

Universidad ORT Uruguay

Facultad de Ingeniería: Escuela de Tecnología

PROGRAMACION PARA DEVOPS

Marcelo Sosa N°141855

Hernán Pintos N°151357

Grupo: N4A

Docentes: Leonardo Genta - Guillermo Ferradas

Declaracion de autoría

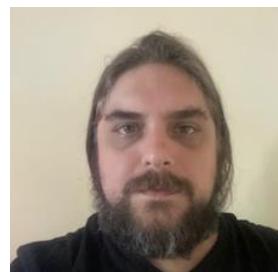
Nombres de los autores: Marcelo Sosa
Hernán Pintos

Declaración de autoría Nosotros, **Marcelo Sosa y Hernán Pintos**, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Aseguramos, bajo nuestra entera responsabilidad, que:

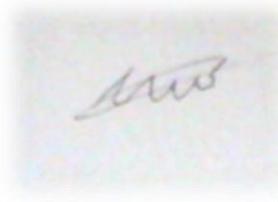
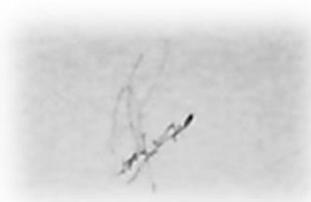
- La obra fue producida en su totalidad mientras realizábamos la carrera ANALISTA EN INRAESTRUCTURA INFORMATICA;- Cuando hemos consultado trabajos publicados por otros, lo hemos atribuido con claridad;- Cuando hemos citado obras de otros autores, hemos indicado las fuentes. Con excepción de dichas citas, la obra es enteramente nuestra;- En la obra, hemos acusado recibo de las ayudas recibidas;- Cuando la obra se basa en trabajo realizado juntamente con otros no pertenecientes al equipo, hemos explicado claramente que parte fue contribuida por dichos terceros, y que parte fue contribuida por nosotros;- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto los casos en que se han realizado las aclaraciones correspondientes.



Hernán Pintos



Marcelo Sosa



REPOSITORIO GIT



<https://github.com/ortdevops2025/obligatorio2025>

INDICE

SECCION I - OBJETIVOS

El objetivo de este documento es dar cumplimiento a las tareas solicitadas por los docentes en la materia de programación para DevOps

Esto se realizará cumpliendo los siguientes ítems:

Creación de script de Bash que cumpla el primer requerimiento de la letra

Creación de script de Python que cumpla requerimientos de AWS pedidos para la segunda parte de la letra del obligatorio

Documento de registros de ejecución e información

Reflexiones sobre los desafíos planteados

Para lograr estos objetivos, utilizaremos los conocimientos y herramientas obtenidos en la materia a lo largo del semestre.

SECCION II – USO DE GIT Y RECURSOS

Para la realización del trabajo obligatorio se utilizó GIT para control de cambios y trabajo en equipo.

Si bien todos los documentos están en el formato solicitado en el repositorio de GIT creado con la finalidad de la realización de este trabajo, se incluye en este documento, en las siguientes secciones, el código de ambos scripts, pruebas de uso y contenido de archivos de ejemplo así como también el ambiente en donde las pruebas fueron realizadas.

Se utilizó también GITHUB Desktop para la carga de documentación y mantenimiento de archivos.

REPOSITORIO DE GIT

Para la realización de este trabajo obligatorio, el siguiente repositorio de GIT fue creado:

<https://github.com/ortdevops2025/obligatorio2025>

The screenshot shows a GitHub repository page for 'obligatorio2025'. At the top, there's a header with navigation icons and the URL 'github.com/ortdevops2025/obligatorio2025'. Below the header, the repository name 'obligatorio2025' is displayed with a 'Private' badge. The main interface includes tabs for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', and 'Insights', with 'Code' being the active tab. A summary bar shows 'main' branch, '1 Branch', '0 Tags', and a search bar labeled 'Go to file'. The repository content section lists a single commit from 'ortdevops2025' with the message 'Initial commit'. This commit contains two files: 'README.md' and 'README'. The 'README.md' file is shown with its content: 'obligatorio2025'. Below the files, a description reads: 'Repository para obligatorio de DevOps Universidad ORT 2025 Sosa - Pintos'.

