Pre-requisitos

1. Para esta sesión va a requerir una cuenta de Amazon Web Services - AWS. Para esto hay varias opciones:

   a) **Usar la cuenta de AWS Academy** enviada a su correo por el instructor, y acceder al curso **Learner Lab** (recomendado).

   b) Utilizar una cuenta propia ya creada.

   c) Crear una cuenta nueva. En este caso recuerde que requiere ingresar datos de una tarjeta de crédito. Usaremos recursos disponibles en la capa gratuita (https://aws.amazon.com/free/), pero es posible que se generen algunos costos menores.

2. **Nota:** este taller se debe realizar en parejas. Defina un nombre para su grupo, defínalo claramente en su **reporte** y use este nombre como parte inicial de **todos los recursos que cree en la nube**.

3. **Nota 2:** 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.

4. Asegúrese de tener instalado Python en versión 3.11 o superior. Para este taller usaremos Python y en particular las librerías *dash*, *plotly* y *pandas*. Puede instalar las librerías con el comando

   ```
   pip install dash plotly pandas
   ```

   o si prefiere repositorios de anaconda use

   ```
   conda install dash plotly pandas
   ```

1. Tableros locales

1.1. Un primer tablero en Dash

Dash es un framework para el desarrollo de aplicaciones web en python, desarrollado por Plotly y construido sobre Flask. En particular, Dash ofrece facilidades para crear tableros para la visualización de datos interactivos y dinámicos que proveen una buena interfaz al usuario del tablero.

Dash interactúa además con Pandas, la librería estándar para la manipulación de datos en python. Esto permite fácilmente cargar datos desde una fuente externa y generar visualizaciones de gran calidad.

Al ser un framework web, el resultado es una página web que puede accederse usando navegadores estándar, ya sea localmente o en un servidor.

Como parte de este taller encontrará algunos ejemplos sencillos de tableros en Dash, que nos sirven como punto de partida.

1. Abra el archivo `app1.py` en un editor de python como Visual Studio code. Para abrir los archivos de base de este taller **evite** usar soluciones orientadas a cuadernos como jupyter.

2. Note que en la parte final del archivo se define qué hacer cuando se ejecuta el archivo. En este caso se llama al objeto `app` y se ejecuta su método `run_server`. El objeto `app` se crea en las primeras líneas del script. En su **reporte** incluya el código con el que se crea el objeto `app`.

3. Después de crear el objeto `app` (y extraer el atributo `server`) el programa define un dataframe de pandas en el que se almacena un conjunto de datos.

4. Ejecute la aplicación usando su editor o ejecutando en consola

   ```
   python app1.py
   ```

   o

   ```
   python3 app1.py
   ```

   dependiendo de su sistema operativo e instalación de python.

5. En su navegador (Chrome, Firefox, Safari, Edge) visite el servidor local usando el puerto 8050, es decir, vaya a la página

   ```
   http://127.0.0.1:8050/
   ```

   En su **reporte** incluya un pantallazo donde aparezca el código en ejecución y la página cargada, lado a lado.

6. Considerando la aplicación resultante, describa en su **reporte** qué hace el comando

   ```
   fig = px.bar(df, x="Fiebre", y="Casos", color="Diagnostico",
      barmode="group")
   ```

   También puede apoyarse en la documentación de plotly:

   - `https://plotly.com/python-api-reference/generated/plotly.express.bar.html`
   - `https://plotly.com/python/bar-charts/`

7. La siguiente (y última sección de código) crea el documento html que es renderizado por el navegador. Note que se crea definiendo secciones (Div) y título (H1), entre otros elementos html. En su reporte describa brevemente qué hacen las líneas

```
dcc.Graph(
  id='example-graph',
  figure=fig
),
```

y

```
html.Div(
  className="Columnas",
  children=[
    html.Ul(id='my-list', children=[html.Li(i) for i in df.columns
   ])
  ],
)
```

8. Realice cambios en los datos: modifique los nombres de las columnas y los valores que toman. Realice todos los cambios necesarios hasta obtener una nueva versión del tablero con datos de su elección. Incluya su archivo modificado como **soporte** de su entrega. Incluya un pantallazo de la aplicación en ejecución en su **reporte**.

1.2. Callbacks

Un mecanismo muy útil de Dash es el uso de callbacks para modificar dinámicamente elementos del tablero. Veamos un ejemplo sencillo de callbacks.

1. Abra el archivo `app2.py` en un editor de python.

2. Note que el código tiene muchas similitudes con el anterior, incluyendo el main y la definición de los objetos `app` y `server`.

3. La gran diferencia de esta nueva aplicación radica en la función

   ```
   def update_output_div(input_value):
     return 'Output: {}'.format(input_value)
   ```

   Note que a esta función se le agrega un *decorador*, el cual extiende la función asociando sus entradas y salidas con elementos del documento HTML

   ```
   @app.callback(
     Output(component_id='my-output', component_property='children'),
     [Input(component_id='my-input', component_property='value')]
   )
   ```

4. En su **reporte** describa qué hace la función `update_output_div`, considerando el decorador y qué elementos el documento HTML asocia con la función.

