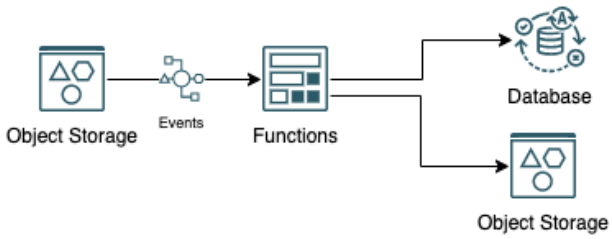
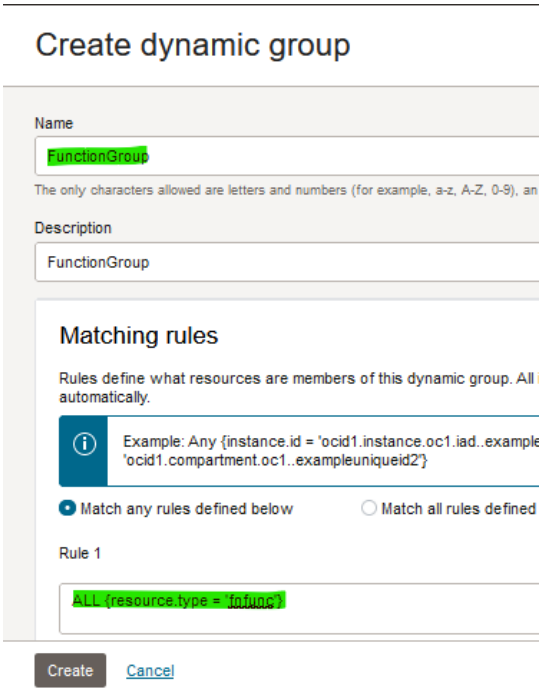
LABORATORIO CONSTRUCCIÓN DE APLICACIONES ORACLE CLOUD SERVERLESS

Este laboratorio esta enfocado a que puedas construir una aplicación Serverless, que realice la lectura, procesamiento y guardado de un archivo separado por comas(*.csv) en una base de datos autonoma, dicha aplicación tendra la siguiente aquitectura:



PRE-REQUISITOS

1. Creación de grupo dinámico con el nombre **FunctionGroup** y políticas de seguridad para manipulación de la infraestructura OCI por parte de la función serverless.

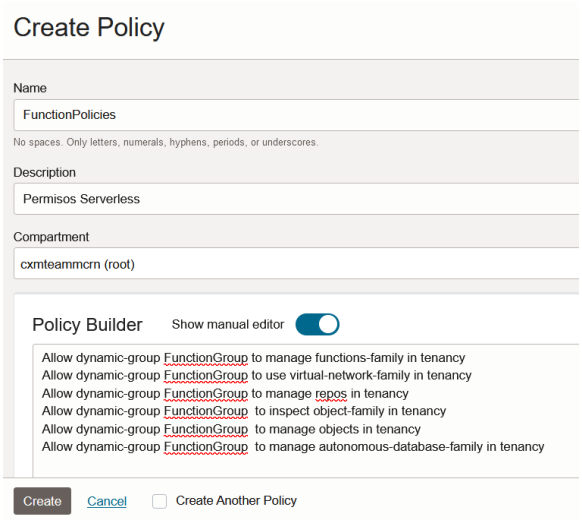


Estableciendo la siguiente regla para el grupo dinámico que permitirá a las funciones serverless acceder a los recursos OCI, a nivel de seguridad

ALL {resource.type = 'fnfunc'}

2. Definición de políticas IAM para la para manipulación de la infraestructura OCI por parte de la función serverless.

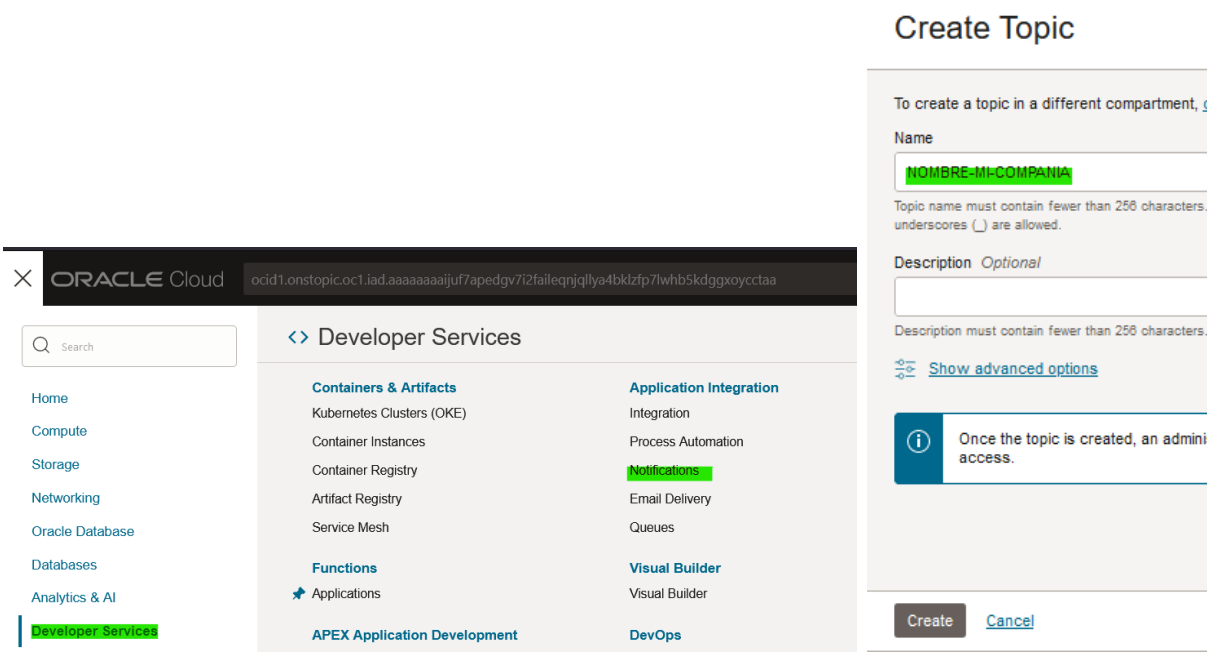
   IMPORTANTE: Estas políticas deben ser definidas a nivel del compartment ROOT



Allow dynamic-group **FunctionGroup** to manage functions-family in tenancy
Allow dynamic-group **FunctionGroup** to use virtual-network-family in tenancy
Allow dynamic-group **FunctionGroup** to manage repos in tenancy
Allow dynamic-group **FunctionGroup** to inspect object-family in tenancy
Allow dynamic-group **FunctionGroup** to manage objects in tenancy
Allow dynamic-group **FunctionGroup** to manage autonomous-database-family in tenancy
Allow dynamic-group **FunctionGroup** to use ons-topics in tenancy

