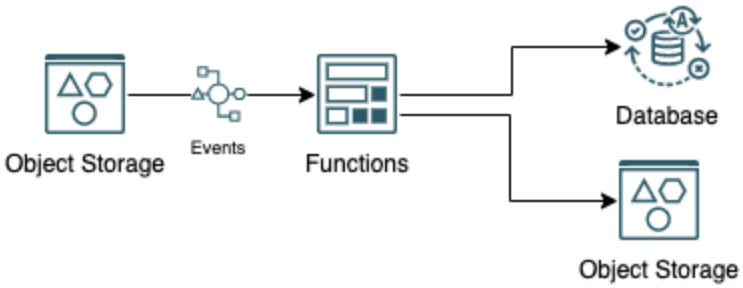
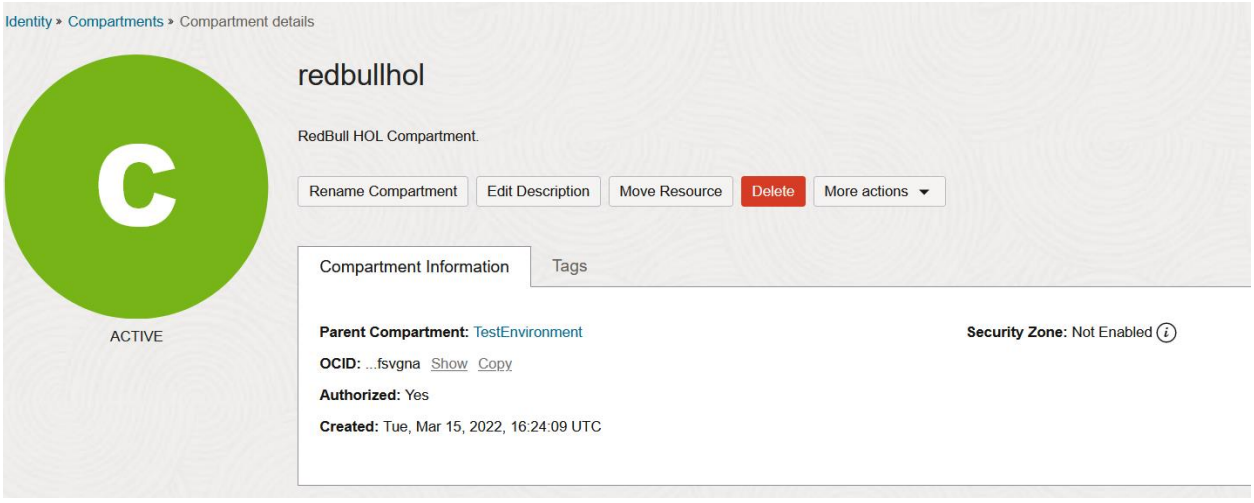
LABORATORIO CONSTRUCCIÓN DE APLICACIONES ORACLE CLOUD SERVERLESS

Este laboratorio esta enfocado a que puedas construir una aplicación Serverless en tu cuenta de Oracle cloud, el objetivo es construir una aplicación que realice la lectura, procesamiento y guardado de un archivo separado por comas(*.csv) en una base de datos autonomos creada previamente, chicha apliaccion tendra la siguiente aquitectura:



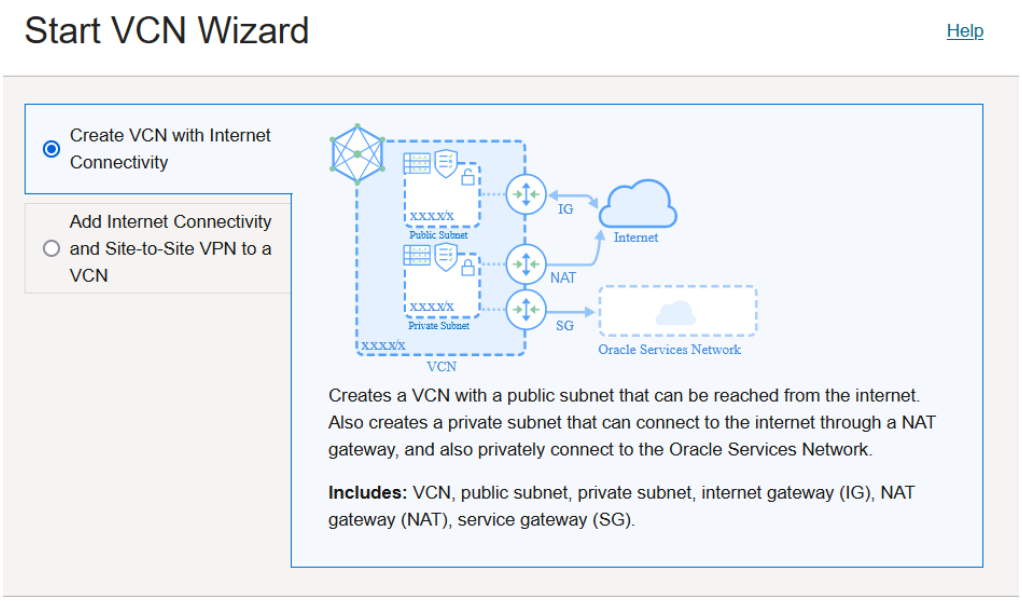
PRE-REQUISITOS

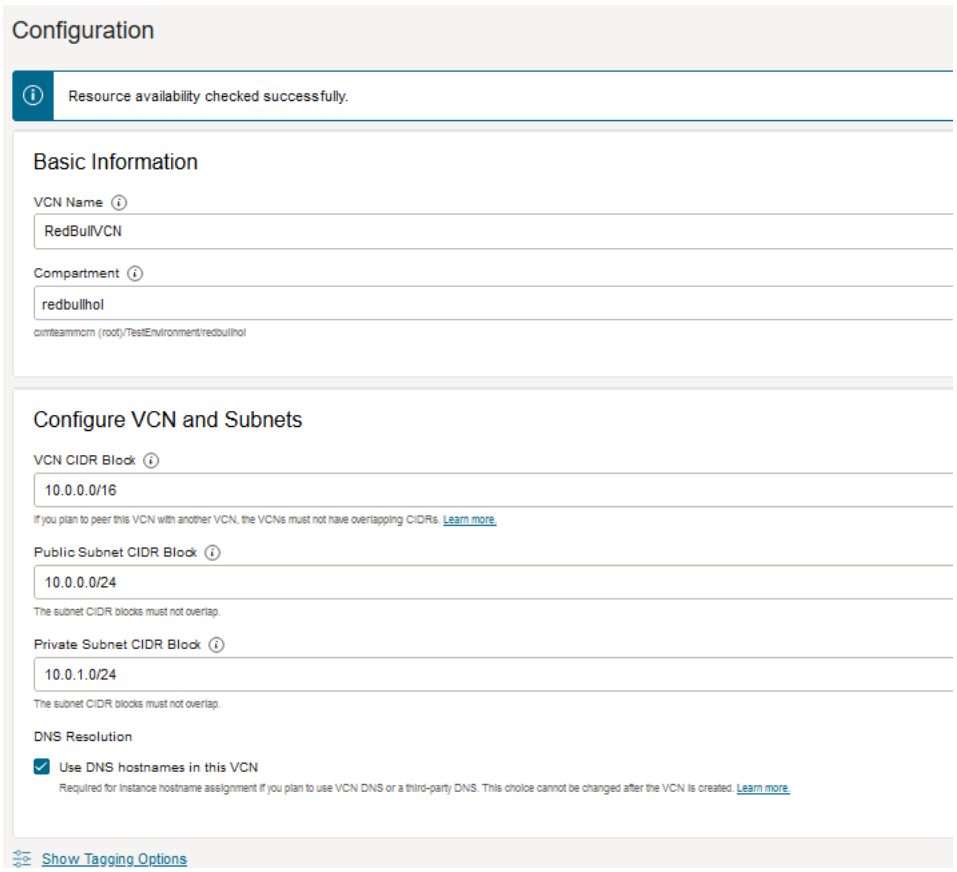
1. Usar el compartment existente para agrupar los recursos del laboratorio



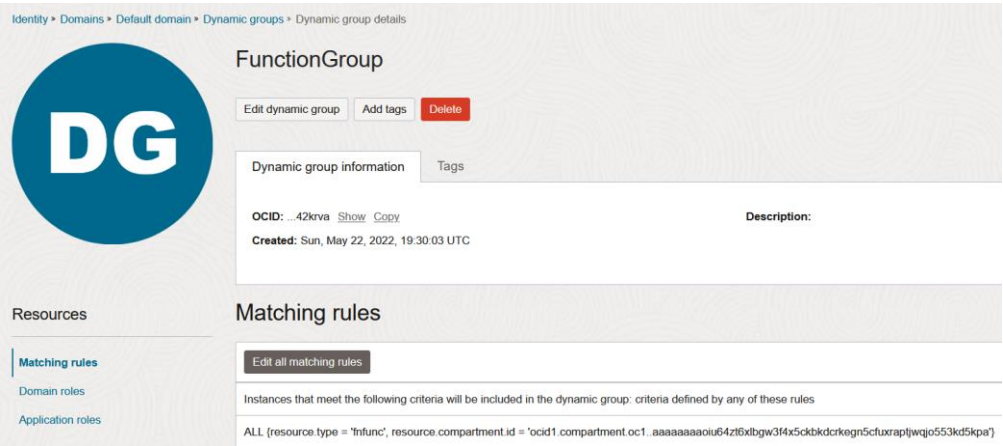
2. Crear o usar una base de datos autónoma existente

3. 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.





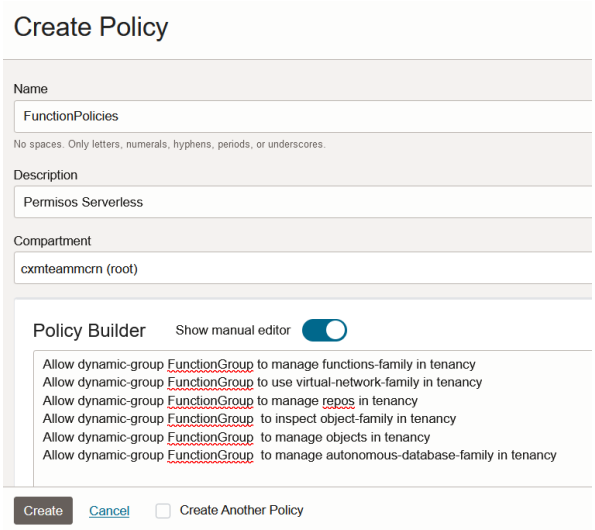
4. Creación de grupo dinámico y políticas de seguridad para manipulación de la infraestructura OCI por parte de la función serverless.



Estableciendo la siguiente regla para el grupo dinámico con el respectivo OCID del compartment donde se encuentre localizada la función serverless

ALL {resource.type = 'fnfunc', resource.compartment.id = 'ocid1.compartment.xxxxxxxxxxxxxxxxxxxxxxxxxxxxx }

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

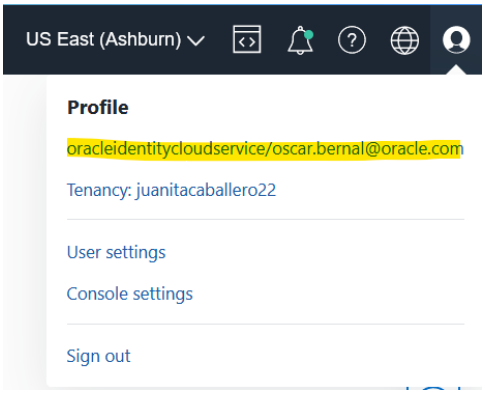


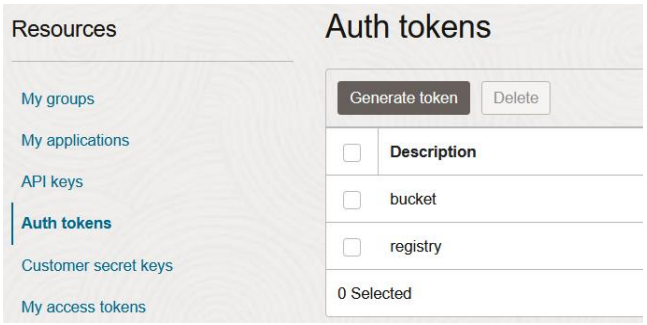
Allow dynamic-group FunctionGroup to manage functions-family in tenancy
Allow dynamic-group FunctionGroup to use virtual-network-family in tenancy
Allow dynamic-group FunctionGroup to manage repos in tenancy

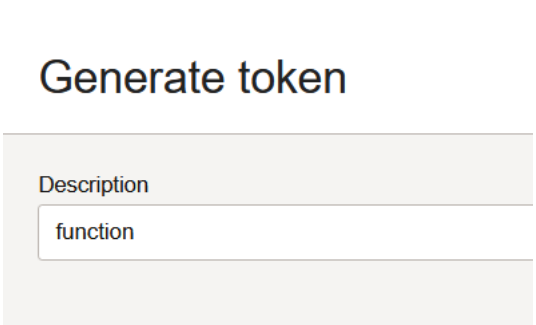
Allow dynamic-group FunctionGroup to inspect object-family in tenancy
Allow dynamic-group FunctionGroup to manage objects in tenancy
Allow dynamic-group FunctionGroup to manage autonomous-database-family in tenancy
Allow dynamic-group FunctionGroup to use ons-topics in tenancy

GENERAR AUTH TOKEN

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

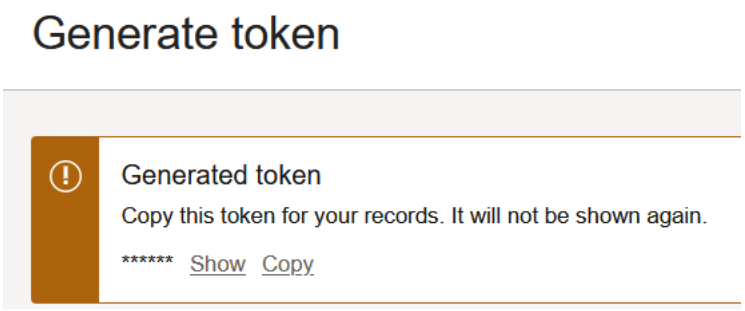






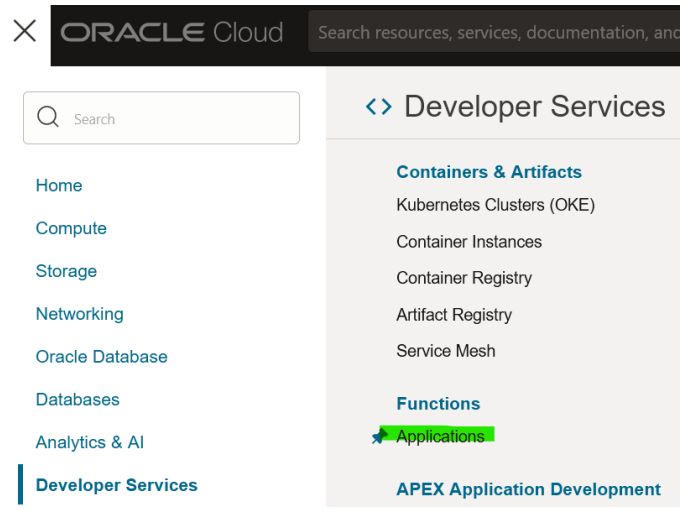
Guardar el valor generado por la consola el cual debemos usar en los pasos posteriores

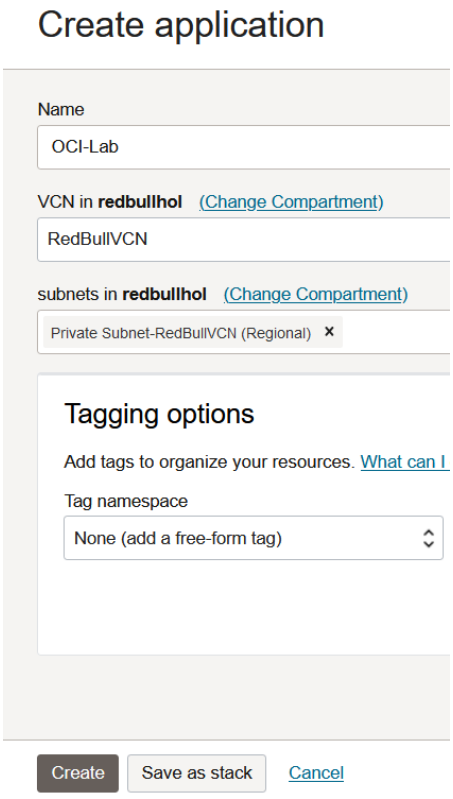
Ejemplo: v#1iD<8Ycx+)Z+XUR5av



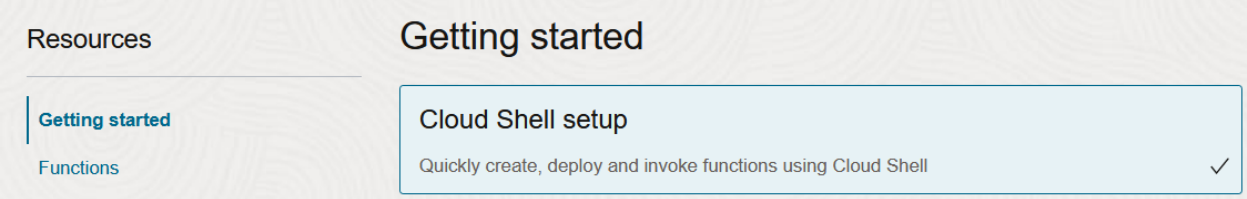
CREACION APLICACIÓN SERVERLESS & SETUP CLOUD SHELL ENVIROMENT

1. Creación de Aplicación Serverless

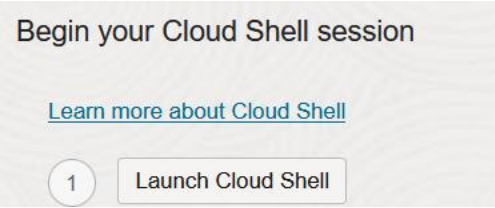




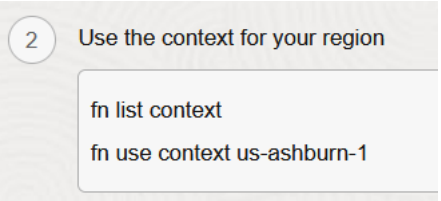
2. Setup del ambiente cloud Shell para esto se deben seguir las instrucciones dadas en la consola en el siguiente apartado:



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

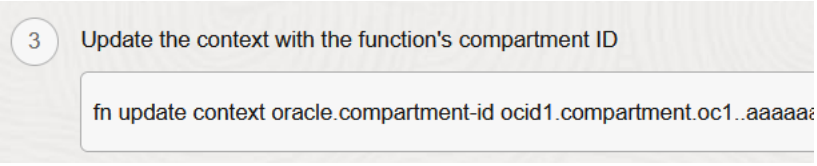


Listar los diferentes contextos serverless correspondientes a cada una de las regiones que se esten usando



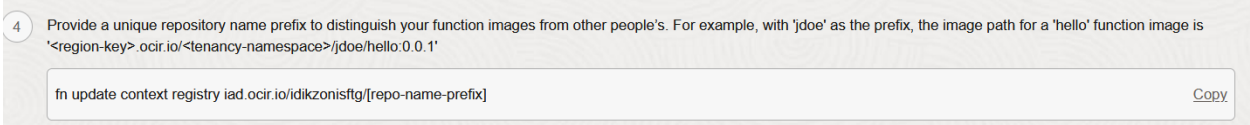

```
Your Cloud Shell machine comes with 5GB of storage for your home directory. Your Cloud Shell (machine and home directory) are located in: US East (Ashburn).
You are using Cloud Shell in tenancy cxmteammcrn as an OCI Local user oscar.bernal@oracle.com

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
```

Setear 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
```

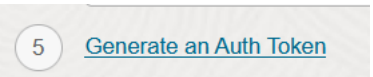
Establecer un pre-fijo para el contexto del repositorio de imágenes Docker



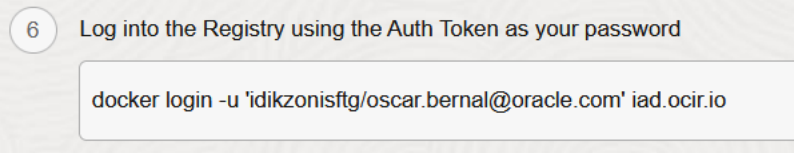
Para este caso ingresamos el prefijo **redbull**

```
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

```
oscar_bern@cloudshell:~ (us-ashburn-1)$ docker login -u 'idikzonisftg/oscar.bernal@oracle.com' iad.ocir.io
Password:
WARNING! Your password will be stored unencrypted in /home/oscar_bern/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Para este ejercicio omitiremos los pasos posteriores ya que en el siguiente repositorio de Gitlab tendremos disponible el código de la aplicación y únicamente tendremos que importarlo y compilarlo en nuestra cuenta cloud.

CONSTRUCCIÓN DE APLICACIÓN

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

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

Cloud Shell

```
oscar_bern@cloudshell:~ (us-ashburn-1)$ git clone https://gitlab.com/oscarbm7/oci-serverless-python.git
Cloning into 'oci-serverless-python'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 2.79 KiB | 2.79 MiB/s, done.
oscar_bern@cloudshell:~ (us-ashburn-1)$
```

2. Ingresamos a la carpeta importada

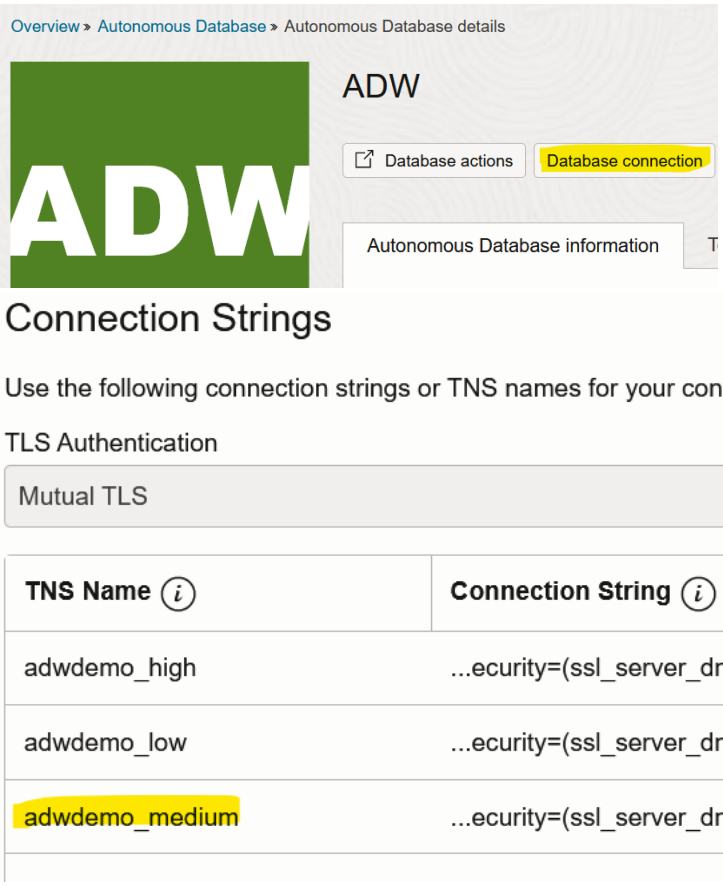
```
oscar_bern@cloudshell:~ (us-ashburn-1)$ cd oci-serverless-python/
```

3. Ajustar el archivo de configuración de aplicación de acuerdo a mi ambiente (conexiones, base de datos, passwords), para la cual editaremos el archivo func.yaml con el editor de Linux Nano:

```
oscar_bern@cloudshell:oci-serverless-python (us-ashburn-1)$ nano func.yaml
```

```
  GNU nano 2.3.1                                              File: func.yaml

schema_version: 20180708
name: load-file
version: 0.0.106
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.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  DBPWD: YYxxxxxx123***
  DBSVC: XXXYYYY_high
  DBUSER: ADMIN
  TNS_ADMIN: /tmp/dbwallet
```

Para el parámetro DBSVC se deben remitir a las conexiones de la base de datos autónomos y tomar cualquiera de las disponibles, por ejemplo:



Despues de editar el archivo basta con guardar los cambios con la combinación de teclas Ctrl +X

4. Validar que el archivo fue actualizado con el comando: cat func.yaml

```
oscar_bern@cloudshell:oci-serverless-python (us-ashburn-1)$ cat func.yaml
schema_version: 20180708
name: load-file
version: 0.0.106
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.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  DBPWD: YYxxxxxx123***
  DBSVC: XXXYYYY_high
  DBUSER: ADMIN
  TNS_ADMIN: /tmp/dbwallet
```

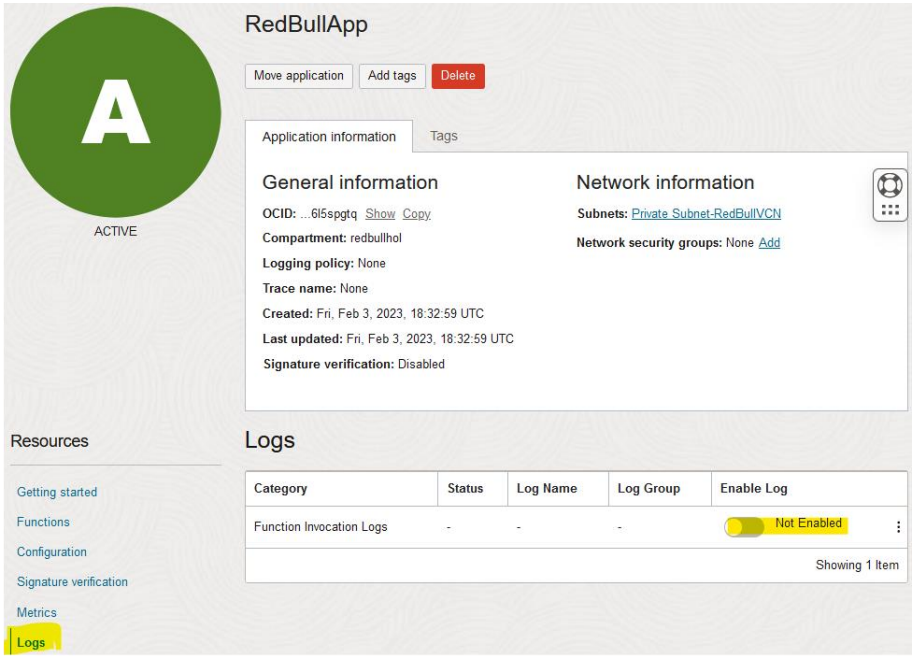
5. Compilar la aplicación serverless: *fn -v deploy --app RedBullApp*

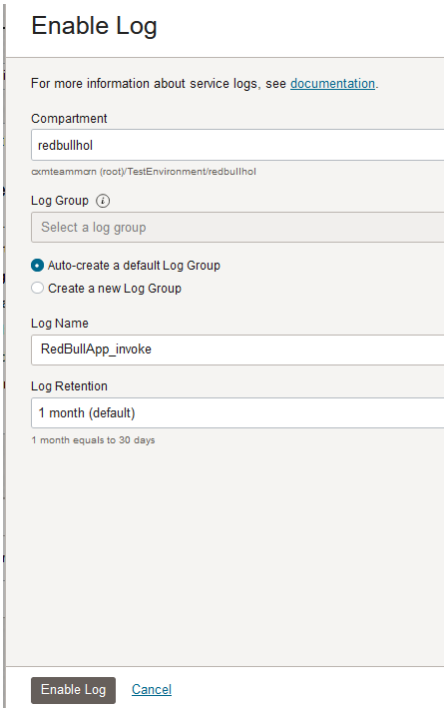
```
oscar_bern@cloudshell:oci-serverless-python (us-ashburn-1)$ fn -v deploy --app RedBullApp
Deploying load-file to app: RedBullApp
Bumped to version 0.0.97
Using Container engine docker
Building image iad.ocir.io/idikzonisftg/redbull/load-file:0.0.97
Dockerfile content
----------------------------------
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

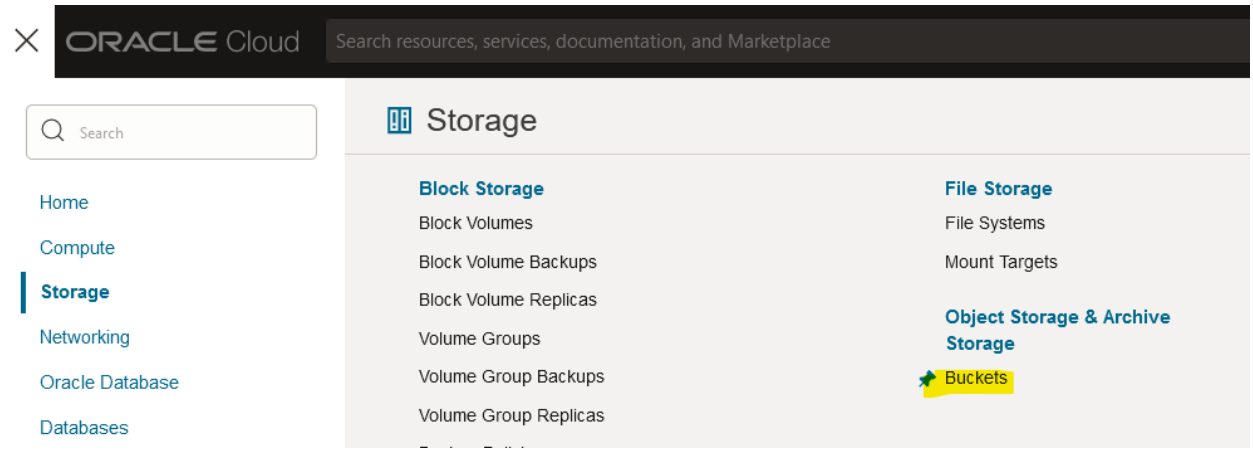
6. Habilitar LOGS para rastreo de errores e información relevante



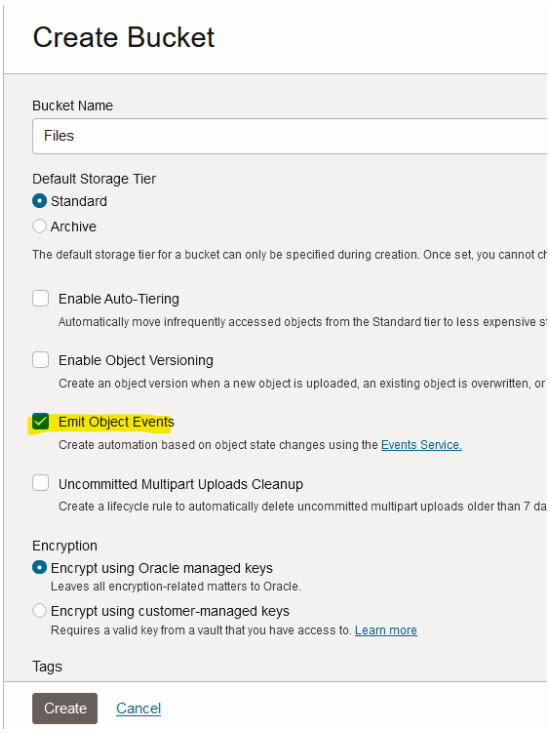

Enable Log
For more information about service logs, see documentation.
Compartment
redbullhol
oxmteammcm (root)/TestEnvironment/redbullhol
Log Group
Select a log group
Auto-create a default Log Group
Create a new Log Group
Log Name
RedBullApp_invoke
Log Retention
1 month (default)
1 month equals to 30 days
Enable Log
Cancel

CREACION BUCKET PARA ARCHIVOS

1. Crear Bucket en el servicio de Object storage

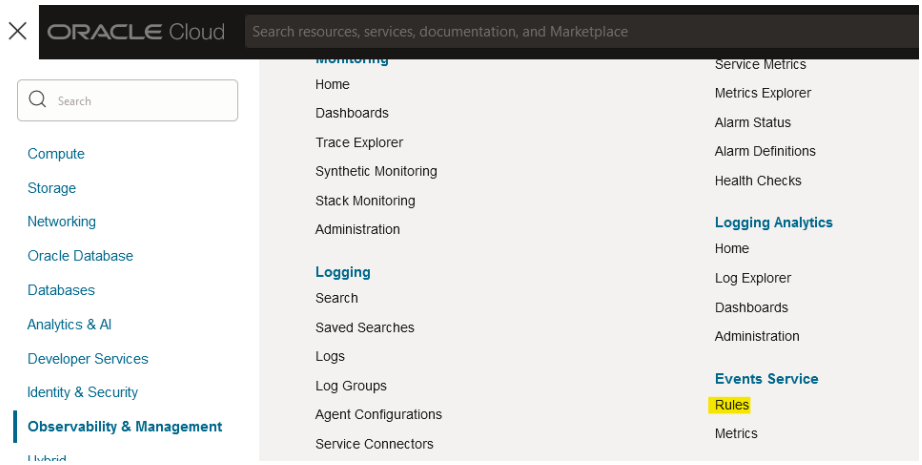


IMPORTANTE: habilitar la opción de emisión de eventos, ya que esto es lo que ejecutara la función serverless para la carga del respectivo archivo.

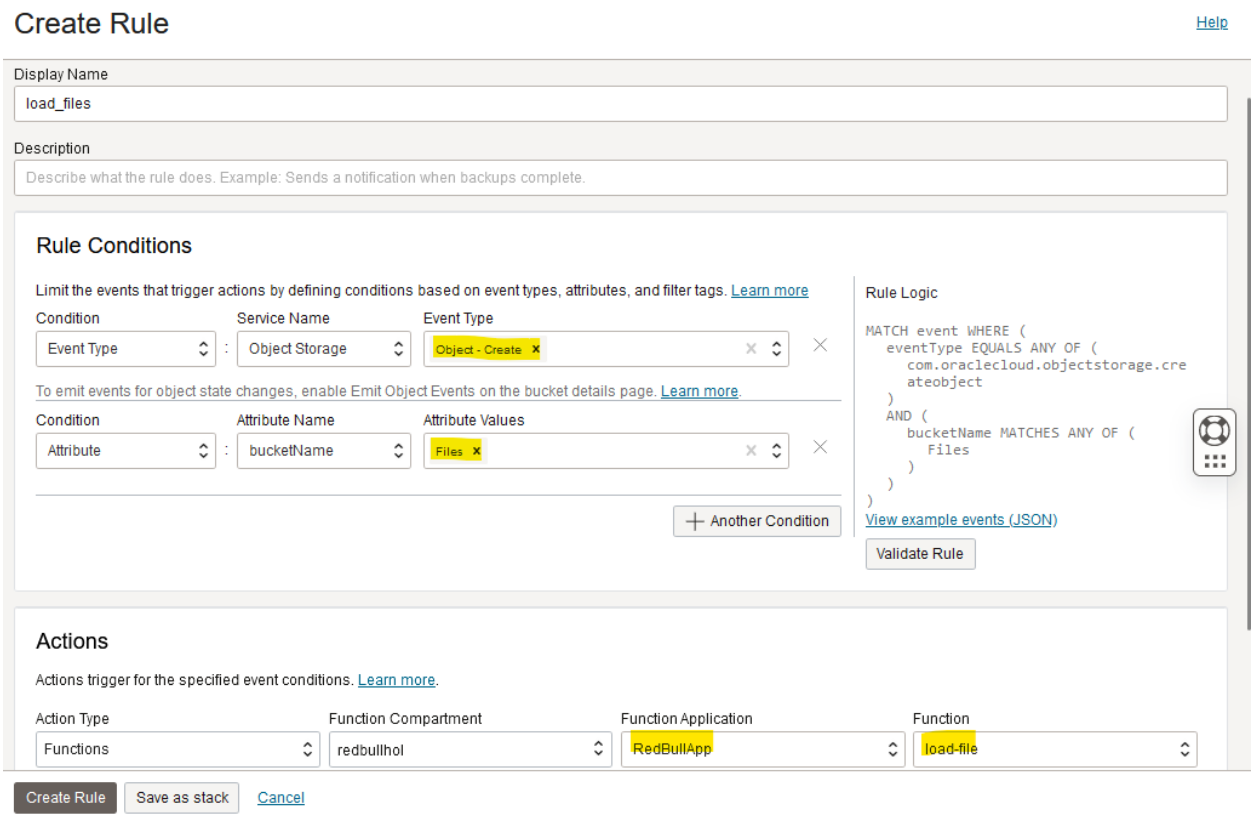


CONFIGURACION SERVICE CONNECTOR HUB

En el módulo de Observability



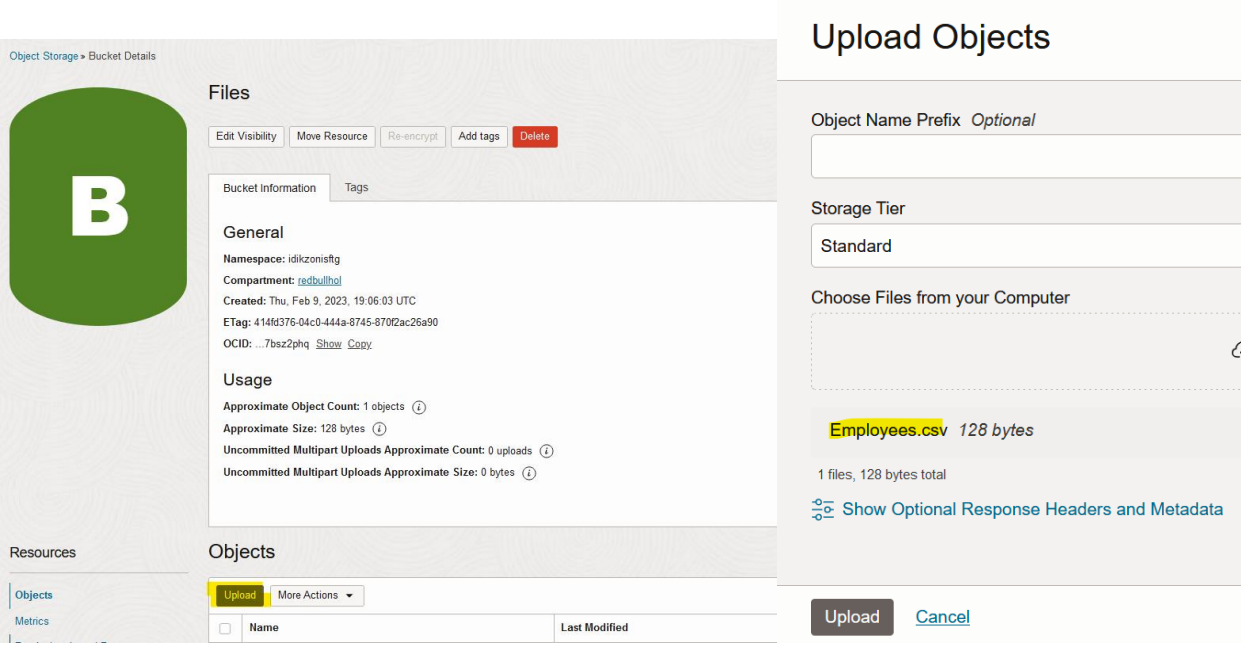
Crear la regla que ejecutara la función cada vez que se cargue el archivo



