSnapshot:

- Crea una clase que represente una instancia de Lightsail, con métodos para iniciar, detener y eliminar la instancia.
- Crea una clase que represente una instantánea de Lightsail, que permita la restauración de instancias a partir de instantáneas.

Gestión de instancias de Lightsail:

- Implementa funciones para crear, actualizar y eliminar instancias de Lightsail.
- Simula la gestión de recursos y la asignación de IPs estáticas.

Gestión de instantáneas:

- Crea funciones que permitan crear, listar y restaurar instantáneas de Lightsail.
- Implementa una funcionalidad para gestionar la recuperación de instancias a partir de instantáneas.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios gestionar instancias y instantáneas de Lightsail.
- Implementa escenarios que simulen el uso de Lightsail en aplicaciones reales y evalúen su rendimiento y disponibilidad.

Ejercicio 9: Implementación de Amazon FSx para Windows y Lustre

Objetivo: Diseñar y simular sistemas de archivos de alto rendimiento similares a Amazon FSx for Windows y FSx for Lustre, que permitan a los usuarios gestionar almacenamiento de archivos y optimizar el rendimiento para diferentes cargas de trabajo.

Instrucciones:

Diseñar clases para FSxForWindows y FSxForLustre:

- Crea clases que representen sistemas de archivos FSx for Windows y FSx for Lustre, con métodos para gestionar archivos y carpetas.
- Implementa métodos para optimizar el rendimiento del sistema de archivos para diferentes tipos de cargas de trabajo.

Gestión de archivos y carpetas:

- Implementa funciones para crear, leer, escribir y eliminar archivos y carpetas en los sistemas de archivos.
- Simula la replicación y la recuperación de datos en caso de fallos.

Integración con Instancias EC2:

- Crea funciones que permitan montar y desmontar los sistemas de archivos en instancias EC2.
- Implementa una funcionalidad para monitorizar el estado y el rendimiento de los sistemas de archivos.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios gestionar sistemas de archivos FSx for Windows y FSx for Lustre.
- Implementa casos de estudio que simulen el uso de sistemas de archivos de alto rendimiento en aplicaciones reales y evalúen su rendimiento y disponibilidad.

Ejercicio 10: Simulación de Amazon Outposts

Objetivo: Crear una simulación de Amazon Outposts, que permita a los usuarios gestionar infraestructura local de AWS y extender servicios de AWS a entornos on-premises.

Instrucciones:

Diseñar clases para Outpost y OutpostService:

- Crea una clase que represente un Outpost, con métodos para gestionar la infraestructura y los servicios desplegados.
- Crea una clase que represente un servicio desplegado en el Outpost, con métodos para gestionar el estado y el rendimiento del servicio.

Gestión de infraestructura local:

- Implementa funciones para desplegar, actualizar y eliminar infraestructura en el Outpost.
- Simula la gestión de recursos y la integración con servicios de AWS.

Gestión de servicios desplegados:

- Crea funciones que permitan gestionar servicios desplegados en el Outpost, incluyendo la monitorización y la recuperación en caso de fallos.
- Implementa una funcionalidad para escalar servicios automáticamente en respuesta a cambios en la carga de trabajo.

Simulación de escenarios reales:

- Diseña una interfaz de usuario que permita a los usuarios gestionar infraestructura y servicios en un Outpost.
- Implementa casos de estudio que simulen el uso de Amazon Outposts en entornos on-premises y evalúen su rendimiento y disponibilidad.