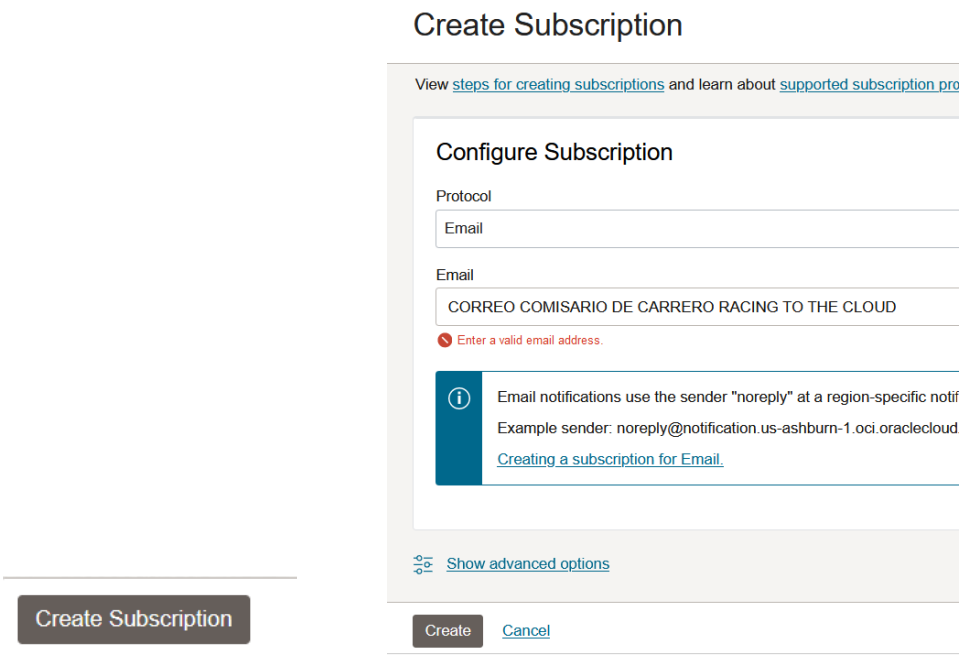
TOPIC PARA NOTIFICACIONES

En el menu general debemos ir al menú de notificaciones, y crear un topic con el nombre de empresa
Ejemplo: **ACME CORP**

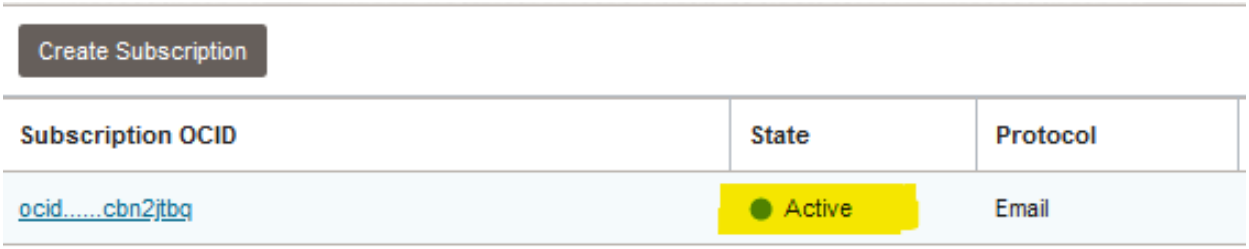


Copia el OCID del topic lo necesitaras más adelante

Dentro del TOPIC se deberá crear una suscripción al correo del **COMISARIO DE CARRERA (nataly.diaz@oracle.com ó jose.borda@oracle.com),** a este correo llegarán las notificaciones que serán fundamental para los **puntajes y clasificación**.

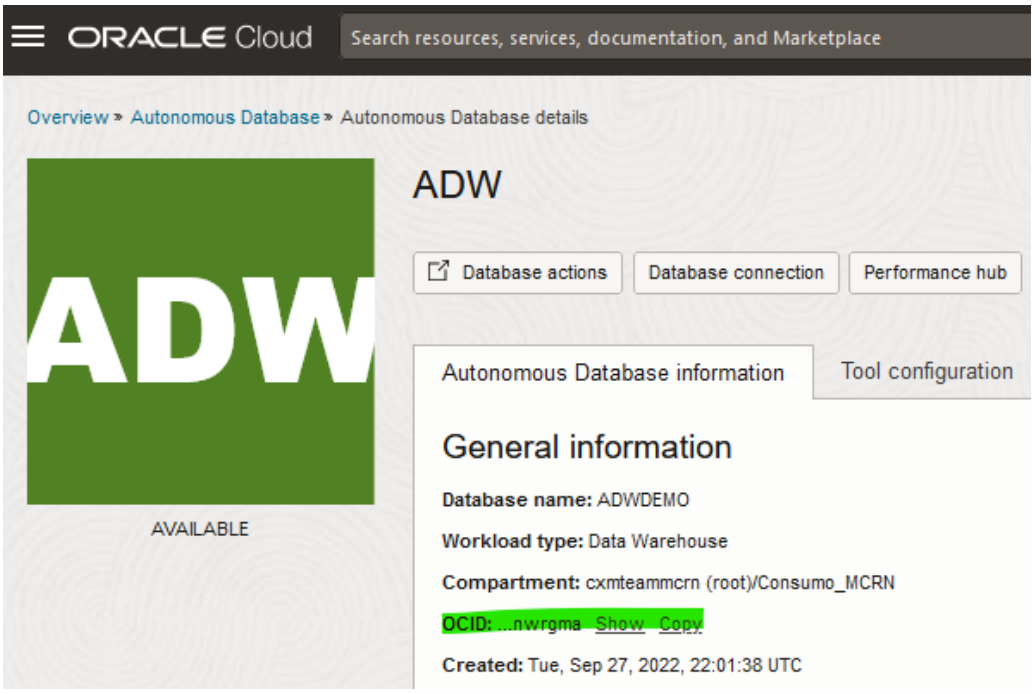


El comisario de carrera deberá **aceptar** la suscripción



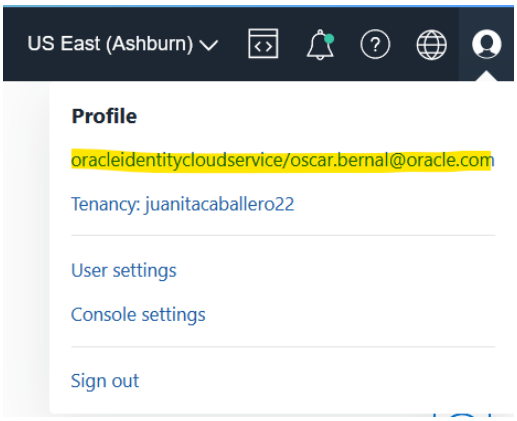
BASE DE DATOS

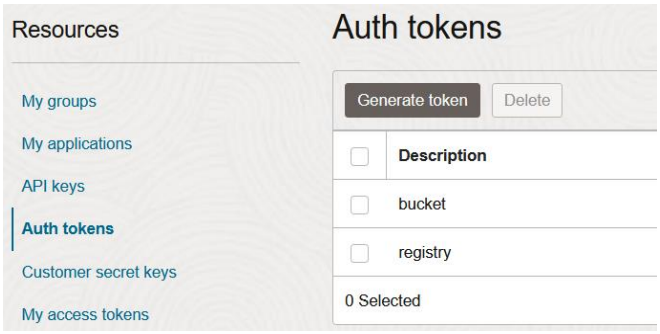
Crear o usar una base de datos autónoma existente, para esto **copia el OCID de esta base de datos lo necesitaras más adelante**