5. Modifique la frase que retorna la función, por ejemplo incluyendo un texto personalizado, además del texto que ingresa el usuario. Incluya el código modificado como **soporte** de su entrega. Incluya un pantallazo de la aplicación modificada en ejecución en su **reporte**.

1.3. Más visualizaciones e interacciones

Hasta el momento hemos visto un tipo de visualización muy sencilla. Ahora consideramos una visualización más elaborada.

1. Abra el archivo `app3.py` en un editor de python.
2. Note que el código tiene muchas similitudes con el anterior, no solo en el main, app y server, sino también en la definición de layout y una función con callback.
3. Ejecute la aplicación.
4. Describa en su **reporte** el layout de esta aplicación. ¿Qué elementos tiene?
5. En su **reporte** describa qué hace la función `update_figure` y su decorador, considerando los elementos del documento HTML asociados.
6. Ahora realice *una nueva visualización* de interés. Considere las opciones de gráficas que ofrece plotly.express `https://plotly.com/python/plotly-express/`. También puede ser de interés la Coda a esta sección. Incluya títulos y textos que describan las visualizaciones incluidas.
7. Incluya el código modificado como **soporte** de su entrega. Incluya un pantallazo de la aplicación modificada en ejecución en su **reporte**.
8. A manera de ejemplo adicional incluimos el archivo `app4.py` donde podrá ver elementos adicionales y una actualización un poco más elaborada de una gráfica y sus elementos usando callbacks.
9. Para terminar esta sección descomprima el archivo data-food-consumption.zip que encontrará en Bloque Neón. Allí encontrará una app tomada del repositorio `https://github.com/plotly/dash-sample-apps`, y en particular de la sección de releases `https://github.com/plotly/dash-sample-apps/releases`. Esta aplicación es específicamente `https://github.com/plotly/dash-sample-apps/releases/download/v0.23.5/dash-food-consumption.zip` con algunas actualizaciones para ejecutar sin errores/warnings en versiones recientes de dash/pandas/statsmodels. Explore la estructura de carpetas y los archivos. Ejecute la aplicación y explore el tablero resultante. En su **reporte** explique **cada uno** de los callbacks que tiene la aplicación, específicamente cuál es su input, su ejecución y su output.

2. Primeros pasos en AWS EC2: lanzar una máquina virtual

1. En esta sección desplegaremos una máquina virtual Linux usando el servicio EC2 de AWS, el cual es de tipo IaaS. Usaremos esta máquina para servir un tablero Dash.
2. En AWS Academy encontrará el curso Learner Lab. Ingrese al curso y en la sección de Contenidos seleccione el módulo Learner Lab.

3. Para empezar de click en el botón AWS en la parte superior del laboratorio, lo que le permitirá ingresar a la consola de AWS.

4. En la parte superior de la consola de AWS hay un filtro de servicios, busque allí EC2, el servicio de máquinas virtuales de AWS.

5. En la consola de EC2, en el panel izquierdo seleccione *Instancias* y click en *Lanzar instancias*. Lance una instancia con las siguientes características:

   a) Nombre: asigne un nombre adecuado.

   b) Imagen (Amazon Machine Image - AMI): Amazon Linux (note que hay muchas opciones).

   c) Tipo de instancia: t2.micro (apta para la capa gratuita).

   d) Par de claves: Cree un nuevo par de claves

      1) Asigne un nombre adecuado.
      2) Seleccione RSA como tipo y .pem como formato de archivo.
      3) Asegúrese de guardar la llave .pem localmente en un sitio de fácil acceso (idealmente en una carpeta creada para desarrollar este taller). En adelante lo llamaremos llave.pem.

   e) En la configuración de red deje los valores por defecto (esto creará un grupo de seguridad con permisos de conexión por SSH, puerto 22).

   f) Deje la configuración de almacenamiento (8 GB de disco) y los detalles avanzados por defecto.

   g) Click en lanzar instancia.

   h) Regrese a la consola de EC2 y en el ítem Instancias debe poder ver la instancia en proceso de inicio.

   i) En su **reporte** tome un snapshot del tablero EC2 que muestre su instancia en ejecución.

   j) Note que el campo Comprobación indica que la máquina está en proceso de inicialización, luego realiza dos chequeos y luego ya aparece como lista para usar.

6. En la consola de EC2 seleccione su instancia y copie algunos datos en su **reporte**:

   a) Dirección IP v4 pública.

   b) Dirección IP v4 privada.

   c) Tipo de instancia.

   d) Plataforma y Detalles de la plataforma.

   e) Tipo de virtualización.

   f) Número de CPU virtuales.

   g) Zona de disponibilidad (lo encuentra en la pestaña Redes).

h) ID de volumen (lo encuentra en la pestaña Almacenamiento).

i) Tome un pantallazo de la pestaña Monitoreo.

