

PORTAL ANDINO

Versión : 2.5.4

Portal Andino es un desarrollo de la Administración Pública Nacional basado en CKAN . Es la tecnología detrás del Portal Nacional de Datos Abiertos y de otros portales de datos abiertos de organismos de la APN, provincias y municipios.

- Comienzo rápido
- Desarrolladores
 - Instalación
 - Actualización
 - Checklist para la puesta en producción
 - Migración
 - Mantenimiento
 - Configuración HTTPS
 - Configuración de DNS
 - Desarrollo
 - Tests

COMIENZO RÁPIDO

INDICE

- Primeros pasos
- Entrar y salir de tu portal
- Permisos de usuario
- Elementos de tu portal
- Organizaciones
- Sección Organizaciones con datos
- Sección Acerca
 - Tipos de Acerca
 - ¿Cómo elijo el tipo de Acerca?
 - ¿Cómo puedo escribir y mostrar información básica?
 - ¿Cómo puedo crear y mostrar mis secciones personalizadas?
 - ¿Dónde añado los archivos requeridos para las secciones?
 - ¿Dónde y cómo puedo guardar imágenes para mostrarlas en mis secciones? ¿Cómo las muestro en mis templates?
- Temas
- Datasets
- Recursos
 - ¿Dónde los veo en el portal?

- Buenas prácticas al crear recursos
- ¿Cómo los creo?
- Campos de un recurso
 - ¿Dónde los veo en el portal?
 - Buenas prácticas al cargar campos de un recurso
- Series de tiempo
 - ¿Cómo documento una serie de tiempo?
- Etiquetas
- Personalizar el portal
- Integrar Andino con Google Analytics
- Consultas sobre Andino
- Otros contenidos útiles

PRIMEROS PASOS

¡Un aplauso, ya estás abriendo tus datos!

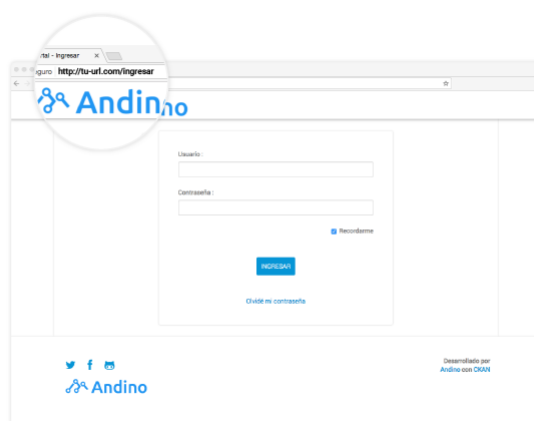
Esta guía te ayudará a:

- Conocer los elementos del catálogo e identificar dónde se ven en el portal.
- Personalizar el portal.
- Crear datasets, recursos, organizaciones y temas.
- Asignar permisos de usuarios.
- Integrar con Google Analytics.

¡Arranquemos!

ENTRAR Y SALIR DE TU PORTAL

Cada vez que quieras entrar al portal, podrás hacerlo desde [http:// tu-url.com /ingresar](http://tu-url.com/ingresar) .



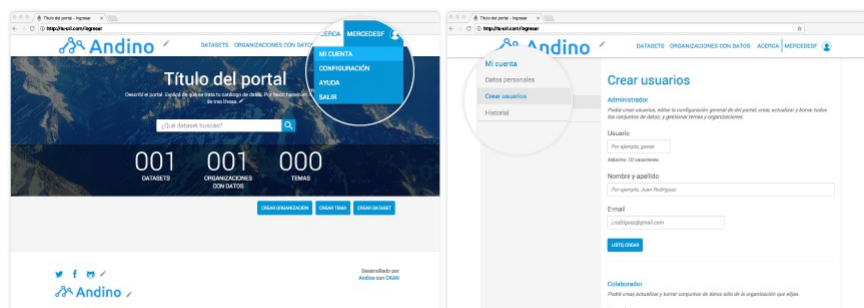
PERMISOS DE USUARIO

Desde agosto de 2017, en Andino **hay dos tipos de usuarios: los administradores y los colaboradores** . El primer usuario administrador siempre es creado por el técnico en sistemas que instaló Andino.

Los administradores de Andino pueden invitar a más personas a colaborar en la apertura de datos, **eligiendo el tipo que quieren asignar** :

- **Administrador** : podrá crear usuarios, editar la configuración general del portal; crear, actualizar y borrar todos los datasets; y gestionar temas y organizaciones.
- **Colaborador** : podrá crear, actualizar y borrar datasets sólo de las organizaciones que tenga asignadas (por eso es importante que primero crees en tu Andino las organizaciones en las que necesitas colaboradores).

Asigná permisos desde Mi cuenta > Crear usuarios .



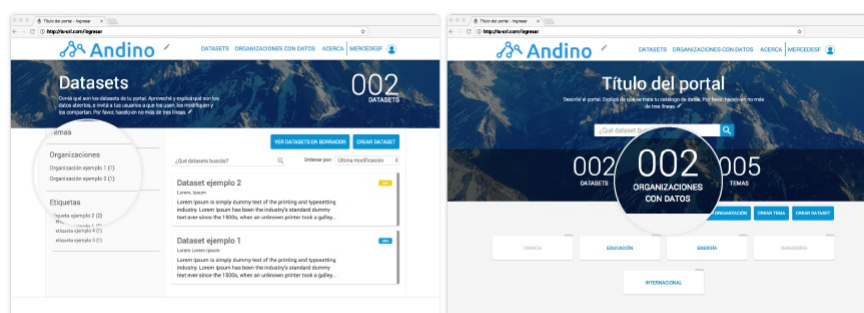
ELEMENTOS DE TU PORTAL

ORGANIZACIONES

Son los organismos que abren o mantienen cada dataset. **Es muy importante que crees las organizaciones antes de que generes un dataset** que esté asociado a ella.

¿Dónde lo veo en el portal?

- Como uno de los filtros de la vista de Datasets.
- Como número agregado, en la Página principal de tu portal, en caso de que hayas elegido la vista que muestra el número de Organizaciones con datos.



Buenas prácticas al crear Organizaciones con datos

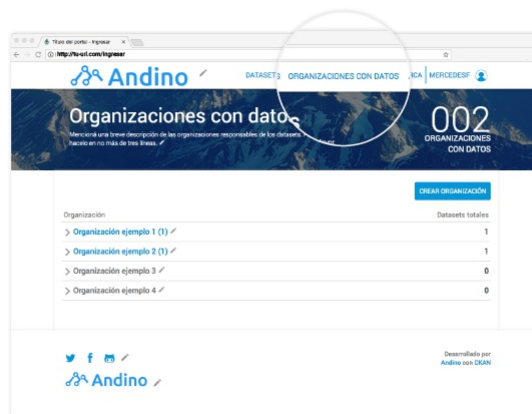
Tené en cuenta que **siempre que borres una organización, todos los conjuntos de datos (y también los recursos), se borrarán definitivamente.**

¿Cómo los creo?

Andá a **Página principal > Crear Organizaciones .**



En caso de que hayas elegido que tu portal tenga una sección de Organizaciones con datos, también se mostrarán allí.



SECCIÓN ORGANIZACIONES CON DATOS

Esta sección es opcional . Te permite armar un árbol de jerarquías con las organizaciones que abrieron datos en tu portal. Los organismos que se muestran en esta parte de tu portal son los mismos que asignás a los datasets y que creás antes de generar estos últimos.

¿Dónde lo veo en el portal?

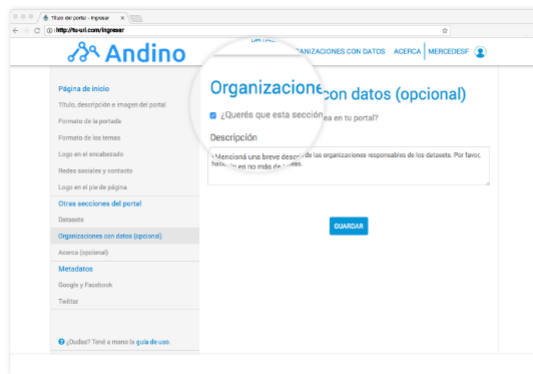
En Página principal > **Organizaciones con datos** .

Buenas prácticas al crear la sección Organizaciones con datos

Es importante que tengas bien en claro qué organizaciones dependen de otras, para que el árbol de jerarquías represente bien las correspondencias.

¿Cómo las creo?

Como la sección Organizaciones con datos es opcional, podés elegir que se vea en tu portal. Para eso, **andá a Configuraciones > Organizaciones con datos** y tildá la habilitación.



SECCIÓN ACERCA

Esta sección es opcional. Te permite darle la oportunidad a los usuarios que naveguen tu portal de leer información del mismo para un mejor entendimiento.

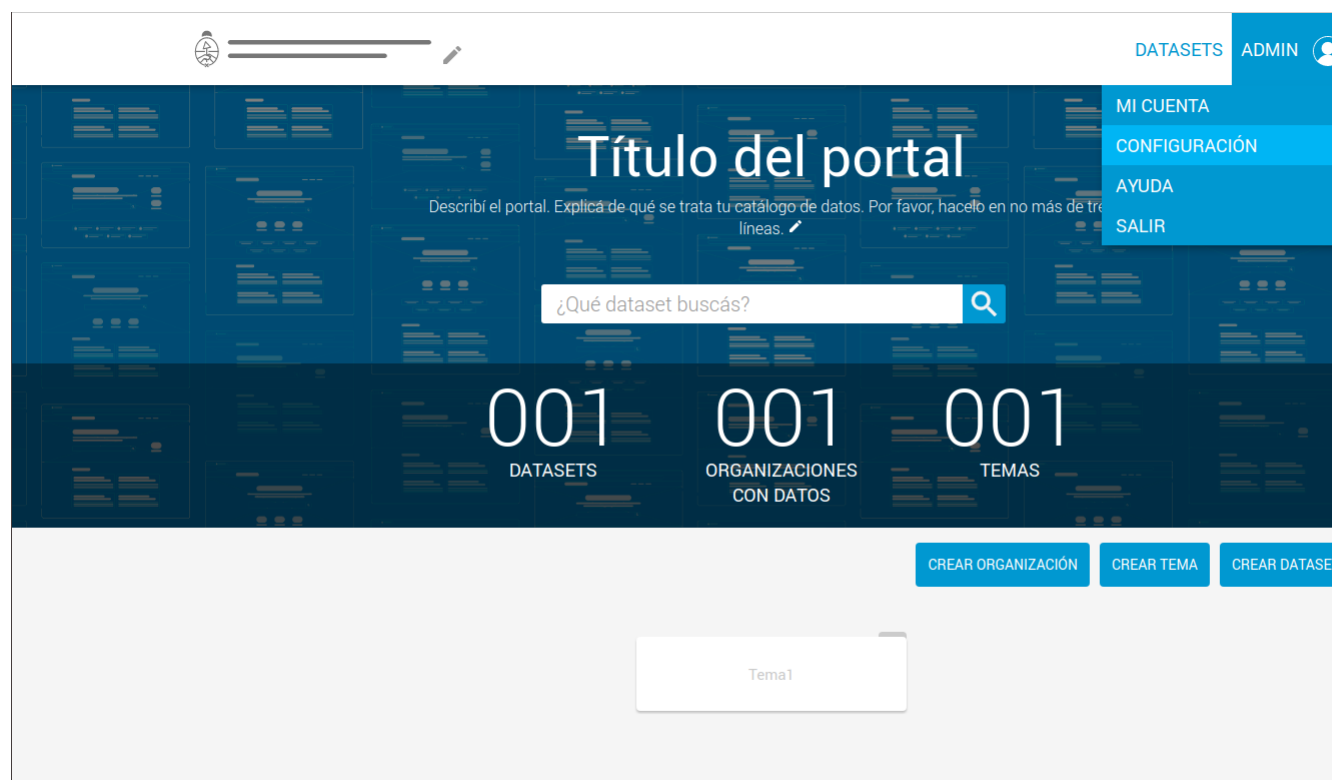
Tipos de Acerca

Existen tres tipos diferentes que vas a poder elegir:

- **Sin sección de acerca** : para cuando se decida no mostrar información.
- **Sección con información básica** : existirá un botón que lleve al usuario a una página donde encontrará información básica sobre el portal (un título y una descripción). Estará ubicado en Página Principal > **Acerca** .
- **Secciones personalizadas** : se crearán varias secciones personalizadas que vas a poder modificar a gusto. El botón Acerca ahora desplegará un menú que contendrá todas y cada una de las secciones que hayas añadido al portal. Para cada sección, se necesitará un archivo (de formato .html) con el contenido que quieras mostrar.

¿Cómo elijo el tipo de Acerca?

Andá a Página Principal > Usuario > Configuración > **Otras secciones del portal > Acerca (opcional)**



<p>Página de inicio</p> <p>Título, descripción e imagen del portal</p> <p>Formato de la portada</p> <p>Formato de los temas</p> <p>Logo en el encabezado</p> <p>Redes sociales y contacto</p> <p>Logo en el pie de página</p> <p>Otras secciones del portal</p> <p>Datasets</p> <p>Organizaciones con datos (opcional)</p> <p>Acerca (opcional)</p> <p>Metadatos</p> <p>Metadatos del portal</p> <p>Google y Facebook</p> <p>Twitter</p>	<h2>Acerca (opcional)</h2> <p>Desde aquí podés seleccionar qué tipo de información querés contar de tu portal. Podés elegir entre brindar una información básica, secciones personalizadas, o elegir no tener esta sección.</p> <p>Seleccioná una opción</p> <div><p>No quiero una sección de acerca</p><p>No quiero una sección de acerca</p><p>Quiero una sección de acerca que dé información básica</p><p>Quiero secciones personalizadas (avanzado)</p></div> <p>GUARDAR CAMBIOS</p>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

¿Cómo puedo escribir y mostrar información básica?

Habiendo elegido la **segunda opción**, podés modificar el título y la descripción.

<p>Página de inicio</p> <p>Título, descripción e imagen del portal</p> <p>Formato de la portada</p> <p>Formato de los temas</p> <p>Logo en el encabezado</p> <p>Redes sociales y contacto</p> <p>Logo en el pie de página</p> <p>Otras secciones del portal</p> <p>Datasets</p> <p>Organizaciones con datos (opcional)</p> <p>Acerca (opcional)</p> <p>Metadatos</p> <p>Metadatos del portal</p> <p>Google y Facebook</p> <p>Twitter</p> <p>¿Dudas? Tené a mano la guía de uso.</p>	<h2>Acerca (opcional)</h2> <p>Desde aquí podés seleccionar qué tipo de información querés contar de tu portal. Podés elegir entre brindar una información básica, secciones personalizadas, o elegir no tener esta sección.</p> <p>Seleccioná una opción</p> <div>Quiero una sección de acerca que dé información básica</div> <p>Información básica</p> <p>Contá qué son los datasets de tu portal. Aprovechá y explicá qué son los datos abiertos, e invitá a tus usuarios a que los usen, los modifiquen y los compartan.</p> <p>Título</p> <div>Acerca de este portal</div> <p>Descripción</p> <div>Contá qué son los datasets de tu portal. Aprovechá y explicá qué son los datos abiertos, e invitá a tus usuarios a que los usen, los modifiquen y los compartan. Por favor, hacelo en no más de tres líneas.</div> <p>En este campo podés usar formato Markdown.</p> <p>GUARDAR CAMBIOS</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

¿Cómo puedo crear y mostrar mis secciones personalizadas?

Habiendo elegido la **tercera opción**, podrás ir creando tus secciones escribiendo un título y un nombre de archivo. Recordá que, para cada sección, será necesario guardar un archivo (de formato .html) para que el portal pueda mostrar su contenido.

<p>Página de inicio</p> <p>Título, descripción e imagen del portal</p> <p>Formato de la portada</p> <p>Formato de los temas</p> <p>Logo en el encabezado</p> <p>Redes sociales y contacto</p> <p>Logo en el pie de página</p>	<h2>Acerca (opcional)</h2> <p>Desde aquí podés seleccionar qué tipo de información querés contar de tu portal. Podés elegir entre brindar una información básica, secciones personalizadas, o elegir no tener esta sección.</p> <p>Seleccioná una opción</p> <p>Quiero secciones personalizadas (avanzado)</p> <p>Secciones personalizadas</p> <p>Para agregar una nueva sección, podés crear archivos .html en la carpeta "custom". Te sugerimos guiarte viendo el archivo "seccion-acerca.html". Lo único que tenés que completar aquí es el nombre de ese archivo y qué título llevará la sección.</p> <p>Sección 1</p> <p>Título</p> <p>Sección 1</p> <p>Nombre del archivo</p> <p>seccion1.html</p> <p>+ Agregar nueva sección</p>
<p>Otras secciones del portal</p> <p>Datasets</p> <p>Organizaciones con datos (opcional)</p> <p>Acerca (opcional)</p> <p>Metadatos</p> <p>Metadatos del portal</p> <p>Google y Facebook</p> <p>Twitter</p>	

¿Dudas? Tené a mano la [guía de uso](#).

¿Dónde añado los archivos requeridos para las secciones?

Dentro del container, dichos archivos deben ser guardados en el directorio **/var/lib/ckan/theme_config/templates/**.

De no existir la carpeta **/templates**, se deberá crearla utilizando dos comandos:

- `mkdir /var/lib/ckan/theme_config/templates`
- `chown www-data:www-data /var/lib/ckan/theme_config/templates/`

(www-data es el usuario con el que se corre el proceso apache en el sistema operativo **Ubuntu**)

Dentro de cada archivo deberá estar el contenido de su sección correspondiente. Por favor, **no te olvides** de que el nombre del archivo **debe coincidir** con lo escrito en el campo 'Nombre del archivo' de la sección.

¿Dónde y cómo puedo guardar imágenes para mostrarlas en mis secciones? ¿Cómo las muestro en mis templates?

Dentro del container, las imágenes deben ser guardadas en una carpeta llamada "user_images".

Para poder copiar y pegar una imagen en dicha carpeta existe un comando que podés usar, pero primero tenés que hacer tres cosas:

- Dentro del contenedor, ejecutar este comando:

```
chown www-data:www-data /usr/lib/ckan/default/src/ckanext-gobar-theme/ckanext/gobar_theme/public/user_images/
```

- En caso de que hayas cambiado manualmente el puerto del contenedor, saber cuál es (el default es 8080)
- Tener preparado el nombre del contenedor (ya que queremos decirle al comando en qué lugar vamos a guardar la imagen)

El nombre del contenedor se consigue escribiendo en una terminal dentro del host:

```
sudo docker-compose -f latest.yml ps |grep <puerto del container>
```

Sólo nos queda guardar la imagen:

```
docker cp <nombre de la imagen> <nombre del contenedor portal>:/usr/lib/ckan/default/src/ckanext-gobar-theme/ckanext/gobar_theme/public/user_images/
```

Supongamos que guardaste una imagen llamada 'mi_imagen.png'; para poder utilizarla, podés escribir dentro de tu template esta línea (completa):

```

```

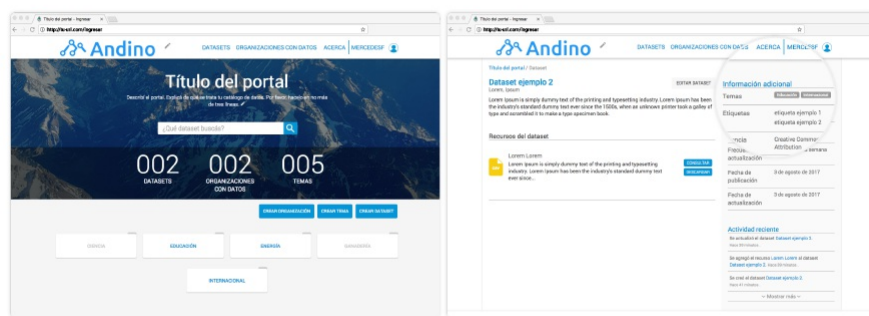
TEMAS

Son las categorías en las que se pueden clasificar todos los datasets de tu portal. Hay dos taxonomías de temas:

- **Temas globales**, que ya vienen con Andino, y que necesitás elegir para cada dataset. Estos temas no se ven en tu portal, pero es necesario que lo elijas para que el portal nacional datos.gob.ar pueda republicar el dataset según esta clasificación. Por ejemplo: "Economía y finanzas".
- **Temas específicos**, que son opcionales, pero que te recomendamos con énfasis que agregues a todos tus conjuntos de datos porque son los temas que van a ver tus usuarios. Por ejemplo, si el tema global era "Economía y finanzas", un tema específico podría ser "Compras".

¿Dónde lo veo en el portal?

- En la Página principal de tu catálogo de datos.
- También como detalle de cada dataset específico.

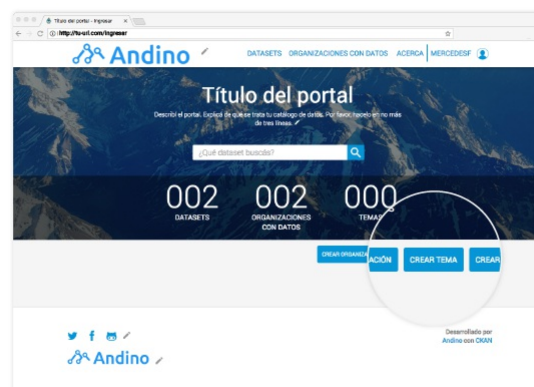


Buenas prácticas al crear Temas específicos

- Intentá crear los temas específicos a conciencia para poder reutilizarlos a futuro.
- No crees un tema específico por cada dataset.
- Asegurate de que cada tema específico sea un aparte más pequeña dentro de los Temas globales.

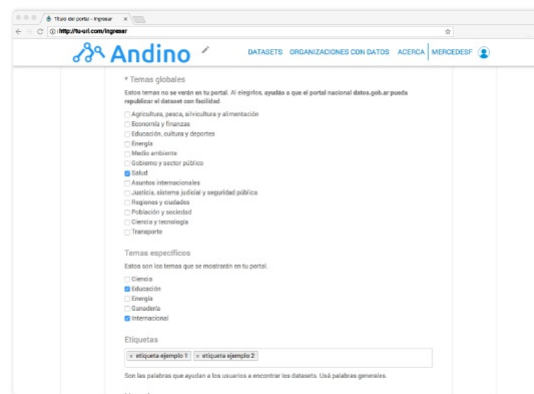
¿Cómo los creo?

Andá a **Página principal > Crear tema**.



¿Cómo los asigno?

Cada vez que generes un nuevo dataset, el formulario te pedirá que asignes temas. Recordá siempre reutilizar los que ya hayas creado y no repetirlos.

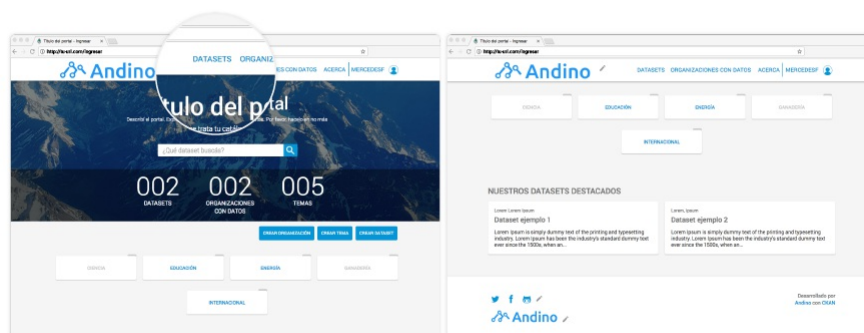


DATASETS

También los llamamos "conjuntos de datos". Son la pieza principal de tu portal o catálogo de datos. Cada dataset está formado por uno o más recursos.

¿Dónde los veo en el portal?

Todos los datasets que subas al portal se verán **en la sección Datasets**. Además, **podrás destacar** los que creas más importantes en la Página principal.

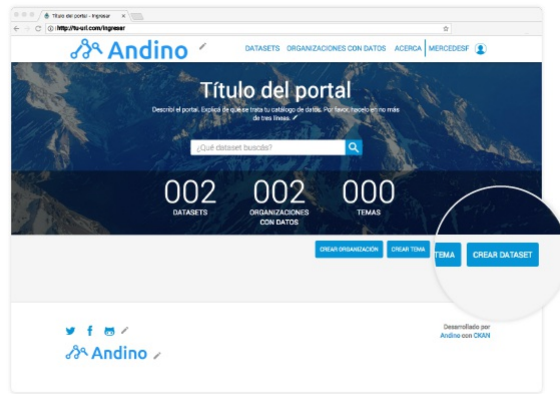


Buenas prácticas al crear datasets

- **Títulos de los datasets** : es el primer vistazo que los usuarios tendrán sobre su contenido. Por eso, intentá no superar los 100 caracteres. Prestá especial atención a las mayúsculas. Sólo los sustantivos propios las necesitan.
- **Descripción de los datasets** : es el detalle que le contás a los usuarios. Por esa razón, es importante que trates de dar una explicación general de los datos con los que se va a encontrar. Intentá no superar los 500 caracteres.

¿Cómo los creo?

Ingresá a tu cuenta y andá a **Página principal > Crear dataset**.



Siempre podrás editar los dataset que hayas creado. Para eso, ingresá a tu cuenta > Página principal > Datasets > **Editar dataset**.

RECURSOS

Cada dataset está formado por, al menos, un recurso. Por eso decimos que los recursos son la pieza de información más pequeña del catálogo y los verdaderos activos de datos del portal.

¿Dónde los veo en el portal?

Página principal > Datasets > Clic en el recurso específico.

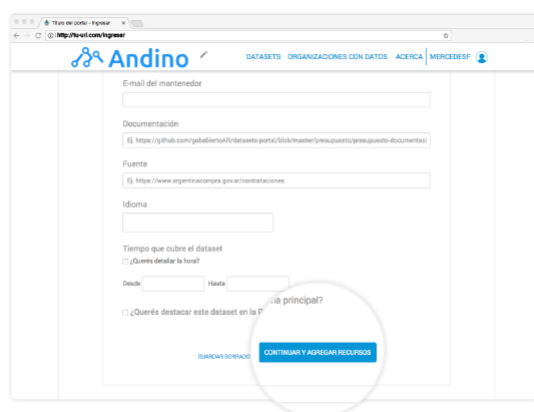
Buenas prácticas al crear recursos

Seguí los mismos criterios de escritura que con los datasets.

- **Títulos de los recursos** : intentá no superar los 150 caracteres.
- **Descripción de los recursos** : intentá no superar los 200 caracteres.

¿Cómo los creo?

Ingresá a tu cuenta y andá a Página principal > Crear dataset. Una vez que completes el dataset, podrás agregar recursos.




Al igual que con los datasets, siempre podrás editar los recursos que hayas creado. Para eso, ingresá a tu cuenta > Página principal > Datasets > Editar dataset > Editar recursos.

CAMPOS DE UN RECURSO

Los recursos pueden tener mucha información y ser difíciles de comprender. Para ayudar a los usuarios que ven nuestros recursos, Andino permite documentar qué contiene cada campo.

¿Dónde los veo en el portal?

Los recursos que tengan información sobre sus campos se visualizarán en la vista de un recurso.







Andino

[DATASETS](#)
[ORGANIZACIONES CON DATOS](#)
[ACERCA](#)
[MÉTCES/DEP](#)

Campos de este recurso

Nombre de la columna	Unidad de medida	Identificador	Tipo de dato	Detalle de datos
id_indice, campo	date	1_UFJA,1983_A,13	date_index	SUPPLY
afirma_global_pib	PIB desestacionalizado, en millones de pesos de 1980 y valores trimestrales	1_1983_A,1983_A,17		
afirma_global_importes	Exportaciones desestacionalizadas, en millones de pesos de 1980 y valores trimestrales	1_1983_A,1983_A,26		
afirma_global_mpr	Exportaciones desestacionalizadas, en millones de pesos de 1980 y valores trimestrales	1_1983_A,1983_A,28		
demanda_global_3d	Interacción bruta interna PIB desestacionalizado, en millones de pesos de 1980 y valores trimestrales	1_1983_A,1983_A,19		
demanda_global_crea_cmu200	Consumo privado desestacionalizado, en millones de pesos de 1980 y valores trimestrales	1_1983_A,1983_A,27		
demanda_global_crea_cmu_publico	Consumo público desestacionalizado, en millones de pesos de 1980 y valores trimestrales	1_1983_A,1983_A,31		

Información adicional

Descargado por Andino con OXAN

Buenas prácticas al cargar campos de un recurso

Puede ser de mucha ayuda para los usuarios que siempre cargues al menos los siguientes campos: - Título de la columna - Tipo de dato - Descripción

También podés usar estos campos para documentar aspectos más avanzados de las columnas de un recurso: - Unidad de medida - Identificador - Tipo de dato especial - Detalle de tipo especial

SERIES DE TIEMPO

Andino permite documentar recursos que publican **series de tiempo** . Un recurso con series de tiempo es un archivo CSV de determinada estructura .

¿Cómo documento una serie de tiempo?

En el formulario de carga de un recurso vas a encontrar campos avanzados y especiales que necesitás usar para documentar las columnas de una tabla que contiene series de tiempo.

Los campos **Tipo de dato especial** y **Detalle de tipo especial** te permiten marcar la columna del índice de tiempo y qué frecuencia tiene.

- **Tipo de dato especial** : marcar la columna que tiene las fechas como "Índice de tiempo".
- **Detalle de tipo de dato especial** : especificar la frecuencia del índice (anual, diaria, mensual, etc).

Indice de tiempo

Tipo de dato

Fecha ISO 8601 (Date)

Por favor, no superes los 40 caracteres.

Indice de tiempo

Campos especiales

Desde aquí puedes agregar un campo especial (como un índice de tiempo) y su detalle.

Estos campos son opcionales.

Tipo de dato especial

Detalle del tipo de dato especial

Indice de tiempo

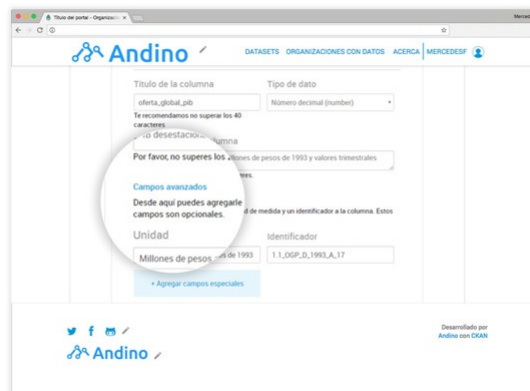
Trimestralmente

• Agregar campos avanzados

Desarrollado por Andino con D2AN

Los campos **Unidad de medida** e **Identificador** te permiten documentar cada una de las columnas, que son tus series de tiempo.

- **Unidad de medida** : es la unidad de medida en que está expresada la serie ("Millones de pesos corrientes", "Kilómetros", "Millones de dólares americanos")
- **Identificador** : es un identificador único de la serie para toda la APN. Debe ser inmutable en el tiempo para la serie, no muy largo (entre 6 y 20 caracteres aproximadamente) y no pisarse potencialmente con otras series de la Administración Pública Nacional.



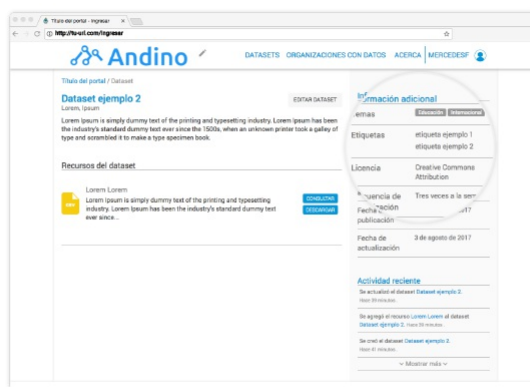
Para conocer más sobre series de tiempo mirá la documentación completa en el perfil de metadatos.

ETIQUETAS

Son las palabras que ayudan a los usuarios a filtrar y encontrar los datasets. Cuanto más amplia y uniforme sea la lista de Etiquetas, mayor es su poder de ayuda.

¿Dónde las veo en el portal?

En cada dataset específico.



Buenas prácticas al crear etiquetas

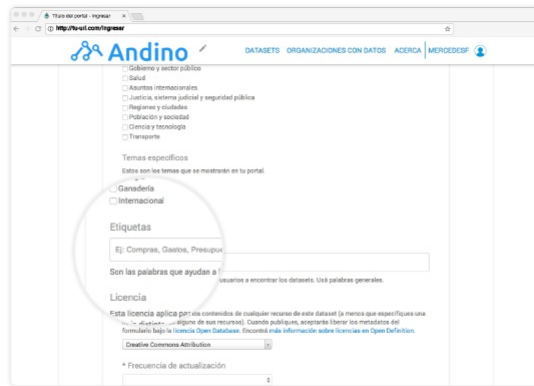
- Usá mayúsculas sólo donde corresponda.
- Identificá palabras claves.
- Respetá las etiquetas anteriores.
- Agregá sinónimos y usá lenguaje natural.
- Tratá de usar una sola palabra, siempre en plural.
- Si la etiqueta tiene más de una palabra, separalas por un espacio. Por ejemplo: "declaraciones juradas".

Éstas son algunas preguntas útiles a la hora de pensar las etiquetas:

- ¿Cuál es el tema?
- ¿Qué aspectos serán de interés para los usuarios?
- ¿De qué otro modo buscaría el usuario esta información?
- ¿De qué tipo de información se trata?
- ¿Qué área la provee?

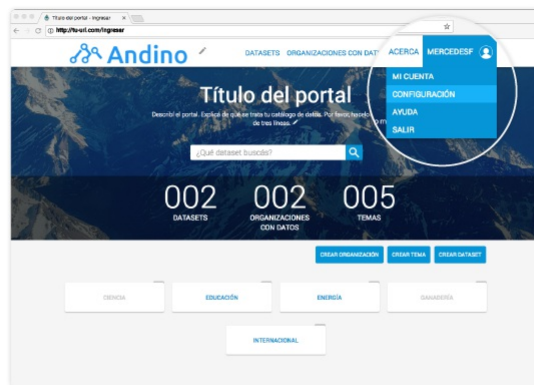
¿Cómo las creo?

Al igual que con los Temas, cada vez que generes un nuevo dataset, el formulario te pedirá que asignes un etiqueta. Recordá siempre reutilizar las que ya hayas creado y no repetirlas.



PERSONALIZAR EL PORTAL

Hacer que tu portal represente tu organización es muy fácil. Por favor, **andá a la Página principal > Configuración** . Llegarás a la sección que te permite cambiar cómo luce la portada de tu catálogo de datos y cada sección en particular.



INTEGRAR ANDINO CON GOOGLE ANALYTICS

Por favor, escribinos y contanos con qué casilla de e-mail querés tener permisos para ver las métricas de tu Andino.

CONSULTAS SOBRE ANDINO

Andino es un portal abierto en constante desarrollo para ser usado por toda la comunidad de datos. Por eso, cuando incorporamos una nueva mejora, **cuidamos mucho su compatibilidad con la versión anterior** .

Como la comunidad de datos es grande, **por ahora no podemos dar soporte técnico frente a modificaciones particulares del código** . Sin embargo, **podés contactarnos para despejar dudas** .

Te invitamos a crear issues o enviarnos sugerencias en caso de que encuentren algún *bug* o *tengas feedback* . También podés mandarnos tu comentario o consulta a datos@modernizacion.gob.ar

OTROS CONTENIDOS ÚTILES

- Guía para el uso y la publicación de metadatos
- Guía para la identificación y uso de entidades interoperables
- Guía para la publicación de datos en formatos abiertos

INDICE

- Instalación
- Dependencias
- Instalación simplificada de andino
- Instalación avanzada de andino
- Desinstalar andino

INSTALACIÓN

Teniendo en cuenta la dificultad de implementación e incluso la cantidad de pasos para lograr un deploy exitoso, existen dos formas de instalar esta distribución de **CKAN**.

- Si no tenés muchos conocimientos de CKAN, Docker o de administración de servidores en general, es recomendable usar la instalación simplificada de Andino. Está pensada para que, en la menor cantidad de pasos y de manera sencilla, tengas un portal de datos funcionando.
- Si ya conocés la plataforma, tenés experiencia con Docker o, simplemente, querés entender cómo funciona esta implementación, te sugiero que revises la instalación avanzada de Andino

DEPENDENCIAS

- DOCKER: Guía de instalación .
- Versión mínima *testada* : 1.13.1
- Docker Compose: Guía de instalación .
- Versión mínima *testada* : 1.12.0

INSTALACIÓN SIMPLIFICADA DE ANDINO

La idea detrás de esta implementación de CKAN es **que sólo te encargues de tus datos**, nada más. Por eso, si "copiás y pegás" el comando de consola, en sólo unos momentos, tendrás un Andino listo para usar. Esta clase de instalación no requiere que clones el repositorio, ya que usamos contenedores alojados en DockerHub

- Ubuntu|Debian|RHCL|CentOS:
- Instalación:

Para la instalación, usamos un script de python llamado `install.py`. El mismo requiere algunos parámetros específicos, y existen otros que son opcionales:

```
# Parametros de install.py
[-h]                               Mostrar la ayuda del script
--error_email                      Email donde se mandaran los errores del por
tal de ser necesario
--site_host                        Dominio o IP del la aplicación *sin el prot
ocolo*
--database_user                   Nombre del usuario de la base de datos a cr
ear
--database_password              Contraseña de la base de datos a crear
--datastore_user                 Nombre del usuario de la base de datos del
datastore a crear
--datastore_password             Contraseña de la base de datos del datastor
e a crear
[--nginx-extended-cache]
    Configura nginx con una configuración extendida y configura el
hook de
    invalidación de cache de Andino para notificar a nginx
[--nginx_ssl]
    Aplica la configuración HTTPS en nginx. Requiere ambos archivos
```

del certificado SSL para poder lograrlo; en caso contrario, se utilizará la configuración default

```
[--ssl_key_path SSL_KEY_PATH]
```

Path dentro del host donde está ubicado el archivo .key para el certificado SSL; será copiado al contenedor de nginx si tanto éste como el .crt pueden ser encontrados

```
[--ssl_crt_path SSL_CRT_PATH]
```

Path dentro del host donde está ubicado el archivo .crt para el certificado SSL; será copiado al contenedor de nginx si tanto éste como el .key pueden ser encontrados

```
[--nginx_port NGINX_PORT]
```

Puerto del servidor "Host" que se desea que se tome para recibir llamadas HTTP.

Por defecto es el 80.

```
[--nginx_ssl_port NGINX_SSL_PORT]
```

Puerto del servidor "Host" que se desea que se tome para recibir llamadas HTTPS.

Por defecto es el 443.

Es importante para los administradores saber que Andino tomará el puerto especificado (o el default) ya sea que el portal use o no use HTTPS. En caso de no querer usar HTTPS y que el host tenga el puerto 443 tomado por un servidor web, es requisito especificar un puerto distinto (ejemplo: 8443) que será reservado por Andino, pero no utilizado.

```
[--datastore_port DATASTORE_PORT]
```

Puerto del servidor "Host" que se desea que se tome para recibir llamadas HTTP al "datastore".

Por defecto es el 8800.

```
[--install_directory INSTALL_DIRECTORY]
```

Directorio donde se desea instalar la aplicación.

Por defecto es `/etc/portal` (recomendado)

Para esta instalación de ejemplo, usaremos estos parámetros para la aplicación. Para los demás, usaremos los valores por defecto:

- Email donde se mandarían los errores: EMAIL=admin@example.com
- Dominio o IP de la aplicación *sin el protocolo*: HOST=datos.gob.ar
- Usuario de la base de datos: DB_USER=<my db user>
- Password de la base de datos: DB_PASS=<my db pass>
- Usuario del datastore: STORE_USER=<my datastore user>
- Password del datastore: STORE_PASS=<my datastore password>

NOTA: Si usamos una IP para la variable HOST, el envío de mails no funcionará. Postfix requiere un "fully-qualified domain name (FQDN)". Ver la documentación de Postfix para más detalles.

NOTA 2: Si utilizamos el nombre 'localhost' para la variable HOST, es posible que ocurra un error al intentar subir un archivo perteneciente a un recurso al Datastore: Error: Proceso completo pero no se pudo enviar a result_url.

```
# Primero especificamos los valores necesarios
```

```
EMAIL=admin@example.com
HOST=andino.midominio.com.ar
DB_USER=my_database_user
DB_PASS=my_database_pass
STORE_USER=my_data_user
STORE_PASS=my_data_pass
```

```
wget https://raw.githubusercontent.com/datosgobar/portal-andino/master/install/install.py
```

```
sudo python ./install.py \
  --error_email "$EMAIL" \
  --site_host="$HOST" \
  --database_user="$DB_USER" \
```

```
--database_password="$DB_PASS" \
--datastore_user="$STORE_USER" \
--datastore_password="$STORE_PASS"
```

INSTALACIÓN AVANZADA DE ANDINO

La instalación avanzada está pensada para usuarios que quieren ver cómo funciona internamente Andino

Para instalar y ejecutar Andino, seguiremos estos pasos:

- Paso 1: Clonar repositorio.

```
sudo mkdir /etc/portal
cd /etc/portal
sudo git clone https://github.com/datosgobar/portal-andino.git .
```

- Paso 2: Especificar las variables de entorno para el contenedor de postgresql.

NOTA: Debemos usar un dominio válido para la variable `DOMINIO`, de otra forma el envío de mails no funcionará. Postfix requiere un "fully-qualified domain name (FQDN)". Ver la documentación de Postfix para más detalles.

```
DB_USER=<my user>
DB_PASS=<my pass>
DOMINIO=andino.midominio.com.ar
ANDINO_VERSION=<version que deseamos instalar>
sudo su -c "echo POSTGRES_USER=$DB_USER > .env"
sudo su -c "echo POSTGRES_PASSWORD=$DB_PASS >> .env"
sudo su -c "echo NGINX_HOST_PORT=80 >> .env"
sudo su -c "echo DATASTORE_HOST_PORT=8800 >> .env"
sudo su -c "echo maildomain=$DOMINIO >> .env"
sudo su -c "echo ANDINO_TAG=$ANDINO_VERSION >> .env"
```

- Paso 3: Construir y lanzar los contenedor de servicios usando el archivo **latest.yml**: `docker-compose -f latest.yml up -d db postfix redis solr`
- Paso 4: Construir y lanzar el contenedor de **andino** usando el archivo **latest.yml**: `docker-compose -f latest.yml up -d portal`
- Paso 5: Inicializar la base de datos y la configuración de la aplicación:

```
EMAIL=admin@example.com
HOST=datos.gob.ar
DB_USER=<my db user>
DB_PASS=<my db pass>
STORE_USER=<my datastore user>
STORE_PASS=<my datastore password>
docker-compose -f latest.yml exec portal /etc/ckan_init.d/init.sh -e "$EMAIL" -h "$HOST" \
    -p "$DB_USER" -P "$DB_PASS" \
    -d "$STORE_USER" -D "$STORE_PASS"
```

- Paso 8: Construir el contenedor de **nginx** usando el archivo **latest.yml**: `docker-compose -f latest.yml up -d nginx`

DESINSTALAR ANDINO

Esta secuencia de comandos va a ELIMINAR TODOS LOS CONTENEDORES, IMÁGENES y VOLUMENES de la aplicación de la vm donde está instalada la plataforma.

Esta operación no es reversible. **Perderás todos tus datos si realizas esta operación** .

```
app_dir="/etc/portal/"
cd $app_dir
docker-compose -f latest.yml down -v
cd ~/
sudo rm $app_dir -r
```


INDICE

- Actualización
- Versiones 2.x
 - Actualización simple
 - Actualización avanzada
 - Andino con plugins ad-hoc
 - Versiones 2.5.5 en adelante
 - Versiones 2.5.0 y 2.5.1
 - Versiones entre 2.4.0 y 2.5.3
 - Versiones 2.4.x a 2.5.x
- Versiones 1.x a 2.x

ACTUALIZACIÓN

VERSIONES 2.X

Nota: si actualizás desde una versión 2.4.x a una versión 2.5.x ver Versiones 2.4.x a 2.5.x ANTES de proceder con la actualización

ACTUALIZACIÓN SIMPLE

Si instalamos la aplicación con la última versión del instalador (este), el mismo no requerirá parámetros, pero contiene algunos opcionales:

```
# Parámetros de update.py
[-h]                Mostrar la ayuda del script
[--install_directory INSTALL_DIRECTORY]
                    Directorio donde está instalada la aplicación.
                    Por defecto es `/etc/portal`
[--nginx-extended-cache]
                    Configura nginx con una configuración extendida y configura el
hook de
                    invalidación de cache de Andino para notificar a nginx
[--nginx_ssl]
                    Aplica la configuración HTTPS en nginx. Requiere ambos archivos
del certificado SSL para poder lograrlo; en
                    caso contrario, se utilizará la configuración default
[--ssl_key_path SSL_KEY_PATH]
                    Path dentro del host donde está ubicado el archivo .key para el
certificado SSL; será copiado al contenedor
                    de nginx si tanto éste como el .crt pueden ser encontrados
[--ssl crt_path SSL_CRT_PATH]
                    Path dentro del host donde está ubicado el archivo .crt para el
certificado SSL; será copiado al contenedor
                    de nginx si tanto éste como el .key pueden ser encontrados
[--nginx_port NGINX_PORT]
                    Puerto del servidor "Host" que se desea que se tome para recibi
r llamadas HTTP.
                    Por defecto es el 80.
[--nginx_ssl_port NGINX_SSL_PORT]
                    Puerto del servidor "Host" que se desea que se tome para recibi
r llamadas HTTPS.
                    Por defecto es el 443.
                    Es importante para los administradores saber que Andino tomará
el puerto especificado (o el default) ya sea que el portal use o no use
                    HTTPS. En caso de no querer usar HTTPS y que el host tenga el puerto
```

443 tomado por un servidor web, es requisito especificar un puerto distinto (ejemplo: 8443) que será reservado por Andino, pero no utilizado.

Para esta actualización de ejemplo, usaremos los valores por defecto:

```
sudo wget https://raw.githubusercontent.com/datosgobar/portal-andino/master/install/update.py
sudo python update.py
```

De esta forma, el script asumirá que instalamos la aplicación en `/etc/portal`.

ACTUALIZACIÓN AVANZADA

Si instalamos la aplicación en otro directorio distinto de `/etc/portal`, necesitamos correr el script de una manera diferente. Suponiendo que instalamos la aplicación en `/home/user/app/`, debemos correr los siguientes pasos:

```
wget https://raw.githubusercontent.com/datosgobar/portal-andino/master/install/update.py
sudo python update.py --install_directory="/home/user/app/"
```

ANDINO CON PLUGINS AD-HOC

Si configuraste tu instancia de Andino con algún plugin de CKAN que Andino no trae por defecto, es importante que antes de la instalación elimines los mismos de la opción de configuración `ckan.plugins` del archivo `/etc/ckan/default/production.ini` del contenedor `portal`. Esto es importante, ya que el proceso de actualización descarga imágenes de Docker nuevas que no contendrán los binarios de los plugins *ad-hoc* y si los mismos están en el archivo de configuración de CKAN, la actualización fallará.

Los pasos adicionales que deberás seguir si tenés plugins *ad-hoc* son:

1. Editar el archivo `/etc/ckan/default/production.ini` del contenedor `portal` y quitar de la lista de `ckan.plugins` los plugins *ad-hoc*.
2. Actualizar Andino.
3. Instalar los plugins *ad-hoc* dentro del *virtualenv* `/usr/lib/ckan/default` del contenedor `portal`.
4. Editar el archivo `/etc/ckan/default/production.ini` del contenedor `portal` y agregar a la lista de `ckan.plugins` los plugins *ad-hoc*.
5. Reiniciar Andino.

VERSIONES 2.5.5 EN ADELANTE

Debido a la posibilidad de que ocasionen problemas durante la instalación, se removieron los plugin `harvest` y `datajson` del archivo de configuración, y se agregó una migración para eliminarlos al actualizar Andino para evitar posibles problemas.

De ser necesaria la utilización de los plugins mencionados, deberán ser añadidos manualmente una vez finalizada la actualización.

VERSIONES 2.5.0 Y 2.5.1

Si actualizás de 2.5.0 o 2.5.1 a 2.5.2 o una versión más nueva, hay que modificar el archivo de configuración para que `googleanalytics` esté *sólo una vez y al final* en `ckan.plugins`.

Para ver cómo modificar el archivo de configuración, ir a la documentación de mantenimiento.

Ejemplo de cómo podría quedar: `ckan.plugins = datajson_harvest datajson harvest ckan_harvester stats text_view image_view recline_view hierarchy_display hierarchy_form dcat structured_data gobar_theme datastore datapusher seriestiempoarexplorer googleanalytics`

VERSIONES ENTRE 2.4.0 Y 2.5.3

Al actualizar un portal cuya versión se encuentra entre 2.4.0 y 2.5.3 a una versión más nueva, existe un comando que se debe ejecutar debido a problemas con el guardado de archivos de recursos (no se puede descargar un archivo de recurso si éste fue editado sin que se actualizara el archivo).

Este comando recuperará los archivos de recursos para los cuales se cumplan estas condiciones: - El recurso es local - El recurso posee el campo `downloadURL` en el `data.json` - El archivo del recurso existe en el Datastore (su extensión debe ser `csv`, `x/s` o `x/sx`) y no está vacío

Existen 3 métodos para ejecutar la actualización de los recursos:

1. Actualizar sólo aquellos recursos cuyos archivos **no** sean descargables y cumplan con las condiciones detalladas más arriba
2. Actualizar todos los recursos que simplemente cumplan con las condiciones detalladas más arriba
3. Especificar uno o más IDs de los recursos que se quieran actualizar (en vez de modificar todos los posibles)

Nota: Los métodos 2) y 3) son combinables.

Para poder implementar la solución una vez hecha la actualización del portal, se debe ejecutar los siguientes comandos:

```
docker-compose -f latest.yml exec portal bash
cd /usr/lib/ckan/default/src/ckanext-gobar-theme
/usr/lib/ckan/default/bin/paster --plugin=ckan reupload-resources-files
--config=/etc/ckan/default/production.ini
exit
```

Para cada método mencionado, la tercera línea a ejecutar (el comando de actualización de recursos) será distinta:

1. Dejar el comando tal y como está, ya que es el comportamiento default
2. Escribir después del texto `reupload-resources-files` el flag `--force=true`
3. Escribir después del texto `reupload-resources-files` (o del flag `--force=true` si se lo utilizó) todos los IDs de los recursos a actualizar

Ejemplo de cómo quedaría si se quiere utilizar los métodos 2) y 3) actualizando dos recursos distintos:

```
/usr/lib/ckan/default/bin/paster --plugin=ckan reupload-resources-files --
force=true a57e4006-9e15-4bc7-b46a-25bf3580e538 t5t4rp09-156s-xzq2-36vl-
2d5e8fghn98q --config=/etc/ckan/default/production.ini
```

VERSIONES 2.4.X A 2.5.X

En el caso de actualizar un Andino de versión 2.4.x a 2.5.x existe un error conocido de CKAN 2.5.8 (Ver issue [ckan/ckan#4168](#)) que **debe solucionarse ANTES de ejecutar la actualización**.

En el procedimiento normal, ocurriría un error: `sqlalchemy.exc.ProgrammingError: (ProgrammingError) column package.metadata_created does not exist`

Para poder solucionarlo, se debe correr el siguiente script **antes de ejecutar el procedimiento normal de actualización**:

```
docker-compose -f latest.yml exec -u postgres db psql -c "
do \$$
begin
  IF NOT EXISTS(SELECT * FROM information_schema.columns WHERE table_name='package' AND column_name='metadata_created') OR
    NOT EXISTS(SELECT * FROM information_schema.columns WHERE table_name='package_revision' AND column_name='metadata_created') THEN

    IF NOT EXISTS(SELECT * FROM information_schema.columns WHERE table_name='package_revision' AND column_name='metadata_created') THEN
      ALTER TABLE package_revision ADD COLUMN metadata_created timestamp without time zone;
    END IF;

    IF NOT EXISTS(SELECT * FROM information_schema.columns WHERE table_name='package' AND column_name='metadata_created') THEN
      ALTER TABLE package ADD COLUMN metadata_created timestamp without time zone;
```

```

END IF;

UPDATE package SET metadata_created=
    (SELECT revision_timestamp
     FROM package_revision
     WHERE id=package.id
     ORDER BY revision_timestamp ASC
     LIMIT 1);

END IF;
end \$$
" ckan

```

VERSIONES 1.X A 2.X

Ver documento de migración

CHECKLIST PARA LA PUESTA EN PRODUCCIÓN

INDICE

- Recomendaciones para la puesta en producción de una instancia de Andino
- Verificá que tu instancia de Andino tenga un nombre de dominio único y bien configurado
 - Verificar si mi Andino tiene el nombre de dominio configurado correctamente
 - Actualizando el nombre de dominio asignado a Andino
- Verificá que el contenedor portal pueda resolver el nombre de dominio asignado a la instancia
- Seguir las recomendaciones de seguridad

RECOMENDACIONES PARA LA PUESTA EN PRODUCCIÓN DE UNA INSTANCIA DE ANDINO

Si estás por configurar tu instancia de Andino para un ambiente productivo, tenemos algunas recomendaciones para que sigas y verifiques si tu instancia está bien configurada.

VERIFICÁ QUE TU INSTANCIA DE ANDINO TENGA UN NOMBRE DE DOMINIO ÚNICO Y BIEN CONFIGURADO

Para el correcto funcionamiento de tu instancia de Andino, es recomendable que la misma tenga un único nombre de dominio asignado.

Una vez que tengas definido el nombre de dominio (ej: `datos.ministerio.gob.ar`) y el mismo se resuelva a la IP pública asignado al *host* de tu Andino (o al *load balancer* / *frontend server* que opcionalmente tengas) es importante que tu instancia de Andino conozca ese nombre de dominio y que esté configurado para responder al mismo.

Verificar si mi Andino tiene el nombre de dominio configurado correctamente

Para saber si tu instancia de Andino tiene el nombre de dominio correctamente configurado seguí los siguientes pasos, ejecutando el comando en el directorio de instalación de Andino (ej: `/etc/portal`):

```
docker-compose -f latest.yml exec portal grep ckan\.site_url /etc/ckan/default/production.ini
```

Si tu sitio está bien configurado, el valor del parámetro de configuración `ckan.site_url` deberá coincidir con el nombre de dominio de tu Andino (incluyendo el *schema*, `http` o `https`):

```
ckan.site_url=http://datos.ministerio.gob.ar/
```

Si éste no coincide, deberás modificar el valor del parámetro en la configuración de tu Andino.

Actualizando el nombre de dominio asignado a Andino

Para actualizar el nombre de dominio que tiene tu andino (por ejemplo `datos.ministerio.gob.ar`) debés ejecutar el siguiente comando:

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "ckan.site_url=http://datos.ministerio.gob.ar/";
```

VERIFICÁ QUE EL CONTENEDOR PORTAL PUEDA RESOLVER EL NOMBRE DE DOMINIO ASIGNADO A LA INSTANCIA

Para asegurar el correcto funcionamiento de algunos componentes de la arquitectura de Andino, es necesario que desde dentro de los contenedores Docker que forman parte de la solución, el nombre de dominio asignado a tu Andino pueda ser resuelto correctamente.

La verificación de tal condición está documentada en la sección DNS .

SEGUIR LAS RECOMENDACIONES DE SEGURIDAD

La presente documentación contiene un apartado acerca de recomendaciones de seguridad. Por favor leelas y seguí las recomendaciones .

MIGRACION

En el presente documento, se pretende explicar cómo llevar a cabo una migración de la versión 1.0 de andino a la versión 2.0 de andino.

INDICE

- 1) Requisitos
- 2) Script de migración automático.
- 3) Migración manual
 - 3.1) Backup de la base de datos
 - 3.2) Backup de los archivos de la aplicación
- 4) Instalación
 - 4.1) Detener la aplicación
 - 4.2) Instalar la aplicación
- 5) Restores
 - 5.1) Restaurar los archivos
 - 5.2) Restaurar la base de datos
 - 5.3) Regenerar el índice de búsquedas

1) REQUISITOS

Se requiere tener instalado:

- jq >= 1.5
- docker
- docker-compose

Se asume que, en el servidor, hay 3 containers de docker corriendo:

- `app-ckan`
- `pg-ckan`
- `solr-ckan`

Además, se debe conocer los `usuarios` y `passwords` de la base de datos (tanto de la usada por `ckan` como por el `datastore`).

2) SCRIPT DE MIGRACIÓN AUTOMÁTICO.

El repositorio cuenta con un script para correr la migración automáticamente. (el mismo se puede encontrar en `install/migrate.sh` , dentro del repositorio). Ciertas variables de entorno y tener instalado `docker` y `docker-compose` . Debe ser ejecutado con `sudo` o `root` .

Ejemplo:

```
export EMAIL=admin@example.com
export HOST=andino.midomionio.com.ar
export DB_USER=usuario
export DB_PASS=password
export STORE_USER=dsuser
export STORE_PASS=dspass
sudo -E ./migrate.sh
```

3) MIGRACIÓN MANUAL

Para realizar la migración manual, debemos conocer las variables con las que se inicializó el portal. En este caso, serán las siguientes:

```
export EMAIL=admin@example.com
export HOST=andino.midomionio.com.ar
export DB_USER=usuario
export DB_PASS=password
export STORE_USER=dsuser
export STORE_PASS=dspass
```

3.1) BACKUP DE LA BASE DE DATOS

Es necesario hacer un backup de la base de datos antes de empezar con la migración. La misma puede llevarse a cabo con el siguiente script:

```
#!/usr/bin/env bash
set -e;

old_db="pg-ckan"
database_backup="backup.gz"

echo "Creando backup de la base de datos."

backupdir=$(mktemp -d)

backupfile="$backupdir/$database_backup"
echo "Iniciando backup de $old_db"
echo "Usando directorio temporal: $backupdir"
docker exec $old_db pg_dumpall -c -U postgres | gzip > "$backupfile"
echo "Copiando backup a $PWD"

cp "$backupfile" $PWD
echo "Backup listo."
```

Este script dejará un archivo `backup.gz` en el directorio actual.

3.2) BACKUP DE LOS ARCHIVOS DE LA APLICACIÓN

Es necesario hacer un backup de los archivos de la aplicación: configuración y archivos subidos. El mismo puede llevarse a cabo con el siguiente script:

Nota: Requiere jq >= 1.5

```
#!/usr/bin/env bash
set -e;

old_andino="app-ckan"
app_backup="backup.tar.gz"

echo "Creando backup de los archivos de configuración."
backupdir=$(mktemp -d)
today=`date +%Y-%m-%d.%H:%M:%S`
appbackupdir="$backupdir/application/"
mkdir $appbackupdir
echo "Iniciando backup de los volúmenes en $old_andino"
echo "Usando directorio temporal: $backupdir"
docker inspect --format '{{.json .Mounts}}' $old_andino | jq -r '.[[]|.
Name, .Source, .Destination] | @tsv' |
while IFS=$'\t' read -r name source destination; do
    echo "Guardando archivos de $destination"
    if ls $source/* 1> /dev/null 2>&1; then
        echo "Nombre del volumen: $name."
        echo "Directorio en el Host: $source"
        echo "Destino: $destination"
        dest="$appbackupdir$name"
        mkdir -p $dest
        echo "$destination" > "$dest/destination.txt"

        tar -C "$source" -zcvf "$dest/backup_$today.tar.gz" $(ls $sourc
e)
        echo "List backup de $destination"
    else
        echo "Ningún archivo para $destination";
    fi
done
echo "Generando backup en $app_backup"
tar -C "$appbackupdir../" -zcvf $app_backup "application/"
echo "Backup listo."
```

Este script dejará un archivo backup.tar.gz en el directorio actual. El mismo, una vez descomprimido, contendrá la siguiente estructura (por ejemplo):

```
- application/
  └─ 61ee6cc7dc974476fe3300cc4325d913ed2f949494419b11a5c7c897fa91910
6   └─ backup_2017-05-19.10:56:09.tar.gz
   └─ destination.txt
0   └─ b1bf820976c3220e54136e4db229a67a9d9292896ad8d91623030e3b7171f21
    └─ backup_2017-05-19.10:56:09.tar.gz
      └─ destination.txt
```

Cada sub-directorio contiene el ID del volumen en docker usado; los números varían de volumen en volumen. Dentro de cada sub-directorio se encuentra un archivo *.tar.gz* junto con un archivo *destination.txt*. El archivo *destination.txt* indica dónde corresponde la información dentro del container. El archivo *.tar.gz* contiene una carpeta *_data* con los archivos.

4) INSTALACIÓN

4.1) DETENER LA APLICACIÓN

Debemos detener la aplicación para lograr que se liberen los puertos usados. Por ejemplo, el puerto 80.

```
docker stop solr-ckan pg-ckan app-ckan
```

4.2) INSTALAR LA APLICACIÓN

Ver la documentación Aquí

Nota: Actualizar la versión de docker y docker-compose de ser necesario.

Ahora, es necesario restaurar tanto la base de datos como los archivos de la aplicación.

5) RESTORES

5.1) RESTAURAR LOS ARCHIVOS

Descomprimir el archivo `backup.tar.gz`. En cada subdirectorio encontraremos el archivo `destination.txt`; el contenido de este archivo nos ayudará a saber donde debemos copiar los archivos. Con el siguiente comando, podremos saber qué volúmenes hay montados en el nuevo esquema y dónde debemos copiar los archivos dentro del `backup_*.tar.gz`

Correr `docker inspect andino -f '{{ json .Mounts }}' | jq :`

El comando mostrará, por ejemplo, lo siguiente:

```
[
{
  "Type": "volume",
  "Name": "ald87160a04e270302582849c9ce5c6dbb44719a94b702158aeaf23835f7862f",
  "Source": "/var/lib/docker/volumes/ald87160a04e270302582849c9ce5c6dbb44719a94b702158aeaf23835f7862f/_data",
  "Destination": "/etc/ckan/default",
  "Driver": "local",
  "Mode": "",
  "RW": true,
  "Propagation": ""
},
{
  "Type": "volume",
  "Name": "7ab721966628bf692a3d451567c9a01b419ba5189b88ef05484de315c73f6275",
  "Source": "/var/lib/docker/volumes/7ab721966628bf692a3d451567c9a01b419ba5189b88ef05484de315c73f6275/_data",
  "Destination": "/usr/lib/ckan/default/src/ckanext-gobar-theme/ckanext/gobar_theme/public/user_images",
  "Driver": "local",
  "Mode": "",
  "RW": true,
  "Propagation": ""
},
...
]
```

Como podemos ver, hay una entrada "Destination" que coincidirá con el contenido del archivo `destination.txt` en cada directorio. Debemos asegurarnos de no copiar el archivo `production.ini`, ya que el mismo cambia bastante de versión en versión.

El restore puede ser llevado a cabo con el siguiente script:

```
#!/usr/bin/env bash
set -e;

echo "Iniciando recuperación de Archivos."
install_dir="/etc/portal";
container="andino"
app_backup="backup.tar.gz"

containers=$(docker ps -q)
if [ -z "$containers" ]; then
    echo "No se encontró ningún contenedor corriendo."
```



```

else
    docker stop $containers
fi

restoredir=$(mktemp -d)
echo "Usando directorio temporal $restoredir"
tar zxvf $app_backup -C $restoredir

docker inspect --format '{{json .Mounts}}' $container | jq -r '.[0][.Name, .Source, .Destination] | @tsv' |
while IFS=$'\t' read -r name source destination; do
    for directory in $restoredir/application/*; do
        dest=$(cat "$directory/destination.txt")
        if [ "$dest" == "$destination" ]; then
            echo "Recuperando archivos para $destination"
            tar zxvf "$directory/$(ls "$directory" | grep backup)" -C "$source"
        fi
    done
done
echo "Restauración lista."
echo "Reiniciando servicios."
cd $install_dir;
docker-compose -f latest.yml restart;
cd -;

```

5.2) RESTAURAR LA BASE DE DATOS

Para restaurar la base de datos, se puede usar el siguiente script contra el archivo previamente generado (backup.gz):

```

#!/usr/bin/env bash
set -e;

install_dir="/etc/portal";
database_backup="backup.gz"
container="andino-db"

echo "Iniciando restauración de la base de datos."
containers=$(docker ps -q)

if [ -z "$containers" ]; then
    echo "No se encontró ningún contenedor corriendo."
else
    docker stop $containers
fi
docker restart $container
sleep 10;

restoredir=$(mktemp -d);
echo "Usando directorio temporal $restoredir"

restorefile="$restoredir/dump.sql";

gzip -dkc < $database_backup > "$restorefile";
echo "Borrando base de datos actual."
docker exec $container psql -U postgres -c "DROP DATABASE IF EXISTS ckan;"
docker exec $container psql -U postgres -c "DROP DATABASE IF EXISTS datastore_default;"
echo "Restaurando la base de datos desde: $restorefile"
cat "$restorefile" | docker exec -i $container psql -U postgres
echo "Recuperando credenciales de los usuarios"
docker exec $container psql -U postgres -c "ALTER USER $DB_USER WITH PASSWORD '$DB_PASS';"

```

```
docker exec $container psql -U postgres -c "ALTER USER $STORE_USER WITH PASSWORD '$STORE_PASS';"
```

```
echo "Restauración lista."
echo "Reiniciando servicios."
cd $install_dir;
docker-compose -f latest.yml restart;
cd -;
```

5.3) REGENERAR EL ÍNDICE DE BÚSQUEDAS

Para regenerar el índice de búsquedas, debemos ir al directorio donde se instaló la aplicación y correr el siguiente comando:

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/run_rebuild_search.sh
```

INDICE

- Mantenimiento
- Exploración de la instancia de andino
 - ¿Qué está corriendo docker?
 - Utilización del archivo latest.yml en los comandos de docker-compose
 - Ingresar al contenedor principal de andino
 - Listar todas las `Propiedades` de cada contenedor
- Administración de usuarios
 - Crear un usuario ADMIN
 - Listar mis usuarios
 - Ver los datos de un usuario
 - Crear un nuevo usuario
 - