Water Data Project para Groots

Dashboards - Gestión de archivos R

[Alcance 1](#_Toc70951560)

[Requerimientos 1](#_Toc70951561)

[Ciclo de vida 1](#_Toc70951562)

[Desarrollo local 3](#_Toc70951563)

[IMPORTANTE 3](#_Toc70951564)

[Cómo autenticarse al servidor de proagenda2030 3](#_Toc70951565)

[Uso del token localmente 3](#_Toc70951566)

[Ejemplo 3](#_Toc70951567)

[Proceso de actualización desde Github 4](#_Toc70951568)

[Subida de archivos 4](#_Toc70951569)

[Eliminación de archivos 5](#_Toc70951570)

[Despliegue automático en servidor 6](#_Toc70951571)

[Solicitar nuevo recurso a sys admin 7](#_Toc70951572)

[Data Model para sys admins 7](#_Toc70951573)

# Alcance

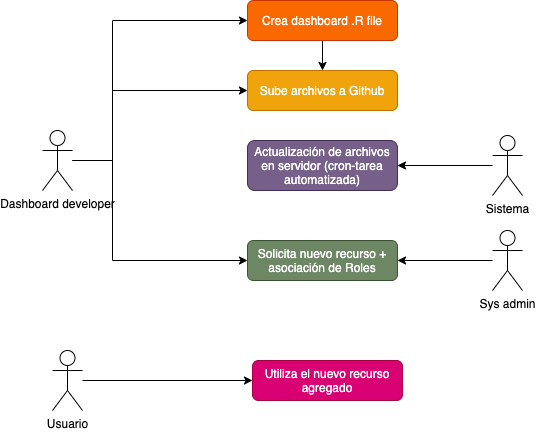
Instrucciones para gestionar los Dashboards del proyecto ProAgenda2030.