Para evitar filtraciones o acceso no permitido, este repositorio fue puesto como privado, solo los usuarios del obligatorio y los docentes tendrán acceso al mismo para la realización del trabajo y posteriormente la defensa de este.



SECCION III – SCRIPT DE BASH

Para la realización y pruebas de este script fue utilizada la máquina virtual de CentOS 8 proporcionada en el curso con algunas actualizaciones.

Versión de CentOS:

```
[alumno@asl final]$ cat /etc/centos-release
CentOS Linux release 8.5.2111
[alumno@asl final]$
```

Se muestra a continuación el script que esta adjunto en el repositorio.

```
#####
#!/bin/bash
#Fecha Modificado 24/11/2025

flaginfo=0      # Variable para -i vale distinto de 0 se muestra flaginforamcion
password=""    # Variable para -c que guarda el password
archivousu=""  # archivo con la lista de usuarios
contadorusuok=0 # Contador de usuarios creados OK

#####
#####Comprobacion de Parametros#####
#####

if [ "$1" = "-i" ] && [ "$2" = "-c" ] #Verifica el orden de los parametros i y c
then
    flaginfo=1
    password="$3"
    archivousu="$4"
else
    if [ "$1" = "-c" ] && [ "$3" = "-i" ] #Verifica segundo orden posible de los parametros i y c
    then
        flaginfo=1
        password="$2"
        archivousu="$4"
    else
        if [ "$1" = "-i" ] #Verifica si se utilizo solo el parametro de informacion
        then
            flaginfo=1
            archivousu="$2"
        else
            if [ "$1" = "-c" ] #Verifica si se utilizo solo el parametro de password
            then
                password="$2"
                archivousu="$3"
            else
                archivousu="$1" #Procsa el archivo si no se utilizron parametros
            fi
        fi
    fi
fi

#####
#####VALIDAR archivousu #####
#####

if [ ! $# -ge 1 ] #Verifica que se usen al menos un parametro en el script
then
    echo "Debe proporcionar al menos un parametro"
```

```

exit 2
fi

if [ ! -f "$archivousu" ] #Verifica que el archivo sea regular
then
    echo "$archivousu no existe o no es regular" >&2
    exit 3
fi

if [ ! -r "$archivousu" ] #Verifica que el archivo tenga read only
then
    echo "$archivousu no tiene permiso de lectura" >&2
    exit 4
fi

if [ "$(id -u)" -ne 0 ] #IAG Geneardo, verifica si es root el usuario al momento de ejecucion
then
    echo "Este script debe ejecutarse como root" >&2
    exit 5
fi

#####
#####DESEMPAQUETADO DE archivousu#####
#####

IFS=$'\n' #Defino separador de lineas como el salto de linea

for i in $(cat "$archivousu")
do

#verificador de campos de linea para que sean 5

campos=$(echo "$i" | tr -cd ':' | wc -c) #Cuenta los delimitadores : para verificar sintaxis de la linea

if [ "$campos" -ne 4 ]
then
    echo "Sintaxis incorrecta del $archivousu pasado como parametro" >&2
    echo "La linea $i no contiene exactamente 5 campos separados por :" >&2
else

#Asigna valores a las variables

USUARIO=$(echo "$i" | cut -d: -f1)
COMENTARIO=$(echo "$i" | cut -d: -f2)
HOME=$(echo "$i" | cut -d: -f3)
CREARHOME=$(echo "$i" | cut -d: -f4 | tr '[:lower:]' '[:upper:]')
SHELL=$(echo "$i" | cut -d: -f5-)

# Comprobacion de existencia de usuario

if id "$USUARIO" >/dev/null 2>&1 #el comando id retorna 0 si es exitoso o distinto de 0 si no lo es
then
    echo "ATENCION: el usuario $USUARIO ya existe"
    continue
fi

# Valores por defecto

[ -z "$SHELL" ] && SHELL="/bin/bash"
[ -z "$HOME" ] && HOME="/home/$USUARIO"
[ -z "$CREARHOME" ] && CREARHOME="SI"

# Creacion del usuario

