Ejercicio de Investigación – Juan Pincheira

Investigación Python

1. Lista y tupla en Python:
- Lista: Colección ordenada y mutable. Permite agregar, eliminar o modificar elementos después de creada. Ej: [1, 2, 3] o ['manzana', 'pera'].
- Tupla: Colección ordenada pero inmutable. No se puede cambiar una vez creada. Es útil para datos que no deberían modificarse. Ej: (1, 2, 3).

2. Espacio de nombres:
Es el "espacio lógico" donde Python guarda la relación entre los nombres (identificadores) y los objetos a los que apuntan. Por ejemplo, cuando defino una variable llamada edad, ese nombre queda registrado dentro de un espacio de nombres (local, global, de módulo, etc.).

3. Variable local vs global:
- Variable local: Solo existe dentro del bloque en el que fue creada, normalmente una función. Cuando termina la función, la variable desaparece.
- Variable global: Se define fuera de las funciones y puede ser usada desde cualquier parte del programa.

4. IDE:
Es una aplicación que reúne en un solo lugar un editor de código, depurador, consola y herramientas adicionales para programar más cómodo. Ejemplos que se usan con Python: VS Code, PyCharm, Thonny, Antigravity.

5. Módulos en Python:
Un módulo es un archivo .py que contiene funciones, variables y/o clases que podemos reutilizar en otros programas usando import. Esto permite organizar mejor el código. Ejemplos de módulos estándar: math, random, datetime.

6. Diferencia entre matriz y lista:
- Lista: Estructura unidimensional. Puede mezclar tipos de datos y es muy flexible para trabajar con colecciones pequeñas o variadas.
- Matriz: Estructura bidimensional o multidimensional, optimizada para operaciones numéricas y científicas. Las listas son más versátiles, mientras que las matrices son mejores para cálculos intensivos. (En Python no existe un tipo nativo llamado matriz, se simula con listas anidadas o se usa NumPy para matrices reales optimizadas.)

7. Operadores en Python:
Son símbolos que indican operaciones sobre uno o más valores.

Ejemplos:
- Aritméticos: +, -, *, /, %, ** (potencia).
- Comparación: ==, !=, >, <, >=, <=.
- Lógicos: and, or, not.

Ejercicio de investigación de Bases de Datos

1. Diferencia entre base de datos relacional y no relacional:
- Relacional: Organiza la información en tablas con filas y columnas. Usa SQL para consultar los datos y suele exigir un esquema bien definido. Es ideal cuando hay muchas relaciones entre entidades y se necesita consistencia fuerte. Ejemplos: MySQL, PostgreSQL, SQL Server.
- No relacional (NoSQL): Almacena los datos en formatos más flexibles como documentos, pares clave-valor, grafos o columnas. Es útil cuando el esquema cambia seguido o se manejan grandes volúmenes de datos distribuidos. Ejemplos: MongoDB (documentos), DynamoDB (clave-valor).

2. Índices:
Son estructuras adicionales que crea la base de datos (por ejemplo, árboles B o tablas hash) para acelerar las búsquedas en una o varias columnas. En lugar de revisar fila por fila toda la tabla, el motor usa el índice para llegar más rápido al dato. La desventaja es que ocupan espacio y pueden hacer que las inserciones y actualizaciones sean un poco más lentas.

3. Claves primarias y secundarias:
- Clave primaria (PRIMARY KEY): Campo o conjunto de campos que identifica de forma única cada fila de una tabla. No puede ser nulo ni repetirse.
- Clave foránea (FOREIGN KEY): Campo en una tabla que apunta a la clave primaria de otra tabla. Sirve para relacionar información entre tablas y mantener la integridad referencial.

4. Uniones internas y externas:
- INNER JOIN: Devuelve solo los registros que tienen coincidencias en ambas tablas según la condición dada.
- LEFT JOIN: Devuelve todos los registros de la tabla de la izquierda, y los que coincidan de la derecha. Si no hay coincidencia, los campos de la derecha aparecen como NULL.
- RIGHT JOIN: Similar al LEFT JOIN pero empezando desde la tabla de la derecha.
- FULL OUTER JOIN: Devuelve todos los registros de ambas tablas, coincidan o no.