Se explica cómo gestionar los archivos directamente desde Github.com

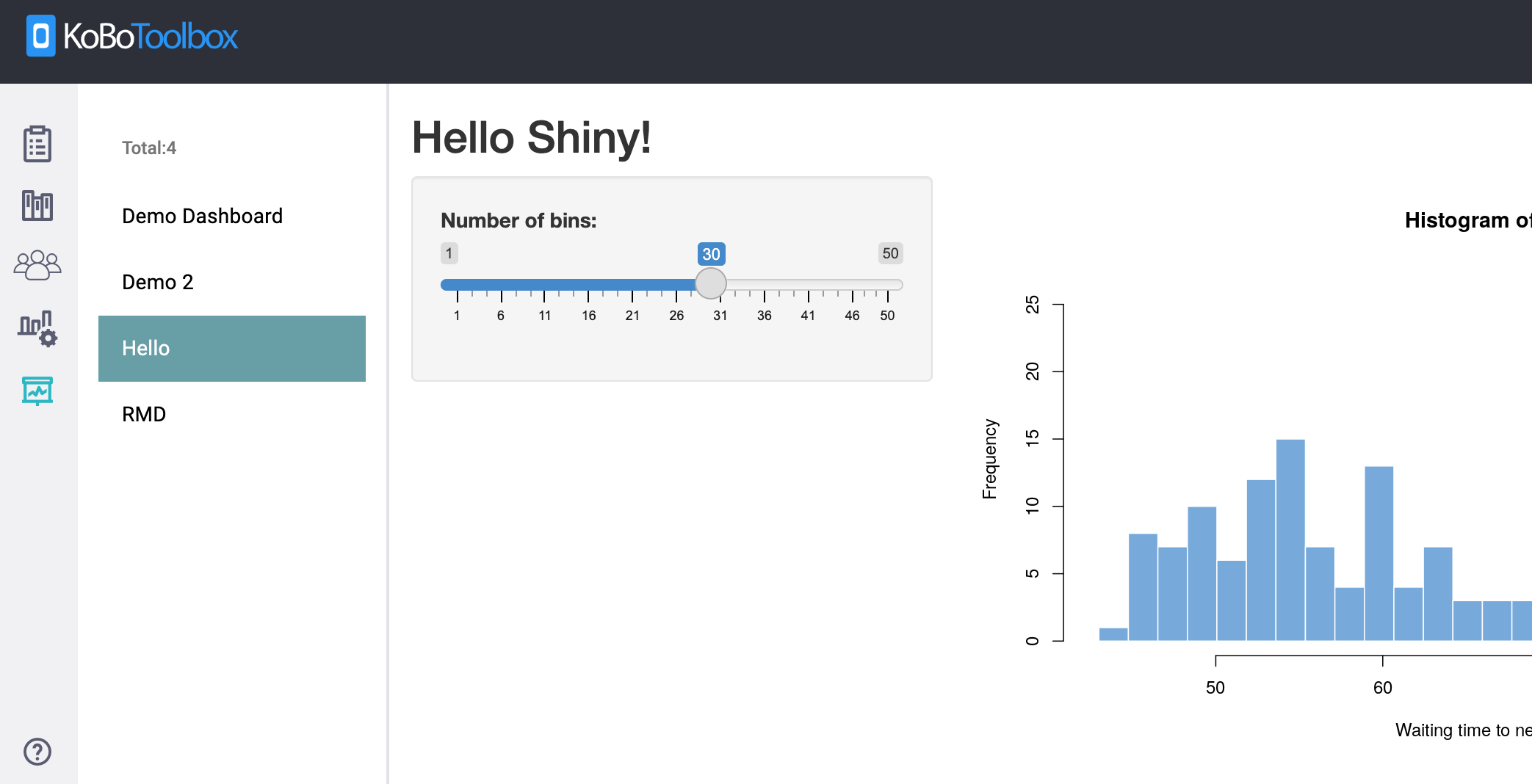
# Requerimientos

* Conocimientos técnicos:
  + Manejo de Github. Básico
  + Manejo de R y proyecto Shiny (RStudio). Medio-Avanzado
* Permisos:
  + Debe solicitar permiso de **Colaborador** al repositorio:
    - <https://github.com/NexionBolivia/dashboards-shiny>

# Ciclo de vida



* Paso 1. El desarrollador de dashboards debe crear y probar localmente su archivo app.R
* Paso 2. Debe actualizar Github para agregar o modificar el archivo
* Paso 3. El proceso automático del sistema realiza una actualización periódica de los archivos de Github dentro del servidor
  + Este proceso es realizado cada 20 minutos o el valor configurado al instalar el sistema
* Paso 4. El desarrollador de dashboards solicita al administrador de sistemas que realice la actualización de Recursos y Asociación del recurso a un rol predeterminado conocido por ambos
* Paso 5. Un usuario con el Rol adecuado puede visualizar el dashboard



# Desarrollo local de dashboards (Paso 1)

Durante el desarrollo o mantenimiento de dashboards será necesario acceder a los datos de KoBoToolbox. El mismo require que las llamadas a su API se realicen a través de un token de autenticación. Este token corresponde a un usuario con los suficientes privilegios para el consumo de estos datos.

**Use el llamado a la variable DASHBOARDS\_KOBO\_TOKEN de la siguiente manera:**

* **Sys.getenv('DASHBOARDS\_KOBO\_TOKEN')**

## IMPORTANTE

No deberá crear sus dashboards con el usuario y contraseña en el archivo, en vez de ello, use el token mencionado en el punto anterior: **DASHBOARDS\_KOBO\_TOKEN**

Es de especial importancia no publicar usuarios y contraseñas en archivos .R debido a que el código fuente del proyecto puede ser leído desde cualquier parte del mundo.

## Cómo autenticarse al servidor de proagenda2030

Se deberá usar la variable de entorno **DASHBOARDS\_KOBO\_TOKEN**:

**kobo\_token** <- Sys.getenv('DASHBOARDS\_KOBO\_TOKEN')

Esta variable ya se encuentra en el servidor

La llamada a una URL se debe hacer pasando el método “add\_headers”.

raw\_form <- GET(url\_form, add\_headers(Authorization = paste("Token ", **kobo\_token**)), progress())

## Uso del token localmente

Para contar con la variable **DASHBOARDS\_KOBO\_TOKEN** de manera local, deberá contar con el archivo .Renviron en su ambiente local con el valor de la variable de la siguiente manera:

*(Archivo* ***.Renviron****)*

**DASHBOARDS\_KOBO\_TOKEN=my-valor-de-token**

El Token será enviado a los responsables de la información por lo que su uso es de alta responsabilidad.