```

```

if [ "$CREARHOME" = "SI" ]
then
    # con -m se crea el directorio home si no existe
    useradd -m -c "$COMENTARIO" -d "$HOME" -s "$SHELL" "$USUARIO" >/dev/null 2>&1
else
    # con -M se evita la creacion del directorio home
    useradd -M -c "$COMENTARIO" -d "$HOME" -s "$SHELL" "$USUARIO" >/dev/null 2>&1
fi

if [ $? -eq 0 ] #verifica que el comando de agregar usuario haya sido exitoso,(exit 0)
then
    if [ -n "$password" ] # se chequea que la variable password no este vacia
    then
        # se establece la contraseña del usuario usando chpasswd, ocultando errores
        # chpasswd asigna la contraseña al usuario leida desde la entrada estandar
        echo "$USUARIO:$password" | chpasswd 2>/dev/null
    fi

    contadorusuok=$((contadorusuok + 1)) #Sumo uno si se agrego el usuario correctamente

    if [ "$flaginfo" -eq 1 ]; then #Muestra informacion si se uso el parametro -i
        echo "Usuario $USUARIO creado con éxito con datos indicados:"
        # echo -e "\t" para lograr identacion
        echo -e "\tComentario: ${COMENTARIO:-<valor por defecto>}"
        echo -e "\tDir home: ${HOME:-<valor por defecto>}"
        echo -e "\tAsegurado existencia de directorio home: ${CREARHOME:-<valor por defecto>}"
        echo -e "\tShell por defecto: ${SHELL:-<valor por defecto>}"
        echo
    fi
else
    echo "ATENCION: el usuario $USUARIO no pudo ser creado"
fi
#done < "$archivousu"
done

#####
###INFORMACION A MOSTRAR###
#####

if [ "$flaginfo" -eq 1 ]; then
    echo "Se han creado $contadorusuok usuarios con exito."
fi
#####

```

PRUEBA DE USO:

Se procede a mostrar ejemplos de uso y los resultados obtenidos incluidos algunos errores testeados:

1 – Script sin Parámetros:

```
CentOS8 clonar [Corriendo] - Oracle VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal ▾
Archivo Editar Ver Buscar Terminal Ayuda
[alumno@asl final]$ ./ej1_crea_usuarios.sh
Debe proporcionar al menos un parametro
[alumno@asl final]$
```

En este ejemplo, al no colocarse ningún parámetro el script devuelve un resultado de error indicando que el script necesita al menos un parámetro para funcionar (el archivo)

2 – Archivo no regular

```
Archivo Editar Ver Buscar Terminal Ayuda
[alumno@asl final]$ ./ej1_crea_usuarios.sh sinarchivo
sinarchivo no existe o no es regular
[alumno@asl final]$
```

3 – Uso normal con un archivo de usuarios que tiene 3 usuarios válidos para agregar y uno que ya existe:

```
pepe:Este es mi amigo pepe:/home/jose:SI:/bin/bash
terran:Jim_Raynor:/home/raynor:NO:/bin/sh
elmaligno::::/bin/el_maligno
root::::/bin/root
```

```
[alumno@asl final]$ sudo ./ej1_crea_usuarios.sh -i -c "123456" usuarios
[sudo] password for alumno:
Usuario pepe creado con éxito con datos indicados:
Comentario: Este es mi amigo pepe
Dir home: /home/jose
Asegurado existencia de directorio home: SI
Shell por defecto: /bin/bash

Usuario terran creado con éxito con datos indicados:
Comentario: Jim_Raynor
Dir home: /home/raynor
Asegurado existencia de directorio home: NO
Shell por defecto: /bin/sh

Usuario elmaligno creado con éxito con datos indicados:
Comentario: <valor por defecto>
Dir home: /home/elmaligno
Asegurado existencia de directorio home: SI
Shell por defecto: /bin/el_maligno

ATENCION: el usuario root ya existe
Se han creado 3 usuarios con éxito.
[alumno@asl final]$
```

En este caso, se deberán de agregar tres usuarios, pepe, Jim Raynor y el maligno.

