Maestría
en Inteligencia
Analítica de Datos

Taller: Desplegando un tablero en la nube

La entrega de este taller consiste en un reporte y unos archivos de soporte. Cree el archivo de su reporte como un documento de texto en el que pueda fácilmente incorporar capturas de pantalla, textos y similares. Puede ser un archivo de word, libre office, markdown, entre otros.

1. Crear un repositorio con Git

1. En este taller haremos uso de Git como herramienta de versionamiento de código y de una maquina virtual para desplegar un tablero en la nube.
2. Empiece por descargar Git en su máquina local a través del siguiente link:
   `https://git-scm.com/downloads`
   asegurándose de instalar la versión correspondiente al sistema operativo local. Durante la instalación, utilice todas las opciones que la herramienta le recomiende por defecto.
3. Después de instalar Git en su máquina local, crearemos una carpeta dedicada a todo el material correspondiente a este taller. Después de crearla, abra la terminal y navegue a la carpeta que acaba de crear. Una vez esté adentro de esta carpeta, emita el comando:

   ```
   git init
   ```

   Este comando inicializa un repositorio en su carpeta actual.
4. Antes de comenzar a usar Git, realizaremos una configuración inicial con un nombre de usuario y su correo. Para esto emita los siguientes comandos, reemplazando `'your-username'` por su nombre de usuario y `'your-email'` por su correo:

   ```
   git config --global user.name 'your-username'
   git config --global user.email 'your-email'
   ```

5. Ahora, utilizando el explorador de archivos, agregaremos todos los archivos necesarios para el funcionamiento del tablero a esta carpeta. Estos archivos los encontrará disponibles en Coursera. Esto incluye el archivo app.py, los datos en datos_energia.csv y la carpeta assets, que contiene dos archivos de tipo css.
6. Una vez estén todos los archivos en la carpeta, regrese a la terminal y asegúrese de estar ubicado en dicha carpeta. Ahora deberá realizar su primer commit. Para hacer esto, como primer paso agregar los archivos a considerar en el commit con los comandos:


```
git add app.py datos_energia.csv

git add assets/base.css assets/clinical-analytics.css
```

A esta etapa se le denomina *staging*.

7. Después de haber agregado los archivos a considerar en la etapa de *staging*, realizamos el commit con el comando:

   ```
   git commit -m "mi primer commit"
   ```

   El texto entre comillas debe ser un mensaje descriptivo del commit que se está realizando. En la terminal deberá observar una respuesta similar a esta:

   ```
   (base) juancamilopico@192 Taller dash 1 % git commit -m "mi primer commit"
   [master (root-commit) e506748] mi primer commit
    4 files changed, 11481 insertions(+)
    create mode 100755 app.py
    create mode 100755 assets/base.css
    create mode 100755 assets/clinical-analytics.css
    create mode 100644 datos_energia.csv
   (base) juancamilopico@192 Taller dash 1 %
   ```

   Figura 1: Mi primer commit.

8. Como siguiente paso, haremos un cambio en el archivo app.py. El código en este archivo define la lógica de un tablero que permite visualizar un pronóstico de producción energética. Para esto, el tablero debe cargar el archivo .csv que contiene los datos históricos de producción junto con el prónostico de energía. En el código encontrará la función `load_data()`, donde usted deberá desarrollar el código para cargar el archivo `datos_energia.csv` como un DataFrame. Para el funcionamiento correcto del tablero, tenga en cuenta que la columna con la fecha debe estar en formato `datetime` (para esto puede hacer uso de la función `to_datetime()` del paquete Pandas) y que el índice del DataFrame deberá ser la fecha. ***Hint:*** **Tenga en cuenta que la función deberá retornar el dataframe.**
9. Para verificar el funcionamiento de su código, instale los paquetes necesarios en su máquina local y ejecute el código en su editor de código preferido (sin embargo, no lo ejecute en un cuaderno de Jupyter). Al ejecutarlo, en la terminal de su editor de código recibirá un mensaje que contiene la dirección en la que esta corriendo su tablero.

Figura 2: Dirección del tablero.

Entre a esta dirección utilizando su navegador para visualizar el tablero (Figura 3)

Figura 3: Tablero.