5. Diferencia entre DROP TABLE y TRUNCATE TABLE:
- DROP TABLE: Elimina completamente la tabla, su estructura y todos sus datos. Después de un DROP, la tabla deja de existir.
- TRUNCATE TABLE: Borra todos los registros de la tabla, pero mantiene la estructura para poder seguir usándola. Suele ser más rápido que DELETE sin condición porque no registra cada fila una por una.

6. Tipos de datos SQL más comunes:
- Numéricos: INT, BIGINT, DECIMAL, FLOAT.
- Texto: CHAR, VARCHAR, TEXT.
- Fecha y hora: DATE, TIME, DATETIME, TIMESTAMP.
- Booleanos: BOOLEAN o TINYINT(1) según el motor. Elegir bien el tipo de dato ayuda a ahorrar espacio y mejorar el rendimiento.

7. Cláusulas WHERE y HAVING:
- WHERE: Filtra filas antes del agrupamiento. Se usa con SELECT, UPDATE o DELETE para indicar qué registros se afectan.

- HAVING: Se usa junto con GROUP BY para filtrar grupos de resultados que ya fueron agregados (por ejemplo, solo grupos con COUNT(*) > 10).

Fundamentos de AWS Cloud – Primera Parte

1. Definición de IaaS, PaaS y SaaS:
- IaaS (Infrastructure as a Service): AWS entrega infraestructura básica (servidores, red, almacenamiento) y el cliente se encarga del sistema operativo y las aplicaciones. Ejemplo: Amazon EC2, Amazon EBS.
- PaaS (Platform as a Service): Además de la infraestructura, se ofrece una plataforma lista para desplegar aplicaciones sin preocuparse tanto por el sistema operativo. Ejemplo: AWS Elastic Beanstalk.
- SaaS (Software as a Service): El usuario consume directamente la aplicación terminada vía web, sin administrar infraestructura ni plataforma. Ejemplos generales: Gmail, Office 365; en AWS, servicios totalmente administrados como Amazon WorkMail.

2. Seis ventajas de la informática en la nube:
- Pago por uso: Solo se paga por los recursos consumidos.
- Elasticidad: Se puede escalar hacia arriba o abajo rápidamente.
- Agilidad: Implementar y probar nuevas ideas es mucho más rápido.
- Alcance global: Los servicios están disponibles en distintas regiones del mundo.
- Alta disponibilidad: AWS diseña su infraestructura para minimizar caídas.
- Reducción de costos iniciales: No se requiere comprar hardware físico ni mantener datacenters propios.

3. Región de AWS y zona de disponibilidad:
- Región: Conjunto de centros de datos ubicados en una misma área geográfica (por ejemplo, us-east-1 en Norteamérica o sa-east-1 en São Paulo).
- Zona de disponibilidad (AZ): Uno o más centros de datos dentro de una región, aislados entre sí a nivel físico y eléctrico, pero conectados mediante baja latencia.

4. Ejemplos de regiones de AWS:
Algunas regiones son: us-east-1 (Norte de Virginia), us-west-1 (Norte de California), eu-west-1 (Irlanda), ap-south-1 (Mumbai), sa-east-1 (São Paulo). Escoger bien la región impacta en la latencia y en el cumplimiento de regulaciones.

5. Categorías en las que se agrupan los servicios de AWS:
- Cómputo (EC2, Lambda, ECS).
- Almacenamiento (S3, EBS, EFS).
- Bases de datos (RDS, DynamoDB, Aurora).
- Redes y entrega de contenido (VPC, Route 53, CloudFront).
- Seguridad, identidad y cumplimiento (IAM, KMS, CloudTrail).
- Analítica, Machine Learning, herramientas de administración y más.

6. Diferencia entre almacenamiento de objetos y almacenamiento en bloques:
- Objetos: Cada archivo se guarda como un objeto con metadatos e ID único. Es ideal para almacenar grandes cantidades de datos estáticos como imágenes, copias de seguridad o sitios

estáticos. Ejemplo: Amazon S3.
- Bloques: El disco se divide en bloques y el sistema operativo los gestiona como si fuera un disco duro tradicional. Es más adecuado para sistemas de archivos y bases de datos. Ejemplo: Amazon EBS.