El usuario root ya existe por lo que el comando debería avisar de esta situación.

Se utilizan los dos parámetros opcionales también -i y -c con un password 123456

Se observa en la imagen el resultado obtenido.

Archivo /etc/passwd mostrando los nuevos usuarios agregados

```
cockpit-wsinstance:x:977:975:User for cockpit-ws instances:/none
flatpak:x:976:974:User for flatpak system helper::/sbin/nologin
rngd:x:975:973:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
prueba1:x:1001:1001:Primera prueba:/home/prueba:/bin/bash
terran:x:1003:1003:Jim_Raynor:/home/raynor:/bin/sh
pepe:x:1004:1004:Este es mi amigo pepe:/home/jose:/bin/bash
elmaligno:x:1005:1005::/home/elmaligno:/bin/el_maligno
latumouast:password:  
$1$GKQHq$CfLJF
```

SECCION IV – SCRIPT DE PYTHON

Para la creación de este script y su ejecución, se deben de satisfacer ciertos requerimientos especiales relacionados con el ambiente en donde el script se va a utilizar.

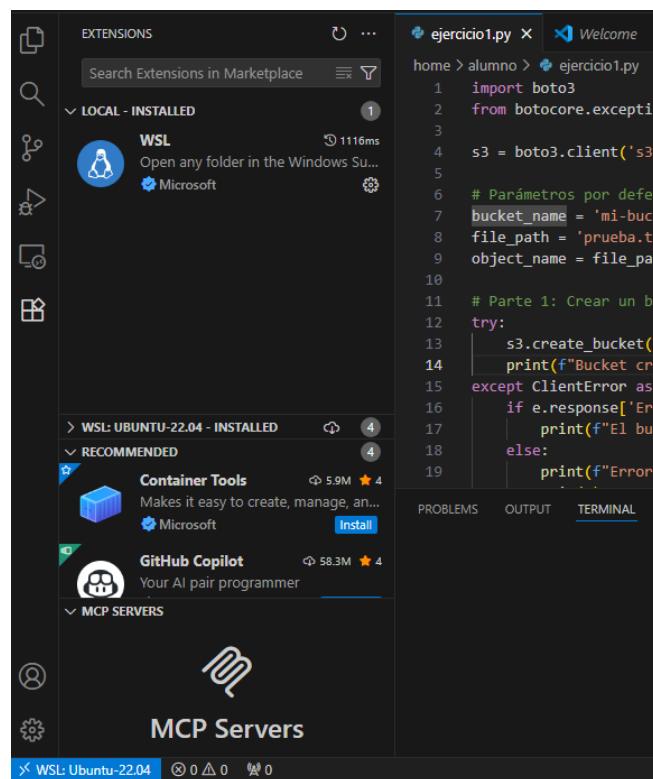
En este caso, se configuro para el desarrollo una instancia de WSL en Windows 11 utilizando la imagen de Ubuntu 22 proporcionada por el repositorio de WSL.

Se instalo el siguiente software una vez configurado el ambiente de WSL:

```

alumno@PCW11:~$ pip3 show boto3
Name: boto3
Version: 1.40.68
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache-2.0
Location: /home/alumno/.local/lib/python3.10/site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
alumno@PCW11:~$ aws --version
aws-cli/2.31.31 Python/3.13.9 Linux/6.6.87.2-microsoft-standard-WSL2 exe/x86_64.ubuntu.22
alumno@PCW11:~$
```

Para el desarrollo del script, se utilizó Visual Studio Code con los plugins de Boto3, WSL y Python instalados.



Configuración de GIT

Se utilizo la versión de GIT para Windows Desktop y también la versión GIT en WSL Ubuntu. Para la configuración de GIT en WSL se procedió a la clonación inicial del repositorio:

```
git clone https://github.com/ortdevops2025/obligatorio2025.git
```

NOTA: PARA LA EJECUCION DE LOS SCRIPTS, SE TOMA EN CUENTA QUE LOS ARCHIVOS ESTAN EN LA MAQUINA DONDE SE CORRERAN LOS MISMOS, POR LO TANTO EL REPOSITORIO DEBE SER CLONADO A LA MISMA COMO SE MUESTRA A CONTINUACION.