10. Para terminar la ejecución del tablero, emita en la terminal donde se está ejecutando su código el comando `CTRL + c`.
11. Ahora que le realizamos cambios al archivo app.py, haremos un nuevo commit. Para esto debemos volver a agregar los archivos a considerar con el comando:

    ```
    git add app.py
    ```

    y haremos el commit con el comando:

    ```
    git commit -m "nueva funcion para cargar datos"
    ```

12. Para observar el historial de commits, podemos utilizar el comando:

    ```
    git log
    ```

    Con este comando debería observar un historial similar a este:


```
(base) juancamilopico@192 Taller dash 1 % git log
commit 6f112f573ed9eda374e8193a77ce932faa3f293f (HEAD -> master)
Author: Juan Camilo Pico <juanca.pico@hotmail.com>
Date:   Wed Jul 19 00:42:42 2023 -0500

    nueva funcion para cargar datos

commit e506748ac267bf447110f9c6098448fc5e304490
Author: Juan Camilo Pico <juanca.pico@hotmail.com>
Date:   Wed Jul 19 00:21:38 2023 -0500

    mi primer commit
(base) juancamilopico@192 Taller dash 1 %
```

Figura 4: Historial de cambios.

2. Subir el tablero a un repositorio remoto

1. Hasta este momento, el tablero solo ha sido lanzado de forma local. Para poder desplegar su tablero en la nube, debemos montar el código en un repositorio remoto al que se pueda acceder desde la maquina virtual.
2. Para esto haremos uso de Github, un servicio de repositorios en la nube que permite compartir y colaborar en repositorios. Si no tiene una cuenta en Github, empiece por crear una (https://github.com/).
3. Ahora cree un nuevo repositorio público en Github sin readme y sin .gitignore. Una vez creado, copie la URL (HTTPS) del repositorio:

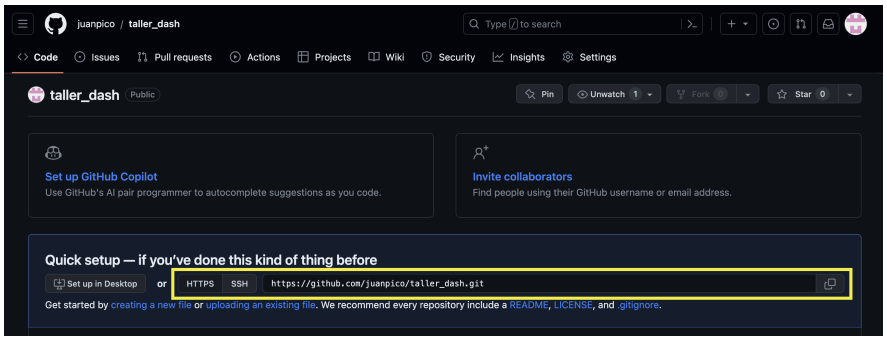


Figura 5: URL del repositorio remoto.

4. Ahora, en la terminal de su maquina local emita el comando:

   ```
   git remote add origin URL
   ```

   donde `URL` hace referencia a la URL que copió en el paso anterior. Este comando conecta su repositorio local con el repositorio remoto.
5. Para enviar su rama master local a la rama master remota, emita el siguiente comando:

```
git push origin master
```

En consola, recibirás un mensaje solicitando el usuario de tu usuario en github:

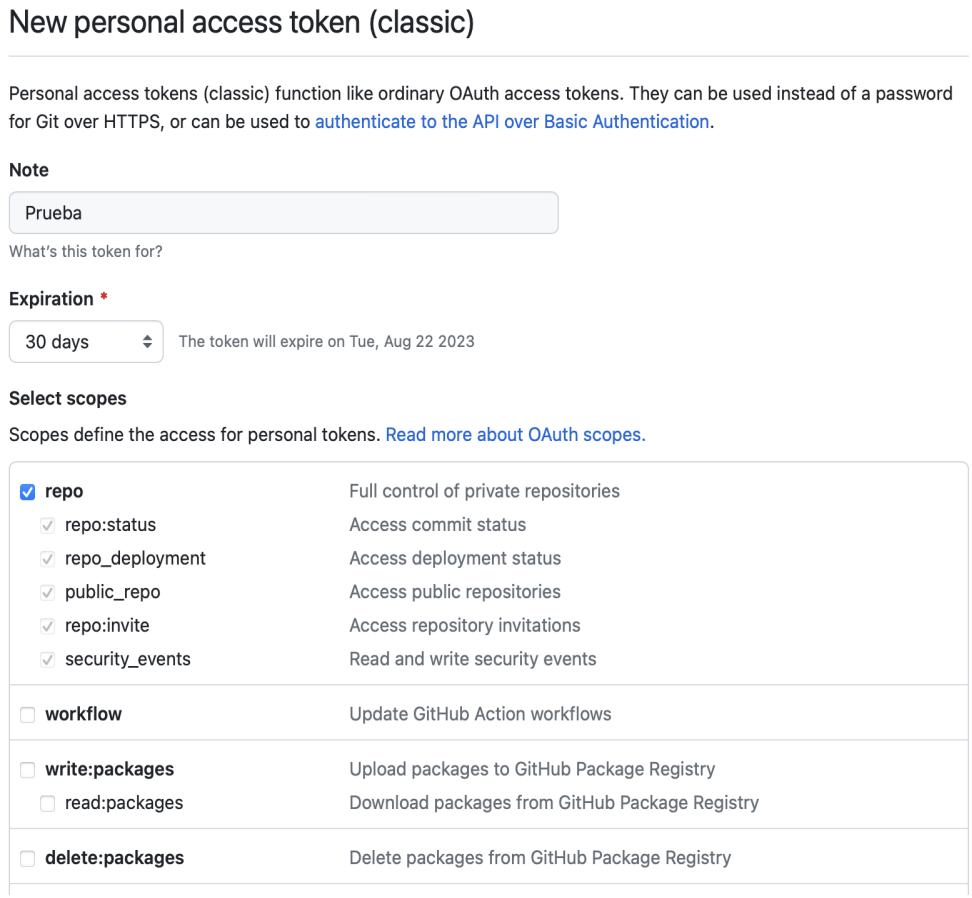
```
Username for 'https://github.com':
```