7. Dos servicios de cómputo de AWS y explicación:
- Amazon EC2: Permite lanzar instancias de servidores virtuales con control sobre sistema operativo, red y almacenamiento. Es muy flexible para casi cualquier tipo de carga.
- AWS Lambda: Servicio de computación sin servidor (serverless) donde se ejecuta código en respuesta a eventos sin administrar servidores ni preocuparse de la infraestructura.

Fundamentos de AWS Cloud – Segunda Parte

1. Modelo de responsabilidad compartida de AWS:
AWS se encarga de la "seguridad de la nube" (infraestructura física, hardware, red, hipervisor, etc.), mientras que el cliente es responsable de la "seguridad en la nube" (configuración de servicios, sistemas operativos, aplicaciones, permisos IAM, cifrado de datos, etc.).

2. Rol de AWS Identity and Access Management (IAM):
Un rol IAM es una identidad que define un conjunto de permisos, pero que no está asociada a una persona específica. En lugar de credenciales de usuario y contraseña, los roles se asumen temporalmente por usuarios, servicios o cuentas externas para realizar tareas concretas.

3. Política de AWS IAM:
Es un documento en formato JSON que especifica qué acciones están permitidas o denegadas sobre qué recursos y en qué condiciones. Las políticas se adjuntan a usuarios, grupos o roles para controlar el acceso.

4. Amazon Machine Image (AMI):
Una AMI es una plantilla que contiene la información necesaria para lanzar una instancia EC2: sistema operativo, configuraciones, software preinstalado, etc. Permite crear múltiples instancias iguales de forma rápida.

5. Distintos tipos de instancias de Amazon EC2:
- Familia T (por ejemplo t3.micro): Uso general con ráfagas de CPU, ideal para entornos de prueba o aplicaciones poco exigentes.
- Familia M: Uso general equilibrado entre CPU, memoria y red, adecuada para aplicaciones empresariales.
- Familia C: Optimizada para cómputo, ideal para procesamiento intensivo, servidores web de alto rendimiento o análisis de datos.
- Familia R: Optimizada para memoria, usada para bases de datos en memoria o cachés grandes.

6. Amazon Virtual Private Cloud (VPC):
Es una red virtual aislada dentro de AWS donde el cliente define su propio rango de direcciones IP, subredes, tablas de enrutamiento y gateways. Permite tener control detallado de la comunicación interna y con internet.

7. Diferencia entre subred pública y privada:
- Subred pública: Tiene una ruta hacia un Internet Gateway, por lo que las instancias en ella pueden tener IP pública y comunicarse directamente con internet.
- Subred privada: No tiene ruta directa a internet. El tráfico hacia fuera suele pasar por un NAT Gateway o VPN, lo que incrementa la seguridad.

AWS Well-Architected Framework

1. Los cinco pilares de Well-Architected Framework:
- Excelencia operativa.
- Seguridad.
- Fiabilidad.
- Eficiencia de rendimiento.
- Optimización de costos.

Estos pilares sirven como guía para evaluar y diseñar arquitecturas en la nube bien construidas.

2. Tres áreas de excelencia operativa en la nube:
- Preparar: Definir procesos, estándares y documentación antes de pasar a producción.
- Operar: Monitorear el sistema, responder a incidentes y automatizar tareas repetitivas.
- Evolucionar: Mejorar continuamente los procesos y la arquitectura basándose en métricas y retroalimentación.

3. Principios de diseño que refuerzan la seguridad del sistema:
- Aplicar el principio de privilegio mínimo con IAM.
- Habilitar trazabilidad mediante logs (CloudTrail, CloudWatch Logs).
- Implementar defensa en profundidad (varias capas de seguridad: red, aplicación, datos).
- Automatizar la seguridad (plantillas, políticas, escaneos automáticos).

4. Principios de diseño que aumentan la fiabilidad:
- Diseñar para fallar: asumir que los componentes pueden fallar y prepararse para ello.
- Usar redundancia y despliegues en múltiples zonas de disponibilidad.
- Automatizar la recuperación (auto scaling, health checks).
- Probar los planes de recuperación regularmente.