La carpeta de aplicación esta ya contenida dentro del repositorio Obligatorio2025.git

INSTRUCCIONES DE EJECUCION PYTHON

Los siguientes pasos son necesarios para el correcto despliegue y ejecución de la aplicación.

Tener AWS CLI configurado / credenciales disponibles en la máquina donde ejecutes los scripts.

Suponiendo que se ejecute este proyecto desde una máquina de LINUX, se debe tener el AWS CLI instalado y configurado, Python 3, Pip, Boto3.

Adicionalmente debemos contar con la KEY de EC2 para el acceso de SSH si es necesario y un rol que nos permita acceder al S3.

Para ello, como ejemplo en Ubuntu AWS el cual fue utilizado ejecutamos:

```
Sudo apt update  
Sudo apt upgrade  
sudo apt install python3  
sudo apt install python3 python3-pip -y  
pip3 install boto3  
sudo apt install unzip  
pip3 install --upgrade --user awscli
```

Comprobacion de versiones usando como ejemplo las utilizadas:

```
apt show awscli | grep Version  
aws-cli/2.13.24 Python/3.11.5 Linux/6.8.0-31-generic exe/x86_64 ubuntu.22 prompt/off
```

También debemos instalar GIT para poder clonar el repositorio:

```
sudo apt install git
```

Configuramos el AWS CLI (En caso de ser necesario, con las credenciales del ambiente a utilizar)

Se utiliza el comando

aws configure

Se pedirán los siguientes campos los cuales salen de la sesión de AWS a utilizar:

AWS Access Key ID

AWS Secret Access Key

AWS Session Token

Region Name: us-east-1 (si corresponde esa region)

Default Format: json

A continuación se ve un ejemplo de la configuración:

```
aLumino@PCW11:~$ aws configure
AWS Access Key ID [*****]: ASIAXSYCAGFDJQ3YUD5R
AWS Secret Access Key [*****]: qh/130wA0q1ZsTu4nmxJzCZ6BDG3I80Tzwa0Tkij
AWS Session Token [*****]: IQoJb3jpZ2luX2VjEF8aCXVzXldc3QtMiJIMEYCIQCEx6YkTg/iV1P7ILhxGyJ4jv60m8wDNYAx6ZG/nNaVCwIhAoDlf1xIEEEdEWobb
fn8SqAaUcrcYZn7sy01Ly8cwZKrzUCCcgQARoMNTIxhxA10DYyNDcwIgzEr7A6KqGM0cD4F4qkgIvldVfGeYxTM7jsRIw+ck+9bycLZGuZC00BY8WhBpSBBz1AUvdeM0sjtVwpStGXqDPLWA1
sycY8aghIwaYsLfghK/7FJt0Fb06Pg0Bu6qhJhAU/TSL7kwLkj6s+VVIVLq8fyTHAQuBgdKyvUIDBH/vzz7w02LV3tFRT8x1LEZSM8hb7l12pg5oZrk54e4hKzerNs8a0MWTcJG5Vwy6qXDWB
vejTORhGyxwByn5SadFLxPqz3McKvYtOTE17/tmw308qsksvrwGEQ8tyD/HoCQfuzyjWvEVHfKnZ3em8UjqdSwT9f3sWTC5nFkvUtbu9Lsr/RGLyppEafZcyhPtg/4hQ+yQuEuTnvnC9tC8FMK
j2zsgG0pwB4L8/TLSvDH18paHMhQTG2K0SDt35bGLIteE9xIqlU8TOW2az3jQzZyFz7WHhrVs9748xbjZFnWXYHGYQwLnzuqJMj8R622wPrCcHylcXIRNmiLQChxEJXiRXb25xeCCkCvHwT0
/3KsmihGcQw7DuY3rNYb0iJCGAIUCgoophPTeVm0o3fJ8QdGar53p9YkvYLhBCclqQX2sU
Default region name [us-east-1]: us-east-1
Default output format [json]: json
```