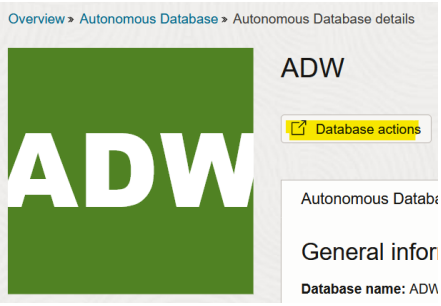
PROBAR APLICACIÓN

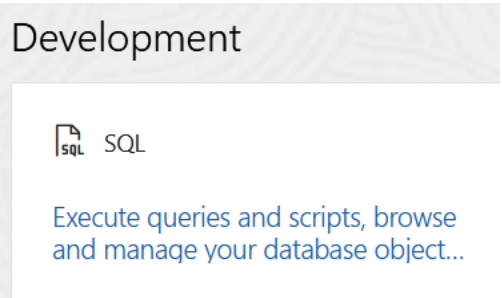
Finalmente puedes probar tu aplicación únicamente cargando el archivo en el bucket del object storage:

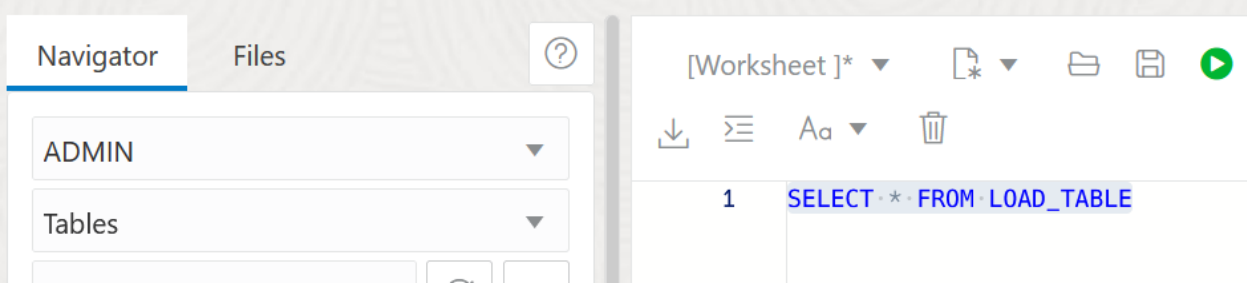
https://objectstorage.us-ashburn-1.oraclecloud.com/p/MSmBkReA-TET1pfUpsvX5ZsC6uTFZpU140p7t7uitURUJ9hwOKOI0z0O5mn7stXJ/n/idikzonisftg/b/DataFile/o/Employees.csv

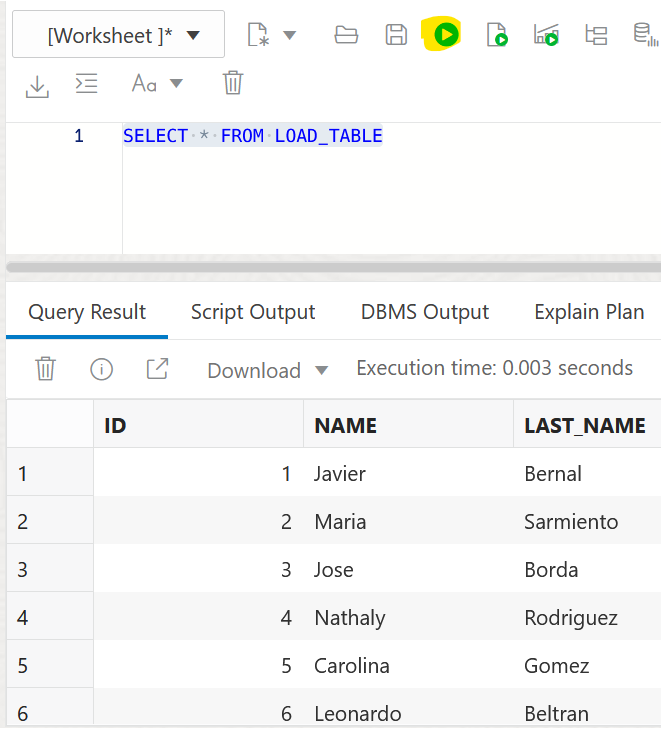


Después de cargado el archivo será procesado por la función Serverless y cargado en la base de datos en la tabla **LOAD_TABLE**









	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