5. Áreas para lograr eficiencia de rendimiento:
- Elegir tipos de instancias adecuados a la carga.
- Utilizar servicios administrados que escalan automáticamente.
- Medir y revisar el uso de recursos para ajustar la arquitectura.
- Aprovechar cachés y CDN para reducir la latencia.

6. Enfoques para utilizar los recursos de AWS de forma rentable:
- Apagar recursos que no se usan (instancias de desarrollo fuera de horario).
- Usar instancias reservadas o Savings Plans para cargas estables.
- Utilizar instancias Spot para trabajos flexibles.
- Monitorear el costo con AWS Cost Explorer y la facturación detallada.

AWS CloudFormation

1. Aprovisionamiento de la configuración:
Se refiere a la creación y puesta en marcha de todos los recursos necesarios (servidores, redes, bases de datos, etc.) con los parámetros deseados. En la nube, lo ideal es que este aprovisionamiento sea automatizado y repetible.

2. Administración de la configuración:
Consiste en mantener los sistemas en un estado conocido y consistente, gestionando cambios de forma controlada. Algunas herramientas de uso común son: Ansible, Puppet, Chef y AWS Systems Manager, que permiten aplicar configuraciones de manera automática a muchos servidores.

3. Integración continua (CI):
Práctica de desarrollo donde los cambios de código se integran frecuentemente en un repositorio central y se ejecutan pruebas automáticas. El objetivo es detectar errores temprano y mantener el código siempre en un estado desplegable.

4. Entrega continua (CD):
Extiende la integración continua hasta el despliegue automatizado a entornos de prueba o producción. Cada cambio que pasa las pruebas puede ser lanzado de forma segura y rápida.

5. Qué es AWS CloudFormation:
Es un servicio que permite definir la infraestructura de AWS como código mediante plantillas en JSON o YAML. Tres ventajas principales son:
- Automatización del despliegue de recursos.
- Repetibilidad: se pueden recrear entornos idénticos en distintas cuentas o regiones.
- Control de versiones de la infraestructura, facilitando auditorías y cambios controlados.

6. JSON vs YAML:
- YAML es más legible para humanos porque usa indentación en lugar de muchas llaves y comas.
- JSON es más estricto y suele usarse también para APIs; YAML permite comentarios y sintaxis más flexible.
- Los mismos datos suelen ocupar menos líneas en YAML que en JSON, lo que simplifica las plantillas grandes.

7. Pila (stack) en AWS CloudFormation:
Es el conjunto de recursos de AWS que se crean, actualizan o eliminan como una sola unidad a partir de una plantilla. Por ejemplo, una pila puede incluir una VPC, subredes, instancias EC2 y un balanceador de carga.

Facturación AWS

1. Planes de soporte de AWS:
- Basic: Incluido sin costo adicional. Acceso a documentación, foros y soporte para problemas relacionados con la cuenta.
- Developer: Pensado para entornos de prueba y desarrollo. Incluye soporte técnico durante horario laboral.

- Business: Para cargas de trabajo en producción. Ofrece soporte 24/7, tiempos de respuesta más rápidos y acceso a guías de buenas prácticas.
- Enterprise On-Ramp y Enterprise: Diseñados para clientes críticos. Incluyen Technical Account Manager (TAM), soporte proactivo y tiempos de respuesta muy reducidos.

2. Calculadora de costo mensual de AWS:
La AWS Pricing Calculator es una herramienta web donde se pueden seleccionar servicios (EC2, S3, RDS, etc.), configurar su uso estimado y obtener una aproximación del costo mensual. Es útil para planificar presupuestos y comparar alternativas de arquitectura.

3. Modelos de precios de Amazon EC2:
- On-Demand: Se paga por hora o segundo de uso sin compromiso a largo plazo. Flexible pero más caro.
- Reserved Instances / Savings Plans: Ofrecen descuentos importantes a cambio de comprometerse a usar cierta capacidad durante 1 o 3 años.
- Spot Instances: Permiten usar capacidad sobrante con grandes descuentos, pero AWS puede interrumpirlas cuando necesite esa capacidad.
- Dedicated Hosts: Servidores físicos dedicados para un solo cliente, útiles para cumplir requisitos de licenciamiento o normativas.