## Ejemplo

El siguiente es un sample de uso del token. Asume que el token existe dentro del archivo .Renviron y utiliza docker-compose mapeando un volúmen al mismo archivo, de manera que sólo será necesaria la ejecución del comando:

**docker-compose up**

(Se requiere docker desktop instalado en su máquina)

Fuente:

<https://1drv.ms/u/s!AlCLwp4k6sViwlS8VfGeRKIBSR9J?e=1BcLQU>

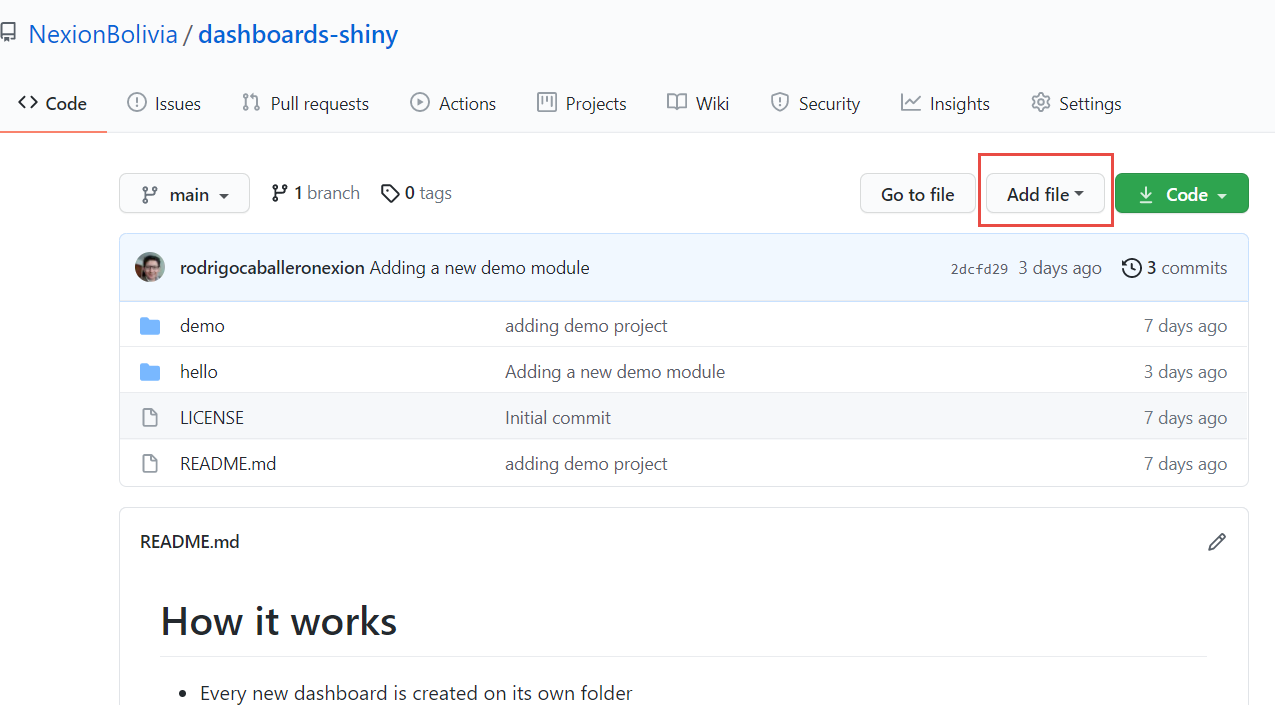
# Proceso de actualización desde Github (Paso 2)

## Subida de archivos

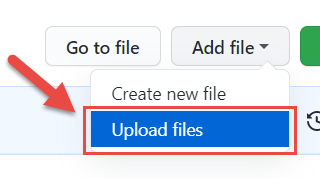
Dirigirse a Github:

* <https://github.com/NexionBolivia/dashboards-shiny>

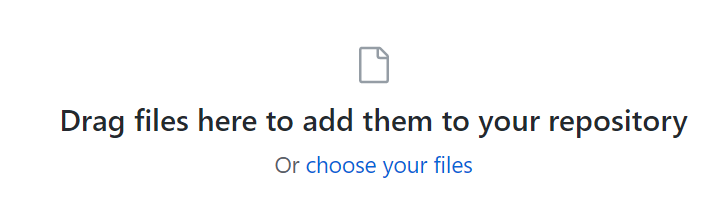
Presionar el botón “Add file”



Seleccionar la opción “Upload files”



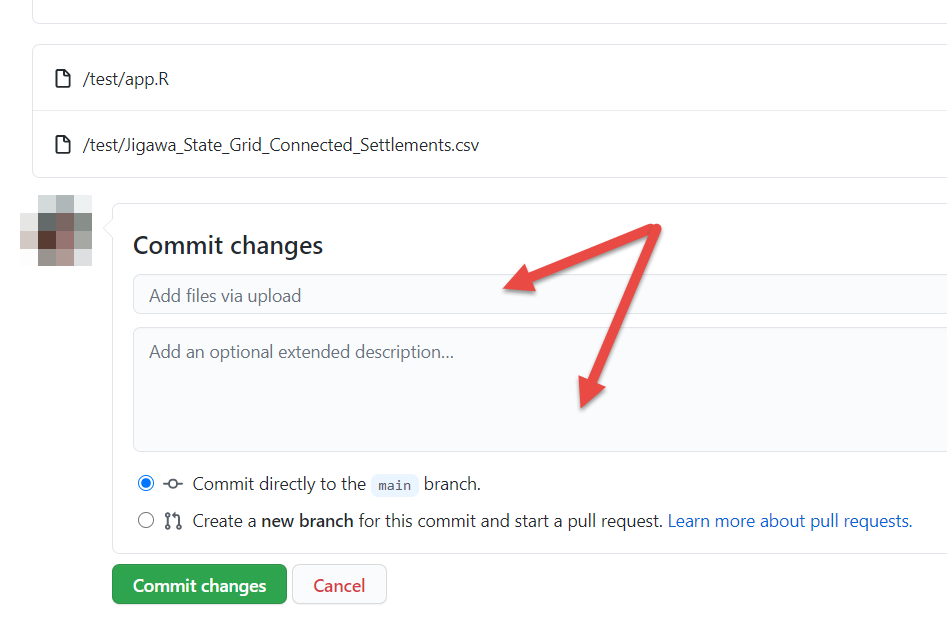
Aparecerá una pantalla para arrastrar archivos o seleccionarlos de disco

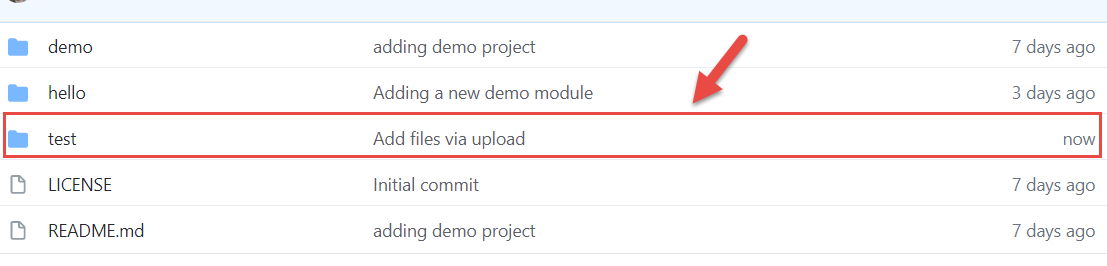


Selecione una carpeta completa. Usualmente la carpeta contiene un archivo con extensión .R (ej. demo.R).

A continuación ingrese el comentario y descripción en los campos que aparecen a continuación.

