Water Data Project para Groots

Deployment – Support API – Docker estructures

[Alcance 1](#_Toc59457709)

[Containers docker 1](#_Toc59457710)

[Support API 2](#_Toc59457711)

[Generación de imagen 2](#_Toc59457712)

[Variables de entorno y Conexión a base de datos 3](#_Toc59457713)

[Base de datos 4](#_Toc59457714)

[Support Bootstrap 4](#_Toc59457715)

[Generación de imagen 4](#_Toc59457716)

[Entrypoint 5](#_Toc59457717)

[Healthcheck 6](#_Toc59457718)

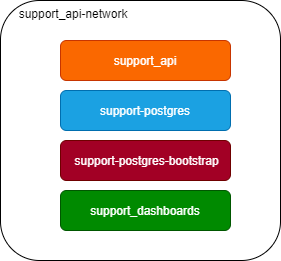
# Alcance

Se muestran los elementos a considerar para realizar mantenimiento a los scripts de docker dentro de Support API. Asimismo se hace referencia a elementos que son útiles en el proyecto instalador Support Install.

# Containers docker

Support API está compuesta de 4 containers:

* Support\_api
  + Código .net de backend
* Support-postgres
  + Base de datos PostgreSQL12
* Support-postgres-bootstrap
  + Código de bootstrap. Este container usualmente está detenido (código 0 significa que se ejecutó sin errores)
* support\_dashboards
  + Container que implementa Shiny dashboards
  + Este container auto actualiza (cada 20 minutos) los archivos de extensión R para tener nuevos análisis para ser visualizados en la aplicación (requiere asociación de roles)



# Support API

**Repo**: <https://github.com/kobo-ProAgenda2030/support-api>

* Este proyecto contiene todas las funcionalidad de Backend y la conexión a la base de datos.
* Contiene un archivo Dockerfile en ./Dockerfile
* Se utiliza el GitHub Action: .github/workflows/build-push.yml
* Se usan Tags con el prefijo “***pa.***” para ejecutar el mencionado GitHub Action

## Generación de imagen

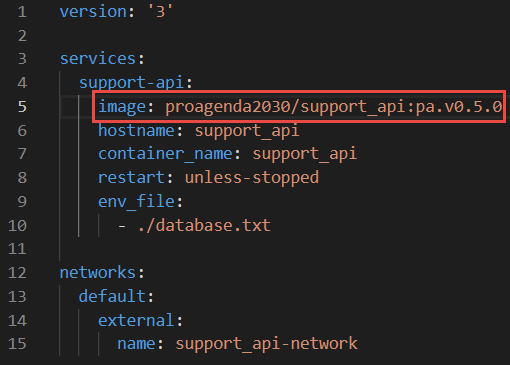
* Para generar la imagen debe crear un tag con el prefijo “**pa.**”

|  |
| --- |
| git tag pa.v#.#.#    git push origin pa.v#.#.# |

* Se podrá revisar el progreso en
  + <https://github.com/kobo-ProAgenda2030/support-api/actions>
* La imagen de docker se encuentra en:
  + <https://hub.docker.com/repository/docker/proagenda2030/support_api>

Una vez con la imagen se tendrá que realizar la actualización en support-install:

* **./support-install/deploy/templates/support-api-env/docker-compose.frontend.yml.tpl**

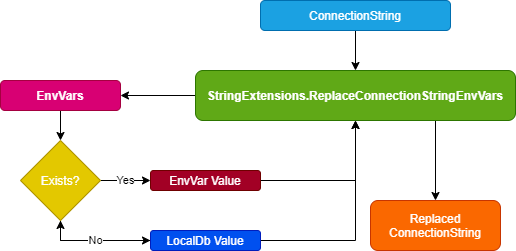


* Docker network: **support\_api-network**
  + Donde *support* es el nombre del proyecto docker para identificar al prefijo de red

## Variables de entorno y Conexión a base de datos

La conexión a la base de datos es se ha basado en el uso de variables de entorno, esto para evitar subir a Git los datos de conexión de producción en un futuro.