Este mensaje solicita que ingrese su nombre de usuario de GitHub para autenticar tus operaciones Git cuando interactúa con el repositorio remoto a través del protocolo HTTPS.
Posteriormente, le solicitará la contraseña de su cuenta en github:

```
Password for 'https://user@github.com'
```

Sin embargo, el soporte para la autenticación con contraseña fue eliminado a partir del 13 de agosto de 2021. Ahora, en lugar de usar una contraseña, deberá generar un token de acceso para poder acceder a su cuenta.
Para esto, debe seguir los siguientes pasos:

a) En github, vaya a *settings*
b) *Developer settings*
c) *Personal access token*
d) *Tokens (classic)*
e) *Generate new token (classic)*
f) Asigne un nombre
g) Defina una duración
h) Seleccione "repo"



i) Click en *Generate token*

j) Copie el *token*

Si le solicita contraseña, ingrese el token.

Después de haber emitido el comando `push` exitosamente, deberá poder observar una respuesta como la siguiente:

```
(base) juancamilopico@192 Taller dash 1 % git push origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 63.14 KiB | 7.89 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/juanpico/taller_dash.git
 * [new branch]      master -> master
```

Figura 6: git push origin master.

6. Al ingresar al repositorio remoto en Github, deberá poder observar los archivos junto con el historial de la rama. **Incluya en su reporte el enlace del repositorio y un *screenshot* de su repositorio con los archivos requeridos.**

3. Configurar la instancia

1. En esta sección haremos uso de una maquina virtual para lanzar el tablero. Para esto, siga los pasos mencionados en la *Guia de tecnologías* para desplegar y acceder a una máquina virtual con Linux. **Incluya en su reporte un *screenshot* de la conexión a la máquina virtual.**
2. Una vez haya accedido a su maquina virtual, se debe actualizar el sistema con el siguiente comando:

   ```
   sudo yum update -y
   ```

   `sudo` emite comandos como administrador del sistema, tenga mucho cuidado al usarlo. `yum` es el administrador de paquetes/programas de varias distribuciones de Linux, entre ellas las de Amazon. La bandera `-y` genera la respuesta `yes` a todas las confirmaciones que surjan con el comando de actualización.
3. Verifique que tiene instalado Python 3 (versión 3.9) con el comando

   ```
   python3 --version
   ```

4. Instale pip3 para administrar paquetes de python

   ```
   sudo yum install python3-pip
   ```

   Verifique la versión de pip3

   ```
   pip3 --version
   ```

5. Instale pandas con pip3

   ```
   pip3 install pandas
   ```

6. Instale el paquete dash

   ```
   pip3 install dash
   ```

7. Instale el paquete gunicorn para python

   ```
   pip3 install gunicorn
   ```

8. Instale Git

   ```
   sudo yum install git -y
   ```

9. Para traer los archivos del repositorio remoto a la maquina virtual, se debe clonar el repositorio remoto con el siguiente comando:

   ```
   git clone URL
   ```

   donde `URL` es la URL (HTTPS) del repositorio remoto. Puede copiar esta URL del repositorio remoto haciendo click en el botón verde *Code*:

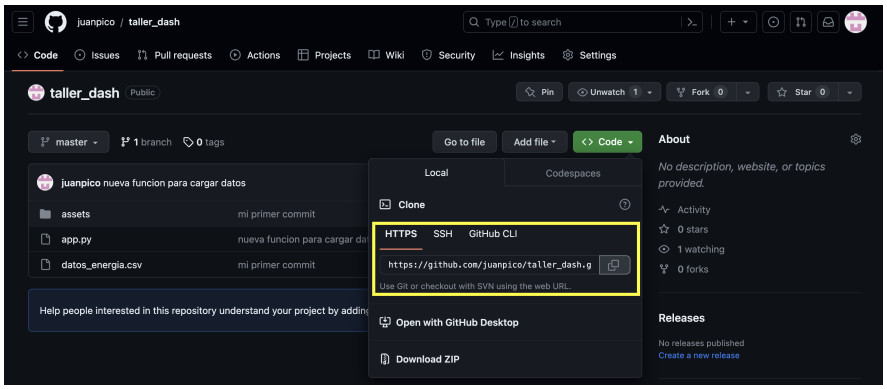


Figura 7: URL (HTTPS) del repositorio remoto.

10. Después de clonar exitosamente el repositorio remoto, debe obtener la siguiente respuesta en la terminal:

```
[ec2-user@ip-172-31-86-144 ~]$ git clone https://github.com/juanpico/taller_dash.git
Cloning into 'taller_dash'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 1), reused 10 (delta 1), pack-reused 0
Receiving objects: 100% (10/10), 63.14 KiB | 10.52 MiB/s, done.
Resolving deltas: 100% (1/1), done.
[ec2-user@ip-172-31-86-144 ~]$
```

Figura 8: git clone URL

Adicionalmente, utilizando el comando `ls` podrá observar que ahora en la maquina virtual se encuentra una carpeta con el nombre del repositorio remoto. Al entrar a la carpeta utilizando el comando `cd` podrá observar todos los archivos del repositorio remoto. **Incluya en su reporte un *screenshot* de la carpeta clonada en su máquina virtual.**

4. Lanzar el tablero en la nube

1. Teniendo en la máquina virtual los archivos necesarios, solo faltan los últimos pasos para desplegar el tablero. Al archivo app.py le debemos realizar una pequeña modificación antes de poder ejecutar correctamente el tablero en la máquina virtual, para lo cual usaremos el comando `nano`. En su máquina virtual, asegúrese de estar ubicado en la carpeta que contiene el archivo app.py y utilice el comando:

   ```
   nano app.py
   ```

   Esto abre un editor de texto. Usando las flechas navegue hasta la parte inferior del archivo y modifique la línea

   ```
   app.run_server(debug=True)
   ```

   para incluir el argumento adicional `host`.

   ```
   app.run_server(host="0.0.0.0", debug=True)
   ```

   Para cerrar el archivo `CTRL+X`, escriba `Y` para confirmar que quiere guardar los cambios, y regresará a la terminal principal.
   Para verificar el cambio puede usar el comando

   ```
   cat app.py
   ```

   que le permite observar el archivo rápidamente, sin acceder a modificarlo.
2. Guarde el cambio realizado en el archivo `app.py` en el repositorio de la maquina virtual y asegúrese de mandar el cambio al repositorio remoto. Recuerde los tres pasos: *staging*, *commit* y *push*. (Nota: Al momento de hacer el push, será necesario que utilice un Personal Access Token siguiendo los pasos de la sección 2, paso 5.)
3. Para que pueda acceder al tablero debe habilitar unos permisos de seguridad en AWS. En la consola de EC2, panel izquierdo, seleccione Security Groups en redes y seguridad. Allí encontrará el grupo de seguridad correspondiente a su máquina virtual, selecciónelo.
4. En la parte inferior verá las reglas de entrada, que definen cómo puede entrar tráfico a la instancia. Click en Editar reglas de entrada. Note que solo tiene una regla, que permite tráfico por el puerto 22, el cuál usamos para conectarnos a la terminal por SSH.
5. Click en Agregar regla. En Tipo seleccione TCP personalizado, en Intervalo de puertos marque 8050, en Origin seleccione Anywhere IPv4. Click en Guardar regla.
6. Ya estamos listos para lanzar el tablero en el servidor. En la máquina virtual corra la aplicación con

   ```
   python3 app.py
   ```

En su navegador verifique que la aplicación esté corriendo y disponible visitando la dirección

`http://IP:8050`

donde `IP` es la dirección IP (v4) pública de su máquina virtual.**Incluya en su reporte un *screenshot* del tablero corriendo y el link del tablero.**

7. Al terminar esta guía diríjase a la consola de EC2 de AWS, seleccione su máquina virtual y en el menú Actions seleccione Terminate, para terminar la máquina completamente. Si no la termina, se seguirán cobrando cargos a su cuenta de AWS Academy.