7. En la consola de EC2 seleccione su instancia y copie la dirección IP (v4) pública.

8. **Conéctese** a la instancia:

   a) Abra una terminal: tecla windows, escriba *cmd* y ENTER)

   b) En la terminal emita el comando

   ```
   ssh -i /path/to/llave.pem ec2-user@IP
   ```

   donde */path/to/* se refiere a la ubicación del archivo *llave.pem* que descargó, e IP es la dirección IPv4 pública de la instancia EC2 que lanzó. Si prefiere, en la terminal puede navegar a la ubicación del archivo llave.pem y emitir el comando

   ```
   ssh -i llave.pem ec2-user@IP
   ```

   c) A la pregunta
   `Are you sure you want to continue connecting (yes/no/[fingerprint])?`
   Responda `yes` y ENTER.

   d) En este momento debe estar **en la terminal de la instancia EC2** que creó. Es decir, los comandos que emita en esta terminal se ejecutan en su máquina virtual (servidor en la nube).

   e) Tome un pantallazo de la terminal e inclúyalo en su **reporte**.

3. Configurar la instancia y lanzar el tablero

Los siguientes comandos se deben ejecutar en la terminal conectada a instancia en la nube, a menos que se indique algo diferente.

1. Actualice 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.

2. Verifique que tiene instalado Python 3 (al menos versión 3.7) con el comando

   ```
   python3 --version
   ```

3. Instale pip3 para administrar paquetes de python

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

   Verifique la versión de pip3

```
pip3 --version
```

4. Instale pandas con pip3

   ```
   pip3 install pandas
   ```

5. Instale el paquete dash

   ```
   pip3 install dash
   ```

6. Instale el paquete gunicorn para python

   ```
   pip3 install gunicorn
   ```

7. Como parte de este taller encontrará el archivo app1.py. Queremos subir este archivo a la máquina en la nube. **En su máquina local, abra otra terminal** y navegue al sitio donde tiene este archivo (en adelante supongo que este archivo está en la misma carpeta que llave.pem). Para copiar ese archivo a la máquina en la nube, use el comando *scp* así

   ```
   scp -i llave.pem app1.py ec2-user@IP:/home/ec2-user
   ```

   que copia el archivo a la carpeta *home* del usuario *ec2-user* en su máquina virtual.

8. Verifique que se copió el archivo en su máquina virtual listando los contenidos de la carpeta /home/ec2-user con el comando

   ```
   ls
   ```

   Incluya un pantallazo de la terminal mostrando el archivo en su **reporte**.

9. En la máquina virtual cree una copia del archivo app1.py con nombre app.py

   ```
   cp app1.py app.py
   ```

   Verifique la creación del archivo con el comando `ls`. Si quiere más detalles puede usar la bandera -la

   ```
   ls -la
   ```

10. Necesitamos realizar una pequeña modificación sobre el archivo app.py, para lo cual usaremos el comando *nano*. En su máquina virtual use 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 un argumento adicional y que quede

    ```
    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 ENTER para regresar 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. Incluya un pantallazo de la terminal mostrando la salida de este comando en su **reporte**.

11. En la consola de EC2, panel izquierdo, seleccione Security Groups en redes y seguridad. Allí encontrará su grupo de seguridad, selecciónelo.

12. 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 cual usamos para conectarnos a la terminal por SSH.

13. Click en Agregar regla. En Tipo seleccione TCP personalizado, en Intervalo de puertos marque 8050, en Origin seleccione Anywhere IPv4. Click en Guardar regla. Incluya un pantallazo del grupo de seguridad modificado en su **reporte**.

14. 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 está corriendo y disponible visitando la dirección

    ```
    http://IP:8050
    ```

    Incluya un pantallazo de la aplicación ejecutándose en su **reporte**. Por el canal de **Slack** incluya la URL completa (enlace) para acceder a su tablero.

15. Al terminar su taller seleccione la máquina en la consola, y en el menú Actions seleccione Terminate, para terminar la máquina completamente. Si no la termina, se seguirán cobrando cargos a su cuenta.

4. En caso de problemas

1. Si tiene problemas para **conectarse a la instancia por la terminal** puede ir a la consola de AWS EC2, seleccionar la máquina creada y dar click en Conectar.

   a) Se abre un nuevo panel con la información de la máquina y del usuario (ec2-user). Click en Conectar.

   b) Tras un instante se abrirá una terminal en el navegador. Puede continuar con la realización del taller desde allí. No es la interfaz ideal pero es una buena alternativa si hay problemas de conexión por terminal.

   c) Para subir el archivo del tablero a la máquina puede usar el comando

   ```
   wget juanfperez.com/actd/24/app1.py
   ```