Las variables de entorno son reemplazadas en tiempo de ejecución. Si las mismas no han sido determinadas se usan valores default del archivo: ***Support.API.Services\Helpers\LocalDb.cs***



La conexión a la base de datos tiene este aspecto:

|  |
| --- |
| Server=SUPPORT\_DB\_INTERNAL\_SERVER; Port=SUPPORT\_DB\_INTERNAL\_PORT; Database=SUPPORT\_DB\_NAME; Username=SUPPORT\_DB\_USER; Password=SUPPORT\_DB\_PASSWORD |

# Base de datos

* Todo basado en docker
* Se usa la imagen de postgreSQL: **postgres:12.5-alpine**
* Ubicación del archivo template de docker-compose en support-install:
  + deploy\templates\support-api-env\docker-compose.db.yml.tpl
* La inicialización está a cargo de run.py (support-install)
* Docker network: **support\_api-network**
  + Donde *support* es el nombre del proyecto docker para identificar al prefijo de red

# Support Bootstrap

**Repo**: <https://github.com/kobo-ProAgenda2030/support-api>

* Proyecto consola
* Parte de Support-API
* Objetivo: Ejecutar código de inicialización de Support-API
* Contiene un archivo Dockerfile en ./Support.Bootstrap/bootstrap.Dockerfile
* Se utiliza el GitHub Action: .github/workflows/build-push-bootstrap.yml
* Se usan Tags con el prefijo “***bootstrap.***” para ejecutar el mencionado GitHub Action

## Generación de imagen

* Para generar la imagen debe crear un tag con el prefijo “**bootstrap.**”

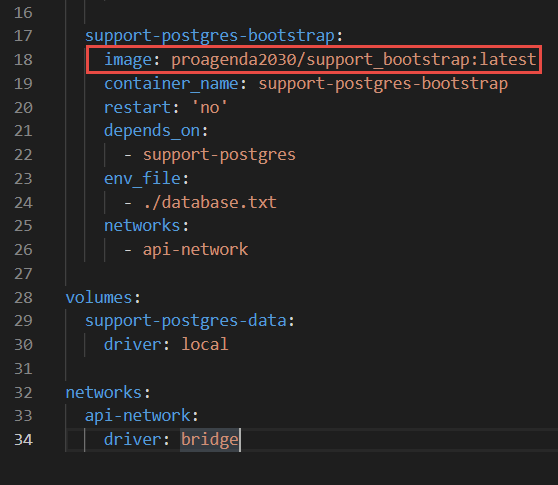
|  |
| --- |
| git tag bootstrap.v#.#.#  git push origin bootstrap.v#.#.# |

* Se podrá revisar el progreso en
  + <https://github.com/kobo-ProAgenda2030/support-api/actions>
* La imagen de docker se encuentra en:
  + <https://hub.docker.com/repository/docker/proagenda2030/support_bootstrap>

Una vez con la imagen se puede realizar la actualización en support-install:

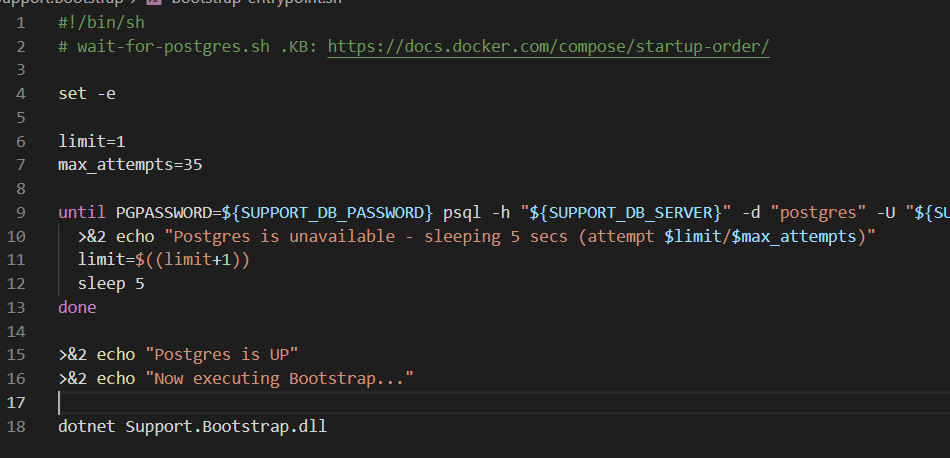
* **./support-install/deploy/templates/support-api-env/ docker-compose.db.yml.tpl**