En caso de que debamos alterar algún valor, se puede correr el comando nuevamente o acceder al archivo de configuración el cual está en texto plano en la ubicación:

/home/usuario/.aws/credentials

Instalación de agente de SSM de AWS en Ubuntu

sudo snap install amazon-ssm-agent –classic

Una vez teniendo todo configurado, clonamos el repositorio siguiente:

git clone <https://github.com/ortdevops2025/obligatorio2025.git>

PASOS PARA DESPLIEGUE DE APLICACION USANDO SCRIPTS:

Una vez clonado el repositorio e instalado las dependencias de la sección anterior, se debe proceder al despliegue de la instalación.

NOTA: Todos los comandos se deben ejecutar desde la carpeta /obligatorio2025, la cual es la raíz del repositorio clonado.

Desde el Directorio del repositorio clonado, ejecutar el script de instalación y despliegue de la aplicación:

python3 obligatorio.py

```
alumno@PCW11:~/obligatorio2025$ python3 experimental.py
Bucket creado: obligatorio2025qwertyuiop
Archivos subidos correctamente.
Ingresar password maestro de la base:
Obligatorio2025
SG EC2 ya existe: sg-06b0be5f05155fd67
SG RDS ya existe: sg-05f277b4eb7a7ff12
La instancia RDS ya existe.
Esperando a que RDS esté listo...
RDS listo.
Endpoint RDS: app-mysql.cxzqtmaexaa8.us-east-1.rds.amazonaws.com
Instancia EC2 creada: i-050cfec16adf02ae7
Esperando a que EC2 este listo
EC2 esta listo, ya puede navegar en el sitio
SG listos y asignados a EC2 y RDS
Navegue a la IP pública del EC2 cuando esté disponible
```

- Bucket creado: obligatorio2025qwertyuiop → Output de Creacion de Bucket y subida de archivos
- Ingresar password maestro de la base: → Password de la Base de Datos ingresado
- SG EC2 ya existe: sg-06b0be5f05155fd67 → Creacion de Security Groups
- SG RDS ya existe: sg-05f277b4eb7a7ff12 → Creacion de la base de RDS
- La instancia RDS ya existe. Esperando a que RDS esté listo... → Creacion de instancia EC2
- RDS listo. → Salida final
- Endpoint RDS: app-mysql.cxzqtmaexaa8.us-east-1.rds.amazonaws.com
- Instancia EC2 creada: i-050cfec16adf02ae7
- Esperando a que EC2 este listo
- EC2 esta listo, ya puede navegar en el sitio
- SG listos y asignados a EC2 y RDS
- Navegue a la IP pública del EC2 cuando esté disponible

PRUEBA DE USO

Una vez finalizado el script, y la instancia de AWS correctamente iniciada, se podrá probar la aplicación ingresando al siguiente link:

[http://\(IP PUBLICA DE INSTANCIA\)/login/php](http://(IP PUBLICA DE INSTANCIA)/login/php)

Se utiliza el log in:

admin
admin123

Se debe desplegar correctamente la información de la aplicación:

ID	Nombre	Email	Sueldo	Puesto	Acciones
1	Juan Perez	juan@example.com	55,000.00	Desarrollador	Editar Eliminar
2	Ana Gomez	ana@example.com	60,000.00	Analista	Editar Eliminar
3	Carlos Ruiz	carlos.ruiz@example.com	48,000.00	Tester	Editar Eliminar
4	Maria Lopez	maria.lopez@example.com	70,000.00	Lider de Proyecto	Editar Eliminar

SECCION VII – RESULTADOS

DIFICULTADES ENCONTRADAS

Compatibilidad de comandos

Si bien no fue una gran dificultad, se intento que el script de bash, pudiese funcionar tanto en CentOS como en UBUNTU debido a que deseábamos que fuese universal.

Si bien la mayoría de los comandos son iguales, pueden darse algunas diferencias en algunas estructuras, creemos que esto fue cubierto en el código satisfactoriamente.

Password por defecto

Se necesito consultar mediante IAG, un ejemplo de comando para poder agregar un password por defecto en caso de que este no sea colocado por el usuario al momento de ejecutar el script.