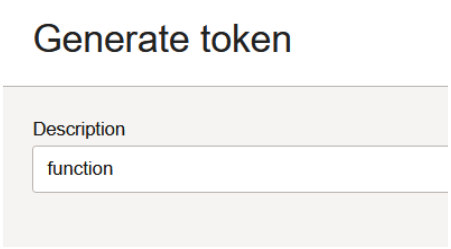


GENERAR AUTH TOKEN

En la esquina superior derecha del portal encontremos el **profile** del usuario donde podremos generar el token de autenticación:

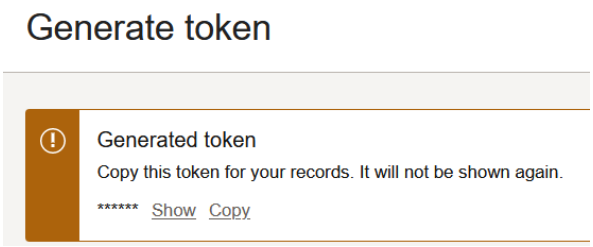






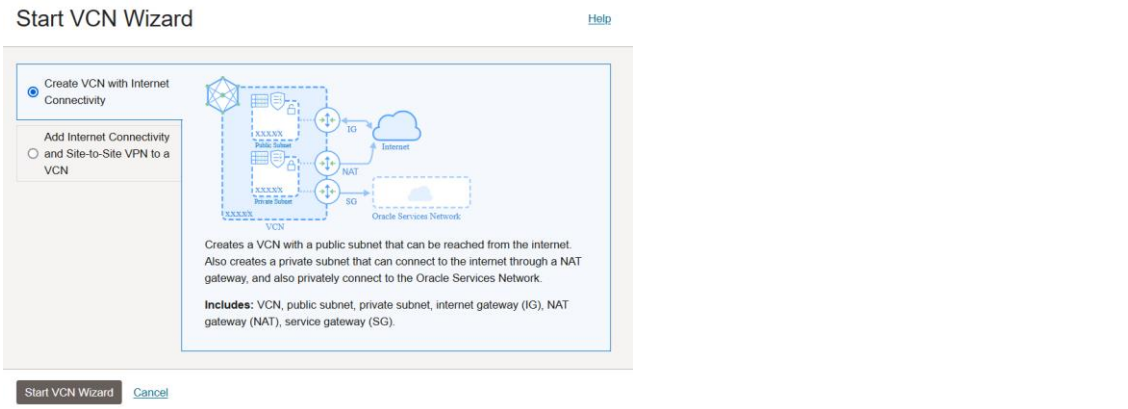
Guardar el valor generado por la consola el cual debemos usar en los pasos posteriores

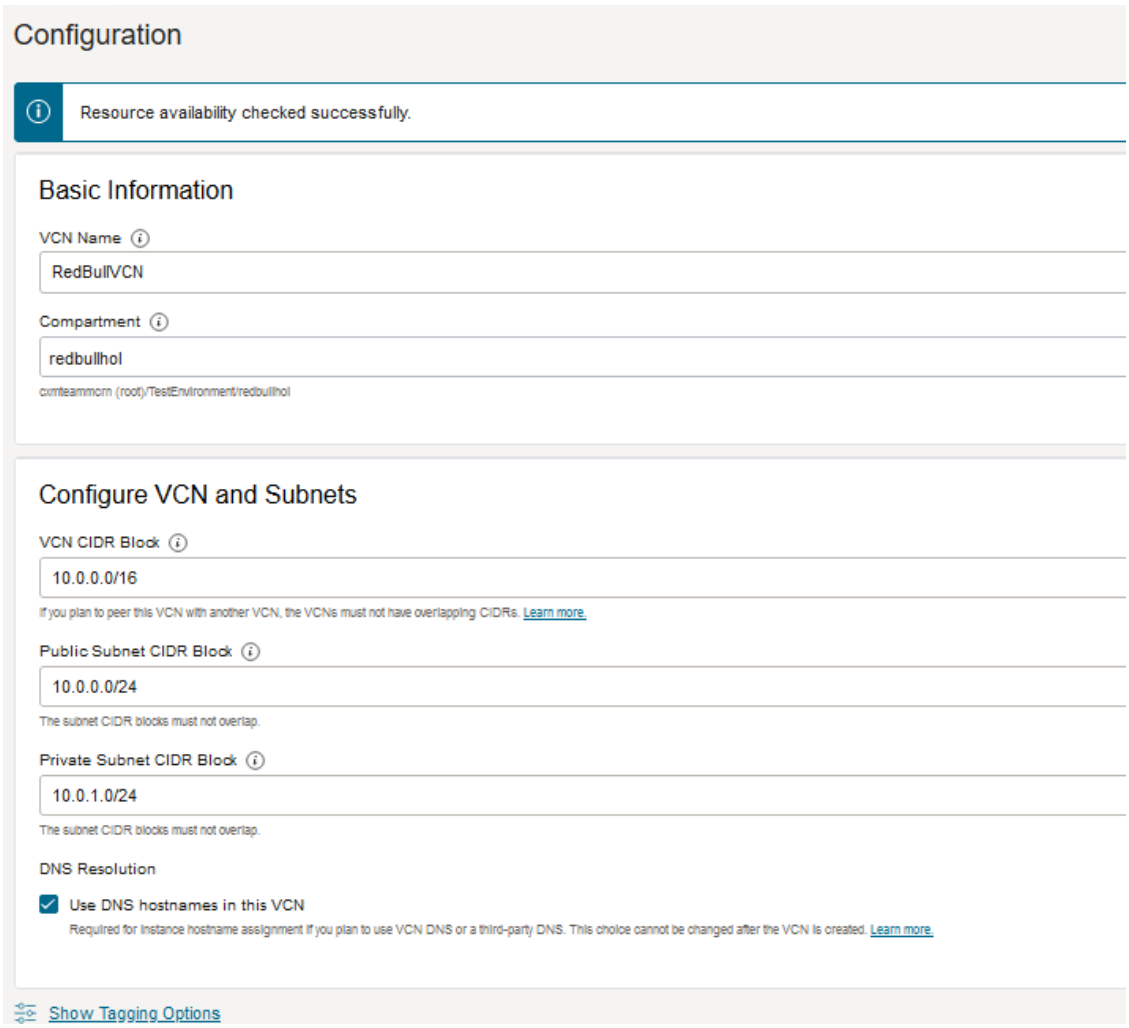
Ejemplo: v#1iD<8Ycx+)Z+XUR5av



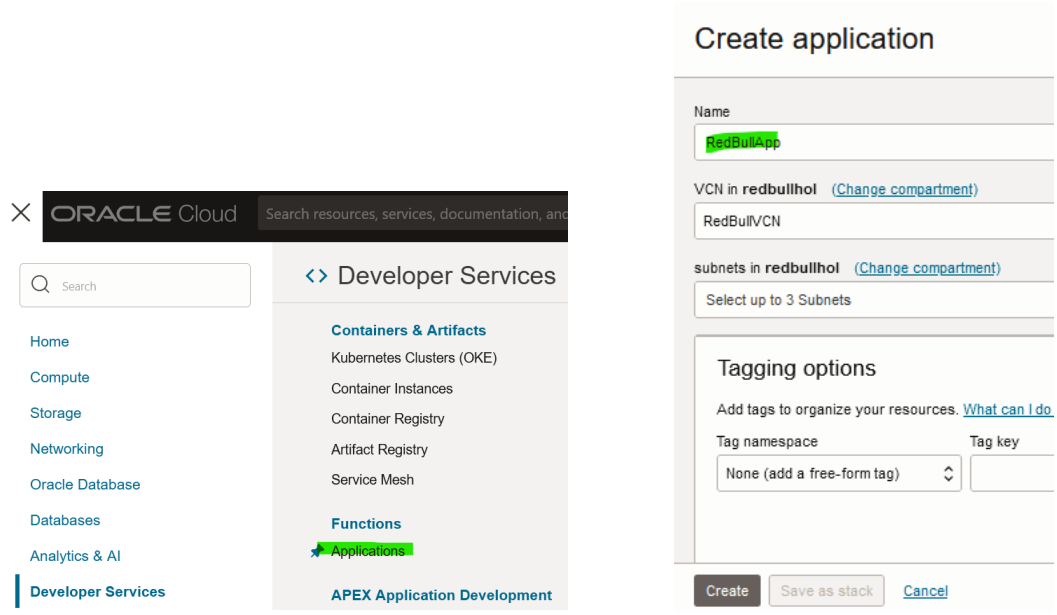
CREACION APLICACIÓN SERVERLESS & SETUP CLOUD SHELL ENVIROMENT

1. Creación o validación de existencias de la capa de red, debe existir una VCN y una subred, si ya tienes creada una VCN puedes usar la existente.

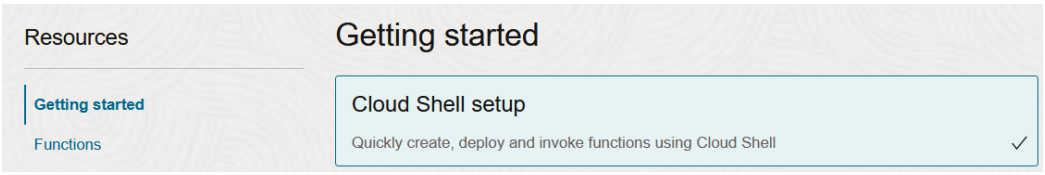




2. Crear de Aplicación Serverless con el nombre **RedBullApp**



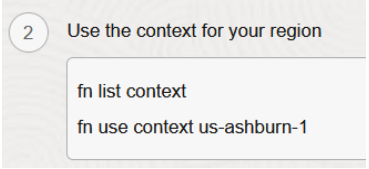
3. Setup del ambiente Cloud Shell para esto se deben seguir las instrucciones dadas en la consola en el siguiente apartado:



Aquí un ejemplo de los comandos y el resultado de cada una de las ejecuciones en Cloud Shell:

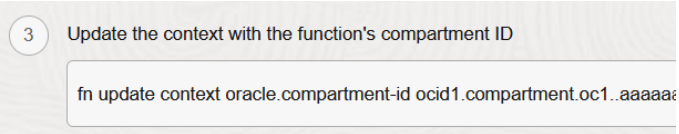
Dara clic en el botón “Launch Cloud Shell”

Listar los diferentes contextos serverless correspondientes a cada una de las regiones que se estén usando, ***copiar y ejecutar los comandos que aparecen en la consola*:**



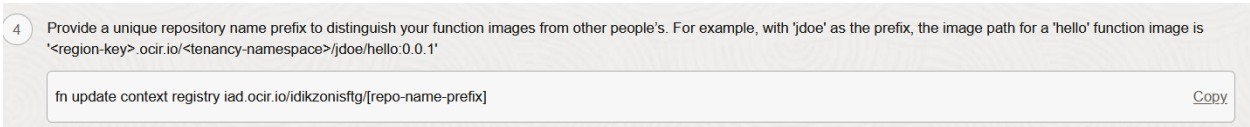
```
Your Cloud Shell machine comes with 5GB of storage for your home directory. Your Cloud Shell (machine and home directory) are located in: US East (Ashburn).
You are using Cloud Shell in tenancy cxmteammcrn as an OCI Local user oscar.bernal@oracle.com

Type `help` for more info.
oscar_bern@cloudshell:~ (us-ashburn-1)$ fn list context
CURRENT NAME           PROVIDER       API URL                                          REGISTRY
        default        oracle-cs
*       us-ashburn-1   oracle-cs      https://functions.us-ashburn-1.oci.oraclecloud.com      iad.ocir.io/idikzonisftg/consumption
        us-phoenix-1   oracle-cs      https://functions.us-phoenix-1.oci.oraclecloud.com      phx.ocir.io/idikzonisftg/javierBM
oscar_bern@cloudshell:~ (us-ashburn-1)$ fn use context us-ashburn-1

Fn: Context us-ashburn-1 currently in use
```

Actualizar el contexto para ser usado




```
oscar_bern@cloudshell:~ (us-ashburn-1)$ fn update context oracle.compartment-id ocid1.compart
Current context updated oracle.compartment-id with ocid1.compartment.oc1..aaaaaaaas76tcr4yeb6
```

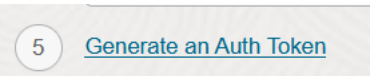
Establecer un **pre-fijo** para el contexto del repositorio de imágenes Docker



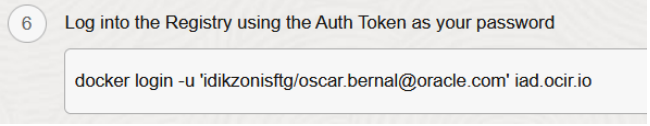
Establecer el repositorio de imágenes, para este caso **debes remplazar [repo-name-prefix] por redbull,** en el comando dado por la consola quedado así:

```
oscar_bern@cloudshell:~ (us-ashburn-1)$ fn update context registry iad.ocir.io/idikzonisftg/redbull
Current context updated registry with iad.ocir.io/idikzonisftg/redbull
oscar_bern@cloudshell:~ (us-ashburn-1)$
```

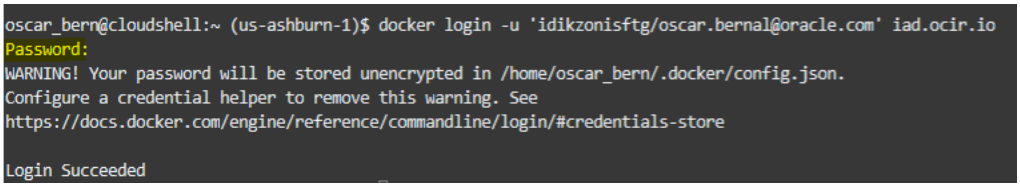
Como en los pre-requisitos ya tenemos generado el token de autorización el paso 5 lo **omitiremos**



Establecer conexión al repositorio de imágenes



Debemos ingresar el **token** creado en pasos previos cuando el sistema nos solicite el **password**

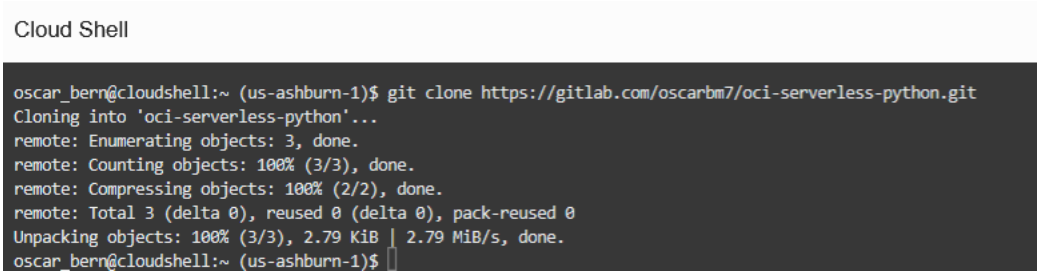


Para este ejercicio **omitiremos los pasos posteriores** ya que el código de la aplicación ya fue construido y se encuentra en un repositorio de Gitlab, solamente tendremos que importarlo y compilarlo en nuestro ambiente.

CONSTRUCCIÓN DE APLICACIÓN

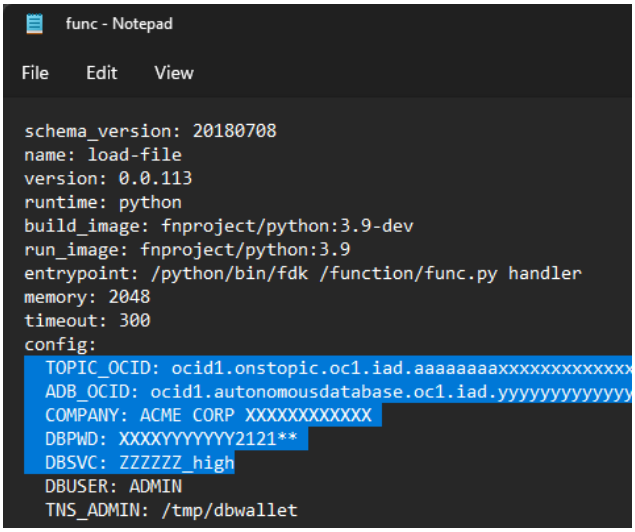
1. En la sesión de Cloud Shell vamos importar el código de la aplicación, con el siguiente comando:

   git clone https://gitlab.com/oscarbm7/oci-serverless-python.git

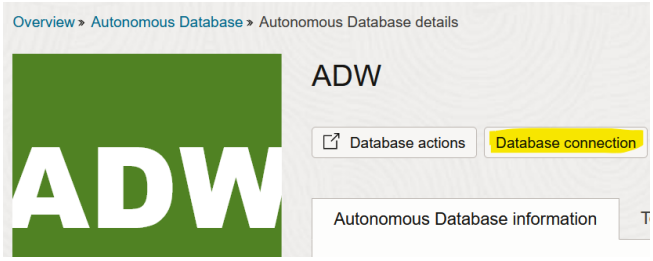


2. Descargar el archivo **func.yaml** el cual tiene los parámetros de configuración de la aplicación, ingresando al siguiente link:
   https://gitlab.com/oscarbm7/oci-serverless-python/-/raw/main/func.yaml?inline=false

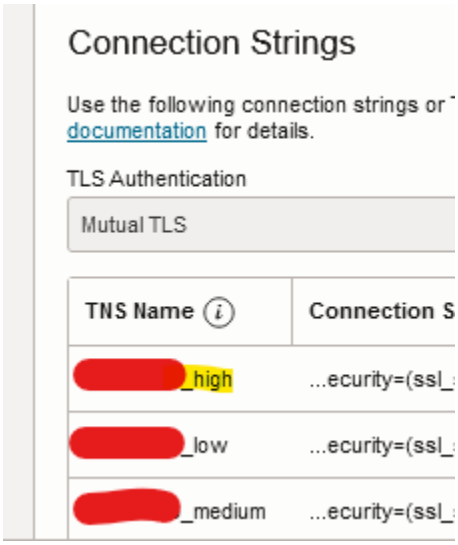
3. Editar el archivo (En Notepad) remplazando los valores resaltados por datos recopilados previamente:



TOPIC_OCID: Es el ***topico de notificaciones*** que creamos en pasos anteriores, al igual que
ADB_OCID: OCID de la base de datos autonoma (creada en sesiones anteriores).
DBSVC: Debes ingresar el nombre de la conexión de la base de datos, la puedes localizar así:



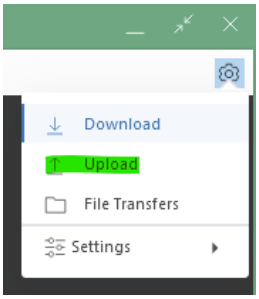
Tomar cualquiera de las disponibles, por ejemplo: **xxxxxx_high** (este valor corresponde según tu base de datos y lo debes poner en el archivo func.yaml)

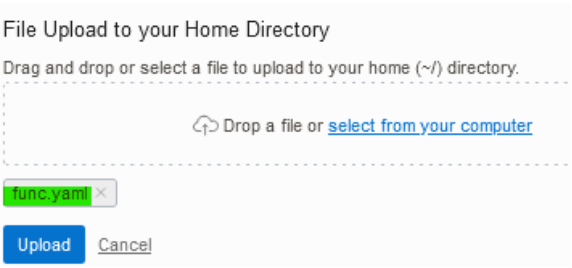


4. El archivo actualizado debe quedar así:

```
schema_version: 20180708
name: load-file
version: 0.0.114
runtime: python
build_image: fnproject/python:3.9-dev
run_image: fnproject/python:3.9
entrypoint: /python/bin/fdk /function/func.py handler
memory: 2048
timeout: 300
config:
  ADB_OCID: ocid1.autonomousdatabase.oc1.iad.anuwcljtubxct
  COMPANY: ACME CORP XXX
  DBPWD: [redacted]
  DBSVC: [redacted]_high
  DBUSER: ADMIN
  TNS_ADMIN: /tmp/dbwallet
  TOPIC_OCID: ocid1.onstopic.oc1.iad.aaaaaaaagruoyyzvxufcz
```

5. Subir el archivo editado al Cloud Shell:





Este subirá al directorio home del Cloud Shell

6. Remplazar el archivo de la función con el subido previamente:

mv func.yaml oci-serverless-python/

7. Ingresamos a la carpeta importada, con el siguiente comando: ***cd oci-serverless-python***

```
oscar_bern@cloudshell:~ (us-ashburn-1)$ cd oci-serverless-python/
```

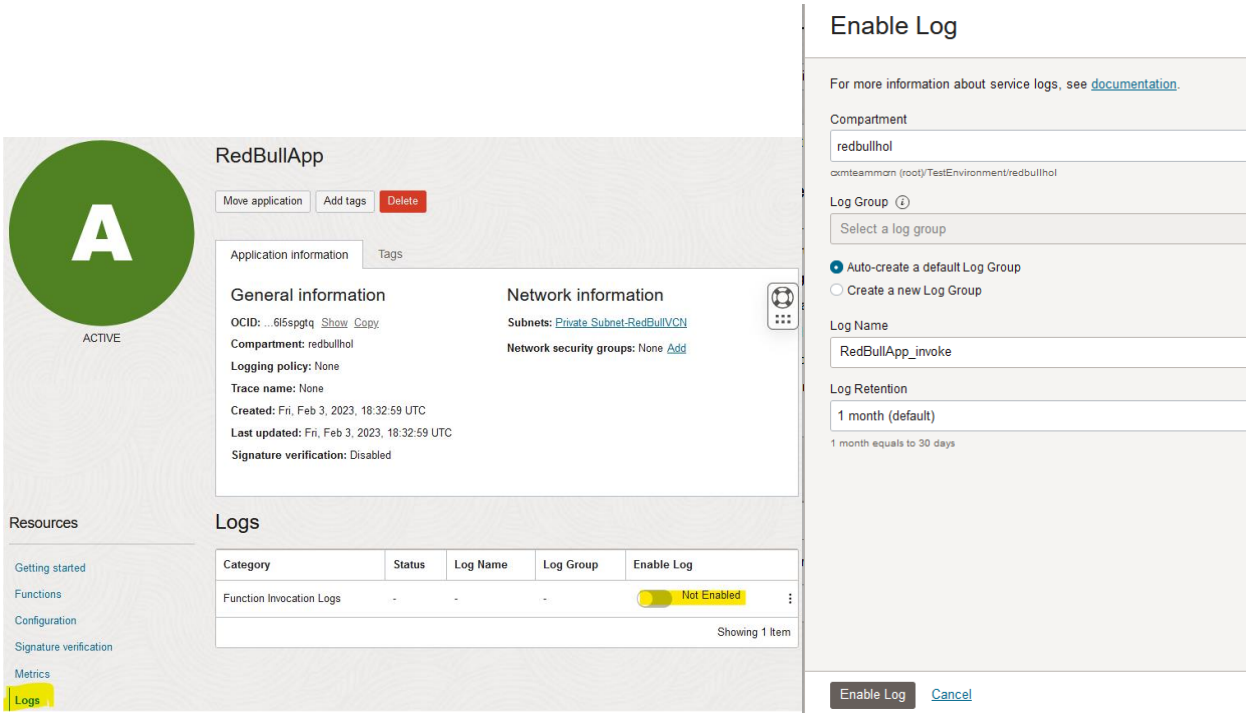
8. Compilar la aplicación serverless, con el comando: ***fn -v deploy --app RedBullApp***

```
oscar_bern@cloudshell:oci-serverless-python (us-ashburn-1)$ fn -v deploy --app RedBullApp
Deploying load-file to app: RedBullApp
Bumped to version 0.0.97
Using Container engine docker
Building image iad.ocir.io/idikzonisftg/redbull/load-file:0.0.97
Dockerfile content
----------------------------------
FROM fnproject/python:3.9-dev as build-stage
WORKDIR /function
ADD requirements.txt /function/

                    RUN pip3 install --target /python/  --no-cache --no-cache-dir -r requirements.txt &&\
                        rm -fr ~/.cache/pip /tmp* requirements.txt func.yaml Dockerfile .venv &&\
                        chmod -R o+r /python
```

Si todo esta correcto el resultado en el CloudShell debe ser:

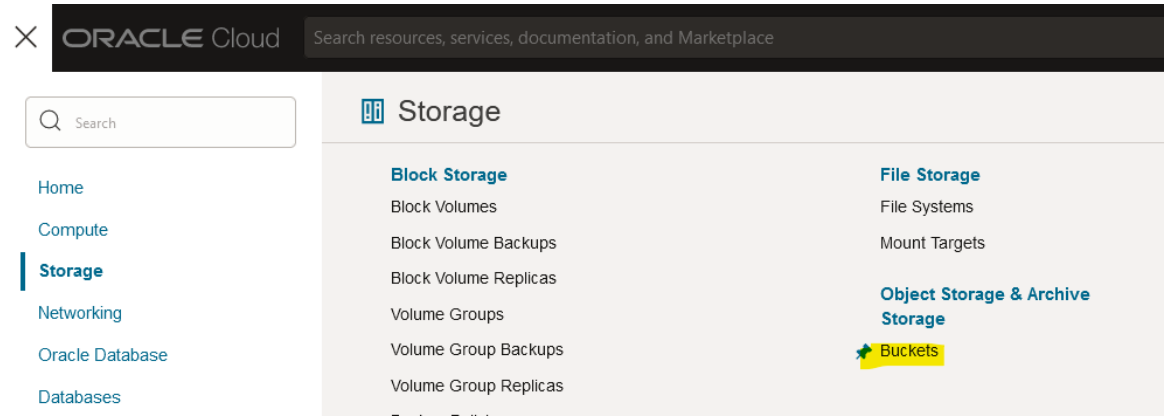
Successfully created function

9. Habilitar LOGS para rastreo de errores e información relevante

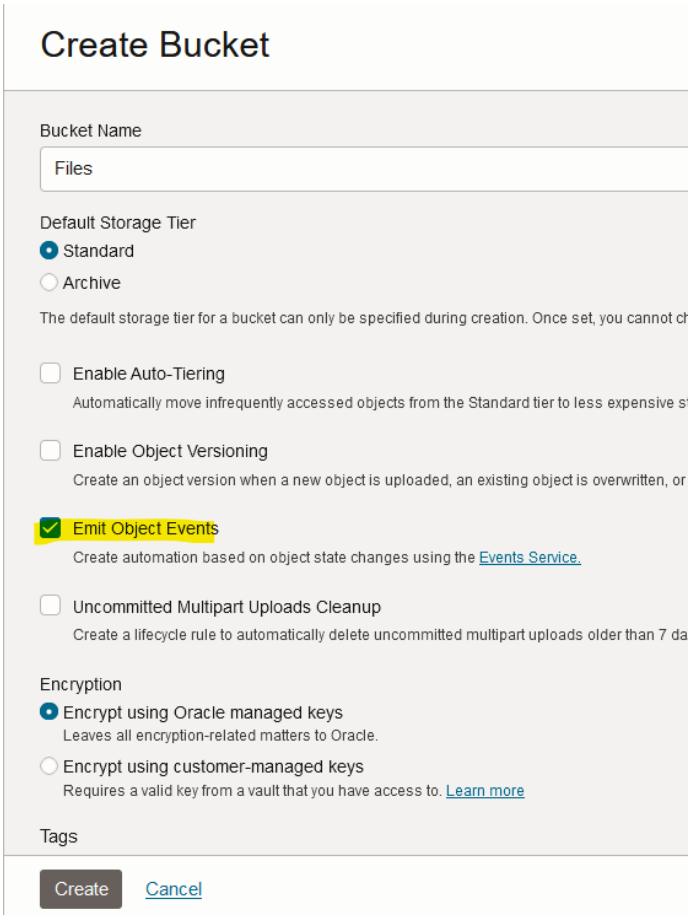


CREACION BUCKET PARA ARCHIVOS

Crear Bucket en el servicio de Object storage

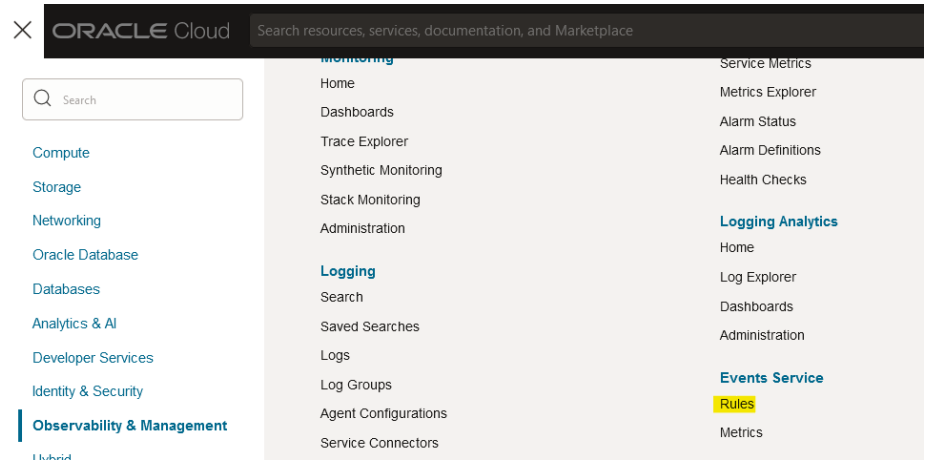


IMPORTANTE: habilitar la opción de emisión de eventos, ya que esto es lo que ejecutara la función serverless para la carga del respectivo archivo.