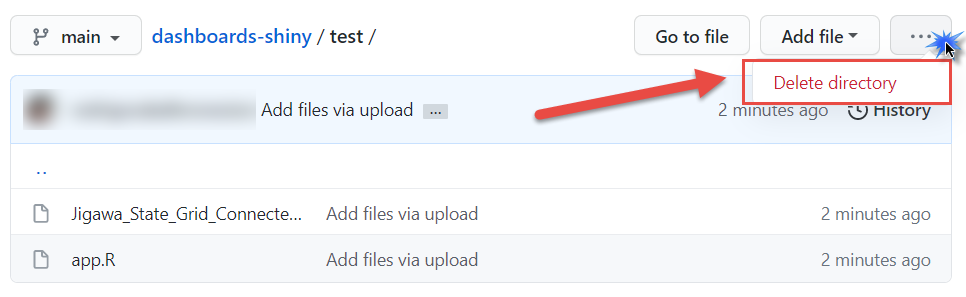
Seleccione la opción: “Commit directly to the main branch.”



Presione: “Commit changes” y podrá ver los archivos subidos junto con el contenido  


## Eliminación de archivos

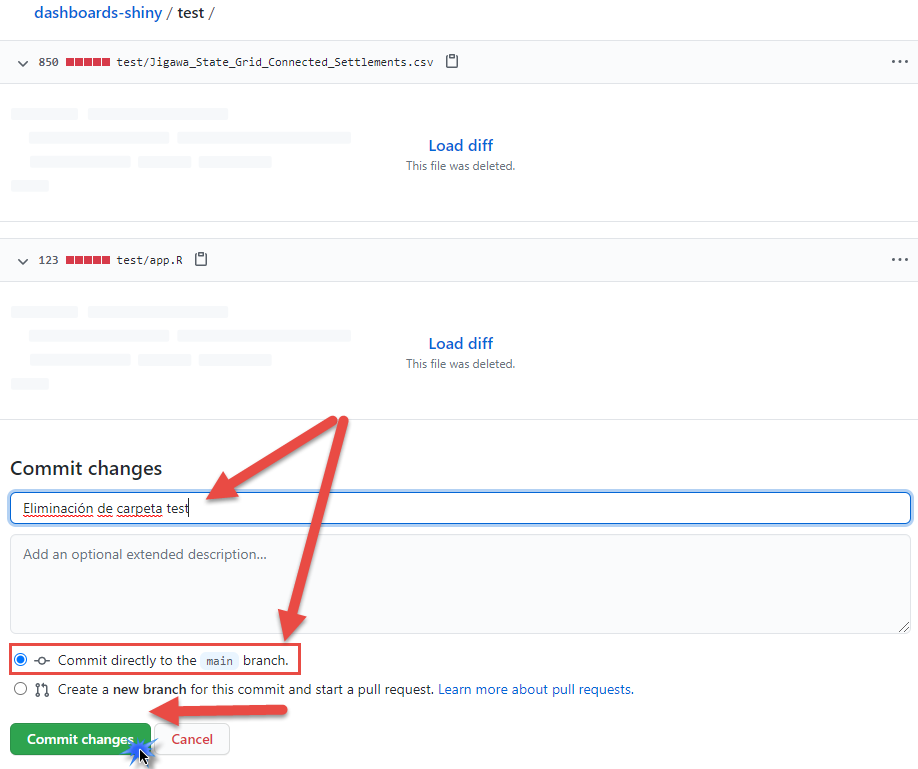
Haga clic en la carpeta que desea eliminar. A continuación clic en el botón con tres puntos “...”



Haga clic en “Delete directory”

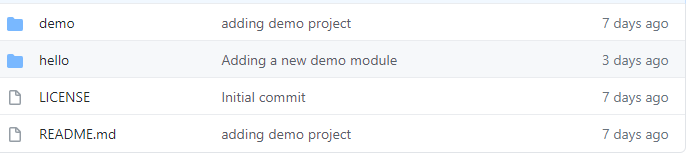
Llene las casillas de comentario, seleccione la opción “Commit directly to the main branch.”

Y presione “Commit changes”



Y presione “Commit changes”

Volverá a la pantalla general y la carpeta habrá desaparecido



# Despliegue automático en servidor (Paso 3)

Si realizó cambios en el repositorio de Github, los mismos serán actualizados dentro del sistema de ProAgenda2030 cada 20 minutos o el tiempo que se haya configurado al instalar el sistema.