Reutilización de ambientes de pruebas en BASH

Al lograr nuestras primeras pruebas exitosas en BASH, aconteció el problema de que el SO se nos llenaba de usuarios de prueba, esto fue solucionado reutilizando la lógica del script principal, mas compacto, en el que pasándole como parámetro el archivo de usuarios, este removía los mismos.

Problema en nombre de base de datos – Python

La base de datos de la aplicación tenía un nombre específico que debía ser el mismo que el utilizado en la configuración, esto retraso el funcionamiento del script hasta que pudimos abrir el .sql e identificar el problema.

Fraccionamiento de Script de Python

Este quizás fue el mayor obstáculo encontrado, se intento en una primera instancia, crear un script separado por cada tarea de Python a realizar (Ec2, RDS, etc) pero traía enormes dificultades a la hora de lidiar con variables que debían ser reutilizadas, por ejemplo el nombre de la base de datos o el password de esta.

Se opto por un script único en un orden específico para solucionar el problema.

ANEXO I – Uso de Inteligencia Artificial Generativa

Para la realización de este trabajo, el uso de **IAG** fue utilizado EXCLUSIVAMENTE para la creación de gráficos específicos o conjunto de comandos básicos con motivo de verificación.

A continuación, se detalla la lista de prompts utilizados.

IAG 1 – Sugerencia de estructura

<https://chatgpt.com/>

Prompt: Sugerir estructura estructura en Bash usando if que retorne ultimo comando y analice si es 0. Si es igual a 0 dejar libre para colocar un comando una acción, sino, devolver un mensaje de éxito.

```
■ Estructura sugerida
bash

# Ejecutar algún comando
comando_que_deseas_ejecutar

# Evaluar el resultado del comando anterior
if [ $? -eq 0 ]; then
    # Acción a realizar si fue exitoso (colocar aquí)
    # Ejemplo: echo "Acción ejecutada correctamente."
else
    echo "El comando no se ejecutó correctamente."
fi
```

IAG 2 – Sugerencia de estructura

<https://chatgpt.com/>

■ Descripción:
chpasswd (abreviatura de *change password*) permite establecer contraseñas para uno o varios usuarios de forma no interactiva, leyendo pares *usuario:contraseña* desde la entrada estándar.
Es ideal para usarlo en scripts, ya que no necesita interacción manual (a diferencia de *passwd*).

★ Ejemplo dentro de un script

bash

```
#!/bin/bash

USUARIO="juan"
CLAVE="1234"

# Crear usuario
useradd "$USUARIO"

# Asignar contraseña automáticamente
echo "$USUARIO:$CLAVE" | chpasswd
```

Prompt: Comando en linux que sirva para colocar un password automáticamente dentro de un script de bash junto a useradd

IAG 3 – Sugerencia de estructura

<https://chatgpt.com/>

```
if [ "$CREAR_HOME" = true ]; then
    # Crear usuario con su directorio home personalizado
    echo "Creando usuario $USUARIO con su directorio home..."
    $CMD --home "/home/$USUARIO" --shell "$SHELL_DEF" --gecos "$COMENTARIO" --disabled-password "$USUARIO"
else
```

Prompt: Ejemplo de bloque en Bash que cree usuarios con el comando adduser, compatible en ubuntu y centos.

Suponer una variable true para crear un home y un false entregaría un home por defecto.

Agregar en el ejemplo, el comentario de usuario

Utilizar if y else preferentemente

IAG 4 – Sugerencia de estructura<https://chatgpt.com/>

Prompt: Como asignar valores por defecto a \$SHELL o \$HOME

```
3 Método clásico con condicional [ -z ]
bash

[ -z "$SHELL" ] && SHELL="/bin/bash"
[ -z "$HOME" ] && HOME="/home/$USUARIO"
```

Qué hace

- Comprueba si la variable está vacía (-z → "zero length").
- Si lo está, asigna el valor indicado.