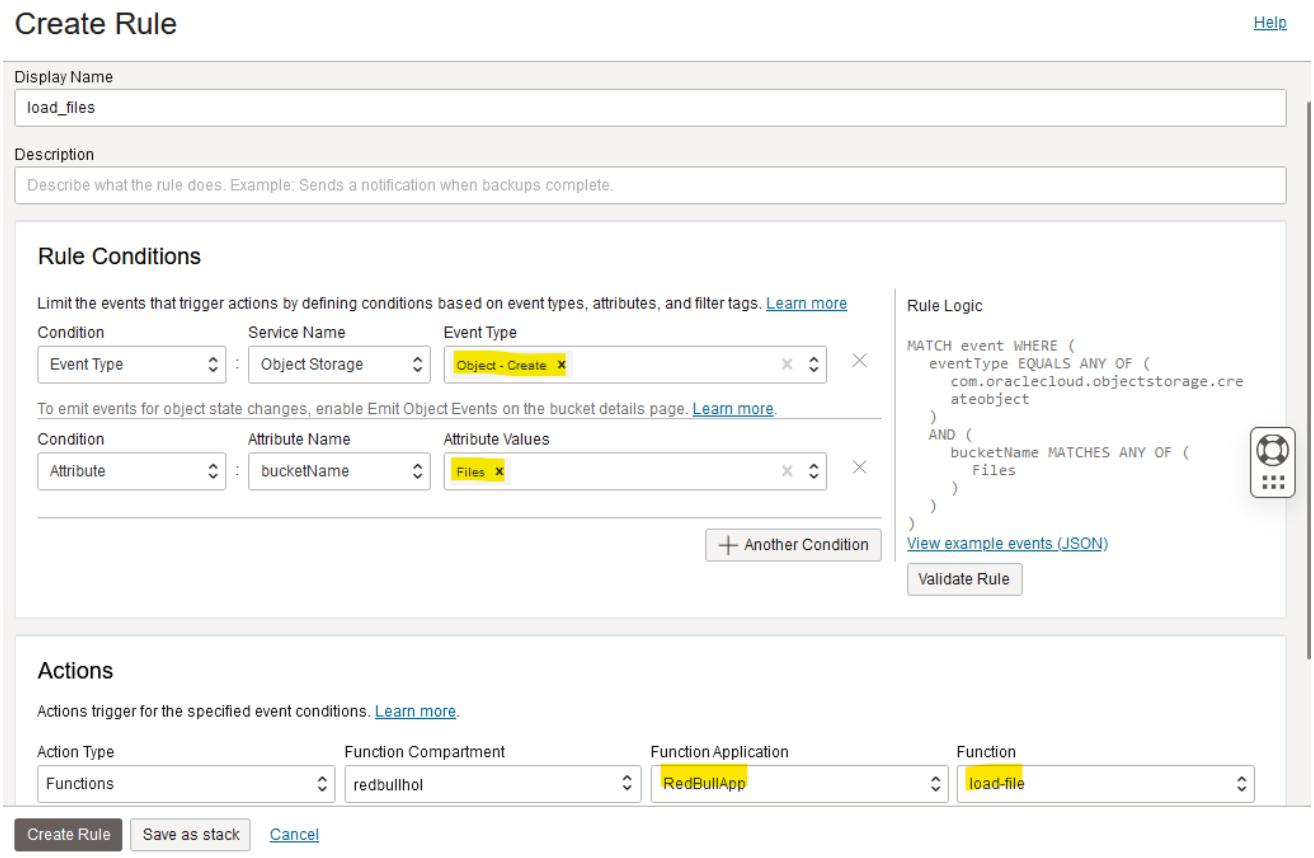


CONFIGURACION SERVICE CONNECTOR HUB

En el módulo de Observability



Crear la regla que ejecutara la función cada vez que se cargue el archivo



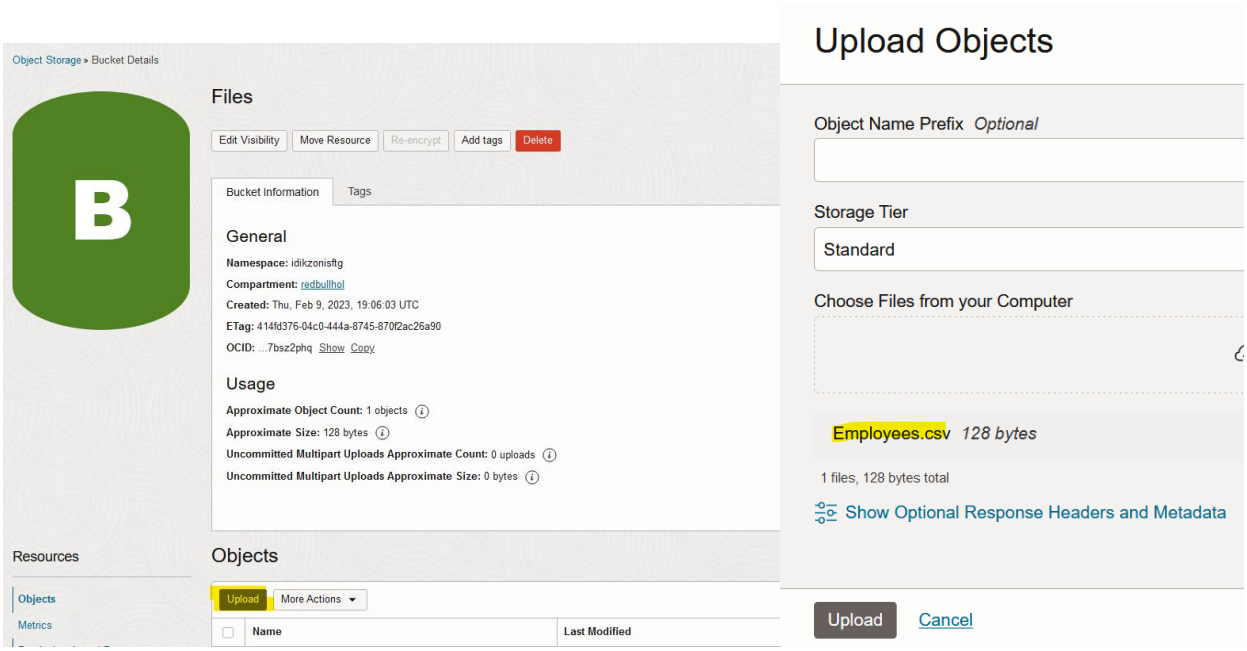
PROBAR APLICACIÓN

Finalmente puedes probar tu aplicación únicamente cargando el archivo en el bucket creado previamente del servicio object storage.

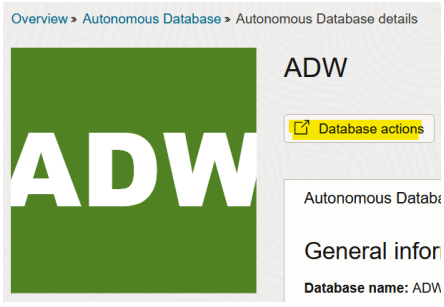
1. Descargar archivo de prueba CSV:

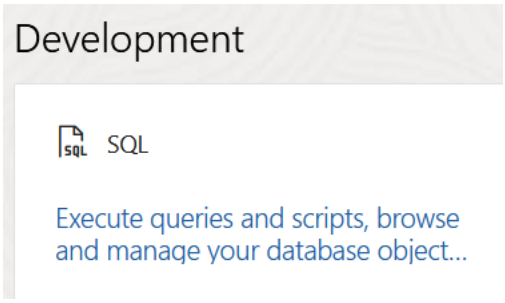
https://objectstorage.us-ashburn-1.oraclecloud.com/p/MSmBkReA-TET1pfUpsvX5ZsC6uTFZpU140p7t7uitURUJ9hwOKOI0z0O5mn7stXJ/n/idikzonisftg/b/DataFile/o/Employees.csv

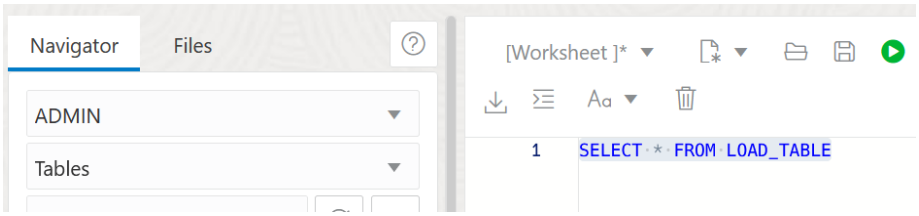
2. Cargar el archivo en el bucket creado



Después de cargado el archivo será procesado por la función Serverless y cargado en la base de datos en la tabla **LOAD_TABLE**

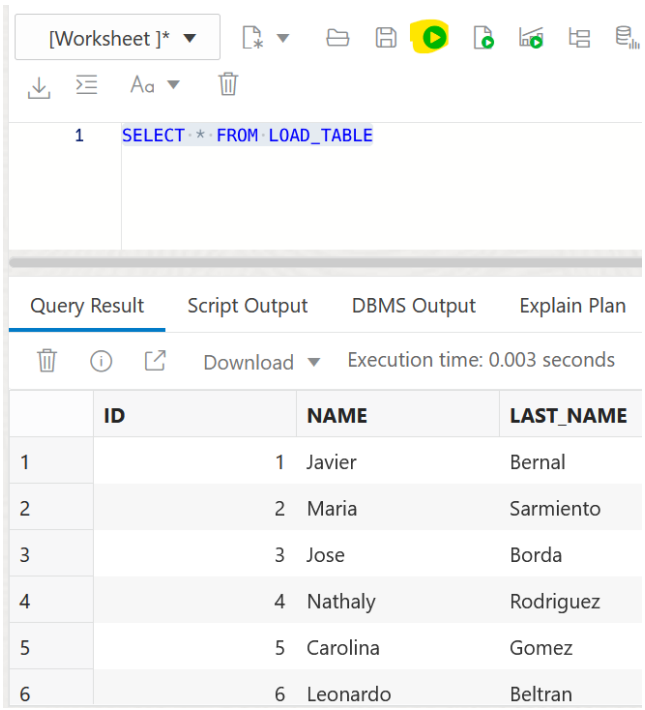






En el editor SQL ingresar la siguiente instrucción:

SELECT * FROM LOAD_TABLE



	ID	NAME	LAST_NAME
1	1	Javier	Bernal
2	2	Maria	Sarmiento
3	3	Jose	Borda
4	4	Nathaly	Rodriguez
5	5	Carolina	Gomez
6	6	Leonardo	Beltran

Bibliografía:

https://docs.oracle.com/en-us/iaas/Content/Functions/Concepts/functionsoverview.htm

https://oracle.github.io/python-oracledb/

https://fnproject.io/

https://medium.com/oracledevs/an-exploration-using-oci-functions-4c5d4e70d00c

https://oracle-cloud-infrastructure-python-sdk.readthedocs.io/en/latest/api