Sin embargo éste ya apunta a latest



## Entrypoint

* Bootstrap requiere que la base de datos esté disponible
* Siendo ***Support.Bootstrap/bootstrap-entrypoint.sh*** el script que valida que postgresql es accesible
* Una vez se tiene esta validación realizada, se ejecuta la aplicación de consola



## Healthcheck

Este proyecto genera un contenedor que siempre termina en estado Stopped. Por lo que será necesario revisar containers con:

|  |
| --- |
| docker ps -a |

|  |
| --- |
| 70c5967eba29 proagenda2030/support\_bootstrap:latest "sh bootstrap-entryp…" 9 hours ago Exited (0) 9 hours ago support-postgres-bootstrap |

Revise los logs del container:

|  |
| --- |
| docker logs support-postgres-bootstrap |

Resultado:

|  |
| --- |
| psql: error: could not connect to server: Connection refused  Is the server running on host "support-postgres" (172.18.0.2) and accepting  TCP/IP connections on port 5432?  Postgres is unavailable - sleeping 5 secs (attempt 1/35)  Postgres is UP  Now executing Bootstrap...  [13:31:22 INF] Generating db create script...  [13:31:24 INF] DB Script executed |

# Dashboards (Docker)

**Repo**: <https://github.com/kobo-ProAgenda2030/dashboards-docker>

* Contiene el archivo Dockerfile usado para manejar la distribución **rocker/shiny**
* Objetivo: **Servir archivos R para Shiny Server**
* Contiene un archivo Dockerfile en ./Dockerfile
* Se utiliza el GitHub Action: .github/workflows/build-push.yml
* Se usan Tags con el prefijo “***pa.***” para ejecutar el mencionado GitHub Action

## Generación de imagen

* Para generar la imagen debe crear un tag con el prefijo “**pa.**”

|  |
| --- |
| git tag pa.#.#.#  git push origin pa.#.#.# |

* Se podrá revisar el progreso en
  + <https://github.com/kobo-ProAgenda2030/dashboards-docker/actions>
* La imagen de docker se encuentra en:
  + <https://hub.docker.com/repository/docker/proagenda2030/dashboards>

Una vez con la imagen se puede realizar la actualización en support-install:

* **./support-install/deploy/templates/support-api-env/docker-compose.shiny.yml.tpl**

## Entrypoint

* El archivo Dockerfile del repositorio dashboards-docker no contiene un entrypoint
* Sin embargo el Entrypoint sí se encuentra en el archivo docker-compose de:
  + **./support-install/deploy/templates/support-api-env/docker-compose.shiny.yml.tpl**
  + Línea 13:
    - command: "bash /shiny-scripts/entrypoint.sh"
* El archivo /shiny-scripts/entrypoint.sh realiza:
  + Registra un cron job dentro del container: *support\_dashboards*
    - La ejecución se realiza cada 20 minutos
    - El script hace un git pull del repositorio:
    - <https://github.com/kobo-ProAgenda2030/dashboards-shiny>
    - Se copian los contenidos a /srv/shiny-server/ (dentro del container)
* Se llama al comando llamado por Shiny

## Healthcheck

Una vez que el container está desplegado, se pueden revisar los logs a través de:

|  |
| --- |
| docker logs support\_dashboards\_1 |

Los mensajes deben hacer referencia a que Shiny Server está funcional