IAG 5 – Orden de deploy

Nº	Script	Descripción
1	<code>create_sg.py</code>	Crea los Security Groups necesarios (uno para EC2 y otro opcional para RDS).
2	<code>create_s3.py</code>	Crea el bucket S3 y sube los archivos de la aplicación (código fuente, <code>init_db.sql</code> , etc.).
3	<code>create_rds.py</code>	Despliega la base de datos RDS (por ejemplo, MySQL).
4	<code>create_ec2.py</code>	Lanza la instancia EC2 (capa web), con un user-data que: <ul style="list-style-type: none"> • Instala Apache o Nginx • Descarga la app desde S3 • Configura <code>config.php</code> o variables de entorno con el endpoint del RDS • Importa el <code>.sql</code>

<https://chatgpt.com/>

Prompt: En que orden debería implementar los scripts en python con boto3 para deployar una aplicación en dos capas, la idea principal es tener, uno para ec2, otro para base de datos RDS, otro de S3 y uno para security groups

IAG 6 – Subida de S3 a EC2<https://chatgpt.com/>

Prompt: como puedo subir todos los archivos de un S3 a un directorio de una instancia EC2 usando Python y boto3

Tip adicional

Si vas a hacerlo frecuentemente, considera usar el comando nativo de AWS CLI (más eficiente):

```
bash Copiar código

aws s3 sync s3://mi-bucket/datos/ /home/ec2-user/descargas_s3/
```

IAG 6 – Variable de entorno para password

<https://chatgpt.com/>

Prompt: Como agregar una variable de entorno

RDS_ADMIN_PASSWORD en Python para ejecutar en linux

```
python

import boto3
import os

print('Ingresar password maestro de la base:')
RDS_ADMIN_PASSWORD = input().strip()

# Guardar en variable de entorno
os.environ['RDS_ADMIN_PASSWORD'] = RDS_ADMIN_PASSWORD
```

```
try:
    response = ec2.create_security_group(
        GroupName=rds_sg_name,
        Description='Permitir tráfico desde EC2 hacia RDS'
    )
    rds_sg_id = response['GroupId']
    print(f"SG RDS creado: {rds_sg_id}")

    ec2.authorize_security_group_ingress(
        GroupId=rds_sg_id,
        IpPermissions=[
            {
                'IpProtocol': 'tcp',
                'FromPort': 3306,
                'ToPort': 3306,
                'UserIdGroupPairs': [{'GroupId': sg_id}]
            }
        ]
    )
```

IAG 7 – RDS Security Group

<https://chatgpt.com/>

Prompt: Optimizar el bloque de Script dado para el security group de RDS para el puerto 3306

IAG 8 – Wait en instancias

<https://chatgpt.com/>

Prompt: Ejemplo para bloque de espera en RDS o EC2 para que la instancia este lista, usar get_waiter

```
print("Esperando a que RDS esté listo...")
waiter = rds.get_waiter('db_instance_available')
waiter.wait(DBInstanceIdentifier=DB_INSTANCE_ID)
print("RDS listo.")

# Obtener endpoint
rds_info = rds.describe_db_instances(DBInstanceIdentifier=DB_INSTANCE_ID)
RDS_ENDPOINT = rds_info['DBInstances'][0]['Endpoint']['Address']
print(f"Endpoint RDS: {RDS_ENDPOINT}")
```

IAG 9 – Subida de S3 a EC2

<https://chatgpt.com/>

Prompt: Bloque en bash que sirva para esperar que una conexión a una base de datos sea satisfactoria con RDS

```
for i in {1..20}; do
    mysql -h {RDS_ENDPOINT} -u {DB_USER} -p{DB_PASS} -e "SELECT 1;" {DB_NAME} && break
    sleep 5
done
```

IAG 9 – Subida de S3 a EC2

<https://chatgpt.com/>



Prompt: Generar banner que tenga el símbolo y y el texto BASH con algún elemento discreto

IAG 10 – Formato de README.MD

<https://chatgpt.com/>

Prompt: (se copia el contenido del readme.md) Generar formato para README.MD de github sin modificar las imágenes insertadas o modificar el código fuente de BASH o PYTHON comentado. No utilizar iconos, solo formatear de manera profesional estándar el mismo.

BIBLIOGRAFIA