El nombre de la carpeta que subió a Github será la ruta que se requiere para que se asigne la creación de un recurso en la base de datos y se pueda dar acceso a los dashboards a diferentes roles.

Debido a que los roles son mantenidos directamente en la base de datos, no existe una Interfaz de usuario, por lo que deberá seguir los pasos a continuación descritos

# Solicitar nuevo recurso a sys admin (Paso 4)

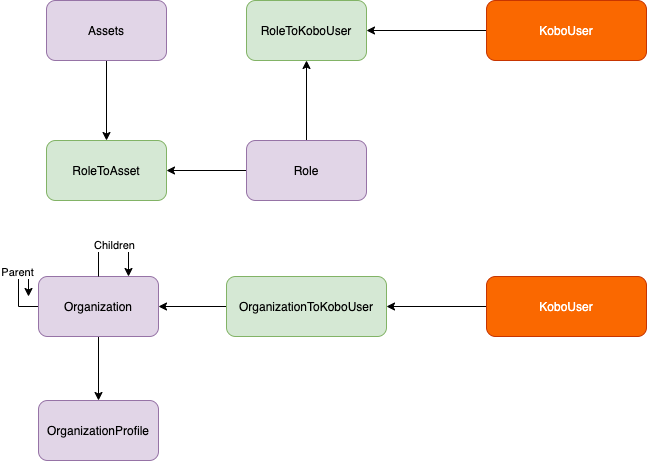
El alcance del proyecto sólo espera recursos fijos en el sistema, por lo que deberá solicitar al administrador del sistema que agregue:

* Recursos
* Asociación de Recursos a Roles

Solicite la creación de este par de acciones para poder visualizar correctamente recursos.

## Data Model para sys admins

A continuación el data model para referencia del administrador de sistema:



Las tables que deben tener los nuevos registros son:

* Assets
* RoleToAsset

# Scripts para Sys Admins

Los siguientes scripts se los debe ejecutar en una sesión SSH dentro del servidor

Recuerde reemplazar los valores entre signos “<” y “>” ej. <DB-PASSWORD>, <PRIVATE-IP>, <ASSET-NAME>, <ASSET-PATH>

### Script para agregar nuevos Role/RoleToAsset

# Execute all this inside Linux machine with Docker containers

export postgres\_pass=<DB-PASSWORD>

export support\_db=support

export target\_server=<PRIVATE-IP> # private IP since public is not accessible due to security groups

export asset\_id=<ASSET\_ID> # This value should change for each new Asset

export asset\_name=<ASSET-NAME>

export asset\_path=<ASSET-PATH>

export type=shiny

PGPASSWORD=${postgres\_pass} psql -d ${support\_db} -U user --port 5440 -h ${target\_server} -c "insert into public.\"Asset\" values (${asset\_id}, '${asset\_name}', '${asset\_path}', '${type}')";

# Roles:

# RoleId | Name

#--------+--------------------

# 1 | Administrador

# 2 | Administrador GAM

# 3 | Solo Vista GAM

# 4 | Administrador CAPY

# 5 | Solo Vista CAPY

# 6 | Visor de reportes

export role\_id=<ROLE\_ID> # Pick any of above

PGPASSWORD=${postgres\_pass} psql -d ${support\_db} -U user --port 5440 -h ${target\_server} -c "insert into public.\"RoleToAsset\" values (${role\_id}, ${asset\_id})";

### Script para Eliminar Role/RoleToAsset

export asset\_id=<ASSET\_ID> # This value should change for each new Asset

export role\_id=<ROLE\_ID> # Pick any of above

# DELETE - Use with care

PGPASSWORD=${postgres\_pass} psql -d ${support\_db} -U user --port 5440 -h ${target\_server} -c "delete from public.\"RoleToAsset\" where \"AssetId\"=${asset\_id}";

PGPASSWORD=${postgres\_pass} psql -d ${support\_db} -U user --port 5440 -h ${target\_server} -c "delete from public.\"Asset\" where \"AssetId\"=${asset\_id}";

### Script para realizar el poll de dashboards directamente en el container

# Update of Dashboards manually

docker exec -it support\_dashboards\_1 sudo bash -c "sudo bash /shiny-scripts/poll.sh