Water Data Project para Groots

Deployment – Flujo de despliegue – Fase 2

[Alcance 2](#_Toc68764128)

[Supuestos 2](#_Toc68764129)

[Productos afectados 2](#_Toc68764130)

[Flujo de despliegue - Desarrolladores 3](#_Toc68764131)

[Flujo de despliegue en KPI 3](#_Toc68764132)

[Flujo de despliegue en Support API 4](#_Toc68764133)

[Flujo de despliegue en kobo-docker 4](#_Toc68764134)

[Flujo de despliegue en support-install 5](#_Toc68764135)

[Runbook para Developers 6](#_Toc68764136)

[Groots-KPI 6](#_Toc68764137)

[KPI 6](#_Toc68764138)

[Support API 7](#_Toc68764139)

[Dashboards Docker 8](#_Toc68764140)

[Runbook para Deployers 9](#_Toc68764141)

[Actualización de Docker Images 9](#_Toc68764142)

[Actualizar kobo-docker 9](#_Toc68764143)

[Actualizar support-install 9](#_Toc68764144)

[Actualizar dashboards-docker 10](#_Toc68764145)

[Instalación de KoBoToolbox 10](#_Toc68764146)

[Instalación de Support API 10](#_Toc68764147)

[Si es la primera vez 10](#_Toc68764148)

[Si support-api ya ha sido instalada previamente 11](#_Toc68764149)

[Revisar logs de docker 11](#_Toc68764150)

[Seed de base de datos 12](#_Toc68764151)

[Consideraciones 12](#_Toc68764152)

[Certificados SSL 12](#_Toc68764153)

[Certificados de Support en KoBoToolbox 13](#_Toc68764154)

[Actualización de Certificados manualmente (deprecated) 13](#_Toc68764155)

[Usando LetsEncrypt 14](#_Toc68764156)

[Archivo Nginx support-api.conf 16](#_Toc68764157)

[Sin usar LetsEncrypt 16](#_Toc68764158)

[Archivo .conf de ejemplo 17](#_Toc68764159)

[Instalando SSL de Support API 18](#_Toc68764160)

# Alcance

El término Runbook hace referencia a los pasos necesarios para llevar a cabo una tarea de manera expectable y reproducible en el tiempo.

El siguiente Runbook está diseñado para realizar:

* Pasos para empaquetar cambios realizados por los desarrolladores
* Pasos para preparar el ambiente de despliegue
* Instalación de KoBoToolbox
* Instalación de Support API

Para la elaboración de este documento de instrucciones se utilizó una máquina **Ubuntu Server 18.04.1 LTS**. Se puede usar un distro diferente y se recomienda que se trate de un Long Term Support (LTS).

## Supuestos

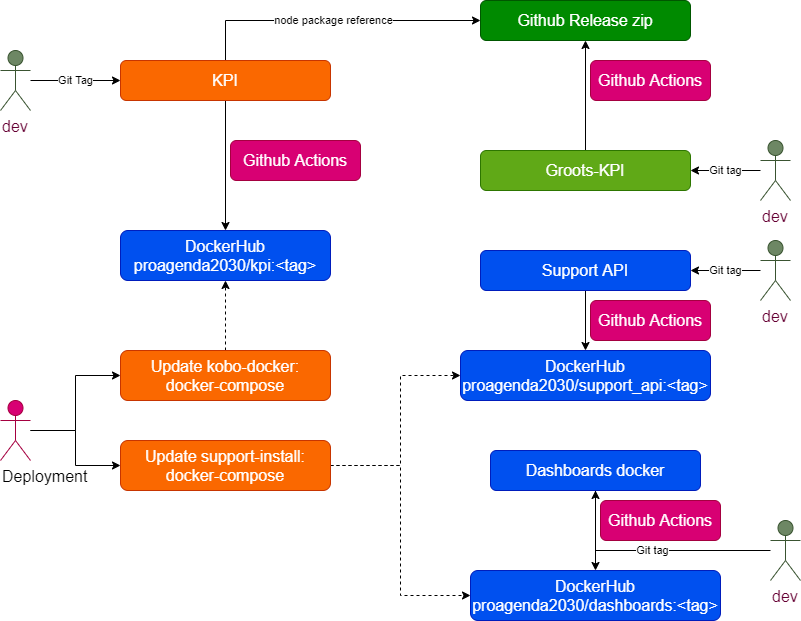
* Se asume que se han seguido los pasos del Runbook: ***Deployment - Runbook - Fase 1*** para tener Docker y Python3 instalados.
* Se conocen las custimizaciones a nivel de código exploradas en el documento: ***Development – KoBoToolbox customizations***

## Productos afectados

* Forks de los repositorios de KoBoToolbox:
  + KPI
  + kobo-docker
  + kobo-install
  + Collect
* Repositorios Support
  + Support API
  + Support Install
  + Dashboards Docker
  + Dashboards Shiny

### 

# Flujo de despliegue - Desarrolladores



El flujo explicado en el diagrama representan las interacciones directas e indirectas a realizar.

Características:

* No todas son necesarias al mismo tiempo
* Debe existir comunicación entre el personal de Deployment y el Desarrollador (dev)
* Las acciones de un dev llegan hasta la generación de imágenes de Docker en DockerHub, lo cual debe ser notificado a Deployment.

Se distinguen tres elementos que son responsabilidad del desarrollador:

* groots-KPI
* KPI (fork)
* support-API
* Dashboards docker

La persona encargada del deployment tiene a responsabilidad dos repositorios:

* kobo-docker (fork)
* support-install
* dashboards-docker

## Flujo de despliegue en KPI

KPI hace referencia al paquete de groots-KPI, el mismo que contiene todas las customizaciones necesarias a aplicar a KPI. La referencia se realiza hacia una versión específica de los releases de GitHub, por lo que cada cambio de paquete en groots-KPI significa una actualización en el archivo packages.json de KPI.

Cualquier cambio en el desarrollo o actualización a versión de groots-KPI implica la generación de una nueva imagen de KPI en DockerHub. El proceso es a través de la generación de un tag de Git que debe cumplir con la expresión requerida en *./.github/workflow/push-to-dockerhub.yml* (ln. 7).

Una vez que se ha generado la imagen en DockerHub, ésta debe ser referenciada en el repositorio de kobo-docker (fork).

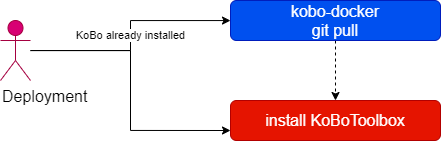
## Flujo de despliegue en Support API

Support API es la aplicación web api que soporta las características custom requeridas por groots-KPI como backend. Es una aplicación bastante independiente de KoBoToolbox al no requerir de su instalador, base de datos o Docker network para funcionar. Sin embargo sí existe una dependencia de su servidor Nginx cuando se usan certificados SSL, esto sólo cuando Support API se instala en la misma máquina en la que corre KoBoToolbox.

Cualquier cambio en el desarrollo o actualización en Support API implica la generación de una nueva imagen en DockerHub. El proceso es a través de la generación de un tag de Git que debe cumplir con la expresión requerida en *./.github/workflow/build-push.yml* (ln. 7).

Una vez que se ha generado la imagen en DockerHub, ésta debe ser referenciada en el repositorio de support-install.

## Flujo de despliegue en kobo-docker



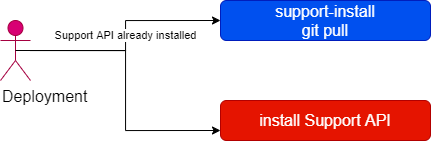
Se deberá realizar una actualización en el archivo *docker-compose.frontend.yml* sólamente para la imagen de KPI (ln. 47) en caso de que ésta haya cambiado.

Una vez que el cambio fue realizado, se deberá ejecutar la instalación de KoBoToolbox en la máquina destino.

**Caso 1**: Si la primera vez (KoBoToolbox no fue instalado antes). Sólo debe seguir la instalación normal, no se requiere hacer ninguna otra acción.

**Caso 2**: Si KoBoToolbox ya fue instalado antes. Se deberá dirigir al directorio de kobo-docker y ejecutar ***git pull***, se debería ver que el cambio es actualizado localmente. Después de esto se debería proceder a ejecutar ***python run.py -s*** en kobo-install.

## Flujo de despliegue en support-install



Se deberá realizar una actualización en el archivo *deploy\templates\support-api-env\docker-compose.frontend.yml.tpl* (ln. 6) en caso de que éste haya cambiado.

Se deberá realizar una actualización en el archivo *deploy\templates\support-api-env\ docker-compose.shiny.yml.tpl* (ln. 5) en caso de que éste haya cambiado.

Una vez que el cambio fue realizado, haga commit. En caso de no encontrarse en el branch principal (de despliegue) realice el merge correspondiente a éste

En la máquina destino navegue al directorio support-install y ejecute

|  |
| --- |
| git pull |

Asegúrese que el branch es el de despliegue

|  |
| --- |
| git branch |

# Runbook para Developers

**Supuesto:** Comprensión de los flujos de despliegue para desarrolladores descritos anteriormente.

## Groots-KPI

* Todos los cambios se deben hacer hacia el Branch: **proagenda2030/master**
* Los branches deben salir de éste como padre y volver como un Pull Request
* El objetivo es generar un artefacto como parte de un Release en GitHub

|  |
| --- |
| 1. Haga commit / push del código hacia el origen remoto |

|  |
| --- |
| 1. (Opcional) Si el push fue en un feature branch, realice el PR y ejecute el merge 2. (Opcional) use el branch proagenda2030/master **git checkout proagenda2030/master** 3. Obtenga los últimos cambios del branch proagenda2030/master   **git pull** |

|  |
| --- |
| 1. Realice Git Tag **git tag v#.#.#**   ejemplo: git tag v1.0.0   1. Realice Push del Tag hacia el origen **git push origin v#.#.#** ejemplo**:** git push origin v1.0.0 |

|  |
| --- |
| 1. Revise que la acción en GitHub se encuentra en proceso y luego terminada <https://github.com/NexionBolivia/groots-kpi/actions> |

|  |
| --- |
| 1. Revise que se ha generado un nuevo release con el Tag realizado <https://github.com/NexionBolivia/groots-kpi/releases> |

# KPI

* Todos los cambios se deben hacer hacia el Branch: **proagenda2030/master**
* Los branches deben salir de éste como padre y volver como un Pull Request
* El objetivo es generar una nueva docker image en DockerHub

|  |
| --- |
| 1. Haga commit / push del código hacia el origen remoto |

|  |
| --- |
| 1. (Opcional) Si el push fue en un feature branch, realice el PR y ejecute el merge 2. (Opcional) use el branch proagenda2030/master **git checkout proagenda2030/master** 3. Obtenga los últimos cambios del branch proagenda2030/master   **git pull** |

|  |
| --- |
| 1. Realice Git Tag **git tag pa.v#.#.#**   ejemplo: git tag pa.v1.0.0   1. Realice Push del Tag hacia el origen **git push origin pa.v#.#.#** ejemplo**:** git push origin pa.v1.0.0 |

|  |
| --- |
| 1. Revise que la acción en GitHub se encuentra en proceso y luego terminada <https://github.com/NexionBolivia/kpi/actions> |

|  |
| --- |
| 1. Revise que la imagen se encuentra en DockerHub <https://hub.docker.com/repository/docker/proagenda2030/kpi> |

# Support API

* Todos los cambios se deben hacer hacia el Branch: **main**
* Los branches deben salir de éste como padre y volver como un Pull Request
* El objetivo es generar una nueva docker image en DockerHub

|  |
| --- |
| 1. Haga commit / push del código hacia el origen remoto |

|  |
| --- |
| 1. (Opcional) Si el push fue en un feature branch, realice el PR y ejecute el merge 2. (Opcional) use el branch main **git checkout main** 3. Obtenga los últimos cambios del branch master   **git pull** |

|  |
| --- |
| 1. Realice Git Tag **git tag pa.v#.#.#**   ejemplo: git tag pa.v1.0.0   1. Realice Push del Tag hacia el origen **git push origin pa.v#.#.#** ejemplo**:** git push origin pa.v1.0.0 |

|  |
| --- |
| 1. Revise que la acción en GitHub se encuentra en proceso y luego terminada <https://github.com/NexionBolivia/support-api/actions> |

|  |
| --- |
| 1. Revise que la imagen se encuentra en DockerHub https://hub.docker.com/repository/docker/proagenda2030/support\_api |

# Dashboards Docker

* Usualmente los cambios se realizarán sobre el archivo Dockerfile para instalar nuevos packages de linux. Sin embargo es probable que no sea necesario modificar este proyecto
* Todos los cambios se deben hacer hacia el Branch: **main**
* Los branches deben salir de éste como padre y volver como un Pull Request
* El objetivo es generar una nueva docker image en DockerHub

|  |
| --- |
| 1. Haga commit / push del código hacia el origen remoto |

|  |
| --- |
| 1. (Opcional) Si el push fue en un feature branch, realice el PR y ejecute el merge |

|  |
| --- |
| 1. (Opcional) use el branch master **git checkout main** |

|  |
| --- |
| 1. Obtenga los últimos cambios del branch master   **git pull** |

|  |
| --- |
| 1. Realice Git Tag **git tag pa.v#.#.#**   ejemplo: git tag pa.v1.0.0 |

|  |
| --- |
| 1. Realice Push del Tag hacia el origen **git push origin pa.v#.#.#** ejemplo**:** git push origin pa.v1.0.0 |

|  |
| --- |
| 1. Revise que la acción en GitHub se encuentra en proceso y luego terminada [https://github.com/NexionBolivia/dashboards-docker/actions](https://github.com/NexionBolivia/supportd/actions) |

|  |
| --- |
| 1. Revise que la imagen se encuentra en DockerHub https://hub.docker.com/repository/docker/proagenda2030/dashboards |

# Runbook para Deployers

**Supuestos:**

* Comprensión de los flujos de despliegue descritos anteriormente.
* Se ha leído el documento: ***Deployment - Runbook - Fase 1***

**General:**

* Se usan dos repos:
  + <https://github.com/NexionBolivia/kobo-install>
  + <https://github.com/NexionBolivia/support-install>
* Se debe ejecutar dentro de un ambiente de producción con:
  + Acceso a internet
  + IP pública
  + Puertos 80 y 443 accesibles

## Preparacion de Docker Images

Deben armarse las imágenes de manera local para tener la ultima version con los siguientes comandos:

|  |
| --- |
| docker build -t proagenda2030/kpi:pa.v0.7 .  <https://github.com/kobo-ProAgenda2030/kpi> |
|  |
| docker build -t proagenda2030/dashboards:pa.1.0.4 .  <https://github.com/kobo-ProAgenda2030/dashboards-docker> |
| docker build -t proagenda2030/support\_postgres\_backup .  <https://github.com/kobo-ProAgenda2030/support-install>  el dockerfile esta dentro de /support-install/build/postgres-backup |
| docker build -t proagenda2030/support\_api:pa.v1.0.0 .  <https://github.com/kobo-ProAgenda2030/support-api> |

## Actualización de Docker Images

### Actualizar kobo-docker

|  |
| --- |
| 1. cd kobo-docker 2. git checkout proagenda2030/master 3. git pull |

|  |
| --- |
| 1. Abra el archivo: *docker-compose.frontend.yml* |

|  |
| --- |
| 1. Modifique la línea 47 Específicamente el nombre del Tag al que se quiere actualizar |

|  |
| --- |
| 1. Realice el Git commit y push correspondiente Si se hizo en otro branch que no sea proagenda2030/master realice el PR y merge |

### Actualizar support-install

|  |
| --- |
| 1. cd support-install 2. git checkout main 3. git pull |

|  |
| --- |
| 1. Abra el archivo: *deploy\templates\support-api-env\docker-compose.frontend.yml.tpl* |

|  |
| --- |
| 1. Modifique la linea 6 Específicamente el nombre del Tag al que se quiere actualizar |

|  |
| --- |
| 1. Realice el Git commit y push correspondiente Si se hizo en otro branch que no sea main realice el PR y merge |

### Actualizar dashboards-docker

|  |
| --- |
| 1. cd support-install |

|  |
| --- |
| 1. git checkout main |

|  |
| --- |
| 1. git pull |

|  |
| --- |
| 1. Abra el archivo: *deploy\templates\support-api-env\ docker-compose.shiny.yml.tpl* |

|  |
| --- |
| 1. Modifique la linea 5 Específicamente el nombre del Tag al que se quiere actualizar |

|  |
| --- |
| 1. Realice el Git commit y push correspondiente Si se hizo en otro branch que no sea main realice el PR y merge |

## Instalación de KoBoToolbox

Para una primera vez (KoBoToolbox no ha sido instalado), siga todas las instrucciones del documento ***Deployment - Runbook - Fase 1.***

En caso de tener una actualización en KPI, Groots-KPI u otro, siga los siguientes pasos:

git clone –b proagenda2030/master https://github.com/NexionBolivia/kobo-install.git

|  |
| --- |
| 1. cd <directorio-instalacion>/kobo-docker |

|  |
| --- |
| 1. git pull |

|  |
| --- |
| 1. cd ../kobo-install |

|  |
| --- |
| 1. python3 run.py -s |

|  |
| --- |
| 1. Siga todos los pasos sin modificar ningún valor (haga Enter en todos los pasos) Espere a que el entorno esté disponible por Web nuevamente |

## Instalación de Support API

### Si es la primera vez

|  |
| --- |
| 1. cd <directorio-instalacion> |

|  |
| --- |
| 1. git clone https://github.com/NexionBolivia/support-install.git |

|  |
| --- |
| 1. cd support-install/deploy |

|  |
| --- |
| 1. python3 run -s |

|  |
| --- |
| 1. Siga todos los pasos requeridos por el instalador   Importante: las contraseñas no deben contener caracteres especiales, para evitar los errores de conexión.  las credenciales de kobotoolbox los puede conseguir con el siguiente comando:  cat .run.conf dentro de la carpeta de kobo-install  Para obtener el token de kobotoolbox utilice el siguiente comando:  curl -u username:password "https:/[kpi-url]/token/?format=json" |

Opcional: Healthchecks

|  |
| --- |
| 1. curl <url>/healthcheck Resultado esperado: “ok” |

Database healthcheck

|  |
| --- |
| 1. curl <url>/healthcheck/db Resultado esperado: “ok” |

### Si support-api ya ha sido instalada previamente

|  |
| --- |
| 1. cd <directorio-instalacion>/support-install |

|  |
| --- |
| 1. git pull |

|  |
| --- |
| 1. cd deploy |

|  |
| --- |
| 1. python3 run.py -s |

|  |
| --- |
| 1. Siga todos los pasos sin modificar ningún valor (haga Enter en todos los pasos) Espere a que el entorno esté disponible por Web nuevamente |

Opcional: healthchek

|  |
| --- |
| 1. curl <url>/healthcheck Resultado esperado: “ok” |

Database healthcheck

|  |
| --- |
| 1. curl <url>/healthcheck/db Resultado esperado: “ok” |

### Revisar logs de docker

Puede revisar los logs de docker en caso de ser necesario. Para ello debe estar en la máquina donde instaló KoboToolbox y Support-API

Obtenga la lista de contenedores de docker:

|  |
| --- |
| 1. docker ps |

* Si no encuentra el container que busca, tal vez éste está en estado Stopped. Para evidenciarlo ejecute:

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

Copie el nombre del contenedor del que desea revisar logs:

|  |
| --- |
| docker logs <nombre-contenedor> |

La lista de nombres de contenedores al momento es:

* support\_api
* support-postgres
* support\_dashboards\_1
* nginxcertbot\_certbot\_1
* nginxcertbot\_nginx\_1
* kobofe\_kobocat\_1
* kobofe\_kpi\_1
* kobofe\_enketo\_express\_1
* kobofe\_nginx\_1
* kobobe\_redis\_main\_1
* kobobe\_mongo\_1
* kobobe\_postgres\_1
* kobobe\_redis\_cache\_1

Puede ingresar dentro de los contenedores con el comando:

|  |
| --- |
| docker exec –it <nombre-contenedor> bash |

**Nota**: En algunos casos el comando ***bash*** no está instalado, use ***sh*** en su lugar

Para revisar datos internos del container puede ejecutar:

|  |
| --- |
| docker inspect <nombre-contenedor> |

## Seed de base de datos

Seed es un endpoint que registra los datos mínimos dentro del sistema. Entre ellos están:

* 1 Profile
* 1 Organization
* 2 Users
* 2 Users Kobo

Definición API :

|  |
| --- |
| PUT /**seed** |
| Body (JSON):  {  "migrate": **true|false**,  "deletedb": **true|false**  } |
| Requires Authorization: true (Bearer token) |

### Consideraciones

* El parámetro “deletedb” sólo se considerará cuando el parámetro “migrate” es true
* “migrate”: true, ejecutará todas las migraciones que existan en support API

### Información importante

La funcionalidad de “Seed” ha sido descontinuada, si no se logra generar el “Seed” como explicado arriba, hay una opción alternativa:

* Importar los datos mínimos desde el archivo “[support\_backup](https://github.com/kobo-ProAgenda2030/documentation/blob/main/Deployment/scripts/support_backup)” (sql inicial)

## Certificados SSL

Support API necesita ser accedido a través de HTTPS, por lo que será necesario que se implemente un certificado digital.

Como se sugiere en el documento ***Deployment - Runbook - Fase 1***, se puede usar LetsEncrypt que es una organización que crea certificados gratuitos. También es posible utilizar un certificado propio, lo cual implica una configuración específica.

### Certificados de Support en KoBoToolbox

**Esta es la forma recomendada cuando se quieren usar certificados de Let’sEncrypt**

Se requieren dos certificados para el funcionamiento de support:

* Certificado para Support API
* Certificado para Dashboards

Ambos certificados se han configurado como parte de los scripts de KoBoToolbox, específicamente el proyecto kobo-install en los archivos:

* [init-letsencrypt.sh.tpl](https://github.com/NexionBolivia/kobo-install/blob/proagenda2030/master/templates/nginx-certbot/init-letsencrypt.sh.tpl#L5)
* [dashboards.conf.tpl](https://github.com/NexionBolivia/kobo-install/blob/proagenda2030/master/templates/nginx-certbot/data/nginx/dashboards.conf.tpl)
* [support.conf.tpl](https://github.com/NexionBolivia/kobo-install/blob/proagenda2030/master/templates/nginx-certbot/data/nginx/support.conf.tpl)

Esto es suficiente para que los certificados sean generados como parte de KoBoToolbox, es decir que durante el proceso de instalación de certificados Let’sEncrypt en el wizard de instalación de kobo-install (python3 run.py -s), se generarán dos certificados a los subdominios:

* **support**.mi-dominio.com
* **dashboards**.mi-dominio.com

(Asumiendo que el dominio principal es mi-dominio.com, entonces los otros subdominios son: kf.mi-dominio.com, ee.mi-dominio.com, kc.mi-dominio.com)

Todos los subdominios se encuentran en:

* <instalación>/nginx-certbot/data/certbot/conf/archive

Se debería encontrar sólo una carpeta con el nombre del subdominio **kf.mi-dominio.com**

En caso de requerir un nuevo subdominio:

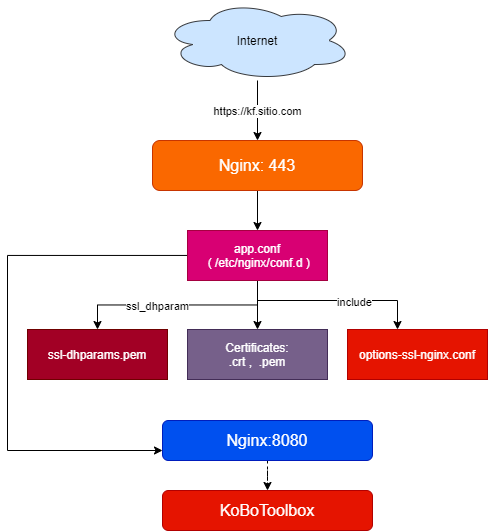
* Adicione el nuevo subdominio en el archivo init-letsencrypt.sh.tpl, línea 5
* Cree un nuevo archivo .conf.tpl en la misma ubicación de dashboards.conf.tpl, support.conf.tpl
  + En este archivo modifique el valor del subdominio
  + El puerto al que se tiene que navegar

### Actualización de Certificados manualmente (deprecated)

De cualquier manera se necesitarán realizar las siguientes acciones:

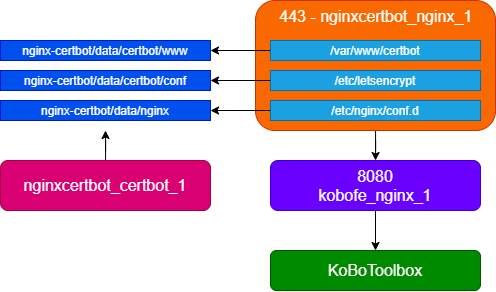
* Copiar los archivos del certificado digital en una locación específica.
* Adicionar un archivo .conf de nginx a una locación específica.
* Realizar un “reload” de nginx una vez se ha realizado estas dos acciones.

El siguiente diagrama muestra cómo los certificados están configurados en nginx.



#### Usando LetsEncrypt

LetsEncrypt es la manera sugerida, sin embargo la misma está embebida dentro de la infraestructura de despliegue de KoBoToolbox, por lo que este diagrama explica cómo funciona en éste despliegue:



**Requisitos**

* Contar con el par de certificados digitales
  + Estos archivos terminan en .crt y/o .pem
* Los certificados deben ser generados para el sitio:
  + support.*mi-dominio.tld*
  + Ejemplo: support.*nexion-dev.tk*
* Se deberá generar un archivo .conf para manejar el acceso al certificado

**Supuestos**

* Usted tiene los certificados generados con certbot
* Certificados en:
  + /<ruta-certs>/archive/support.nexion-dev.tk
* Certificados como symlink en:
  + /<ruta-certs>/live/support.nexion-dev.tk
* Archivo nginx en:
  + /<ruta-nginx-support>/support-api.conf

**Runbook:**

|  |
| --- |
| 1. Copiar los certificados del directorio archive sudo cp -R /<ruta-certs>/**archive**/support.nexion-dev.tk /<ruta-kobotoolbox>/nginx-certbot/data/certbot/conf/**archive**/support.nexion-dev.tk |

|  |
| --- |
| 1. Copiar los certificados del directorio live sudo cp -R /<ruta-certs>/**live**/support.nexion-dev.tk /<ruta-kobotoolbox>/nginx-certbot/data/certbot/conf/**live**/support.nexion-dev.tk |

|  |
| --- |
| 1. Copy nginx config file sudo cp -rf /<ruta-nginx-support>/support-api.conf /<ruta-kobotoolbox>/nginx-certbot/data/nginx/ |

|  |
| --- |
| 1. Reload Nginx docker exec -it nginxcertbot\_nginx\_1 sh -c 'nginx -s reload' |

##### Archivo Nginx support-api.conf

|  |
| --- |
| server {  listen 80;  server\_name support.mi-dominio.tld;  server\_tokens off;    location /.well-known/acme-challenge/ {  root /var/www/certbot;  }    location / {  return 301 https://$host$request\_uri;  }  }    server {  listen 443 ssl;  server\_name support.mi-dominio.tld;  server\_tokens off;    ssl\_certificate /etc/letsencrypt/live/support.mi-dominio.tld/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/support.mi-dominio.tld/privkey.pem;  include /etc/letsencrypt/options-ssl-nginx.conf;  ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;    # Allow 100M upload  client\_max\_body\_size 100M;    location / {  proxy\_pass http://MY\_PRIVATE\_IP:8500; # This is the private IP in AWS. Localhost cannot be reached directly due to nginx's own network  proxy\_set\_header Host $http\_host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-Forwarded-Proto https;  }  } |

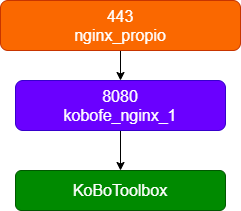
Los archivos **options-ssl-nginx.conf** y **ssl-dhparams.pem** ya deberían encontrarse creados, sin embargo estos se indican en la siguiente sección.

#### Sin usar LetsEncrypt

Si se ha configurado KoBoToolbox se asume que se han respondido las siguientes preguntas durante la instalación:

|  |
| --- |
| Auto-install HTTPS certificates with Let's Encrypt?  1) Yes  2) No - Use my own reverse-proxy/load-balancer  [2]: 2 |
| Is your reverse-proxy/load-balancer installed on this server?  1) Yes  2) No  [2]: 1 |
| Internal port used by reverse proxy?  [8080]: |

Se necesita una configuración de Nginx para poder servir al puerto 8080



##### Archivo .conf de ejemplo

**/etc/nginx/conf.d/app.conf**

|  |
| --- |
| server {  listen 80;  server\_name kf.mi-dominio.tld kc.mi-dominio.tld ee.mi-dominio.tld;  server\_tokens off;    location /.well-known/acme-challenge/ {  root /var/www/certbot;  }    location / {  return 301 https://$host$request\_uri;  }  }    server {  listen 443 ssl;  server\_name kf.mi-dominio.tld kc.mi-dominio.tld ee.mi-dominio.tld;  server\_tokens off;    ssl\_certificate /etc/nginx/certs/certificate.crt;  ssl\_certificate\_key /etc/nginx/certs/privateKey.key;  include /etc/nginx/certs/options-ssl-nginx.conf;  ssl\_dhparam /etc/nginx/certs/ssl-dhparams.pem;    # Allow 100M upload  client\_max\_body\_size 100M;    location / {  #proxy\_pass http://IP-ESPECIFICA:8080;  proxy\_pass http://localhost:8080;  proxy\_set\_header Host $http\_host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-Forwarded-Proto https;  }  } |

Archivo: **options-ssl-nginx.conf**

|  |
| --- |
| # This file contains important security parameters. If you modify this file  # manually, Certbot will be unable to automatically provide future security  # updates. Instead, Certbot will print and log an error message with a path to  # the up-to-date file that you will need to refer to when manually updating  # this file.    ssl\_session\_cache shared:le\_nginx\_SSL:10m;  ssl\_session\_timeout 1440m;  ssl\_session\_tickets off;    ssl\_protocols TLSv1.2 TLSv1.3;  ssl\_prefer\_server\_ciphers off;    ssl\_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA"; |

Archivo: **ssl-dhparams.pem**

|  |
| --- |
| -----BEGIN DH PARAMETERS-----  MIIBCAKCAQEA//////////+t+FRYortKmq/cViAnPTzx2LnFg84tNpWp4TZBFGQz  +8yTnc4kmz75fS/jY2MMddj2gbICrsRhetPfHtXV/WVhJDP1H18GbtCFY2VVPe0a  87VXE15/V8k1mE8McODmi3fipona8+/och3xWKE2rec1MKzKT0g6eXq8CrGCsyT7  YdEIqUuyyOP7uWrat2DX9GgdT0Kj3jlN9K5W7edjcrsZCwenyO4KbXCeAvzhzffi  7MA0BM0oNC9hkXL+nOmFg/+OTxIy7vKBg8P+OxtMb61zO7X8vC7CIAXFjvGDfRaD  ssbzSibBsu/6iGtCOGEoXJf//////////wIBAg==  -----END DH PARAMETERS----- |

##### Instalando SSL de Support API

Asegúrese que el archivo support-api.conf se encuentra en el directorio:

* /etc/nginx/conf.d

El archivo support-api.conf es el mismo que se explica en la sección de LetsEncrypt. Sin embargo, necesita asegurarse que los valores corresponden a los del directorio de instalación de Nginx.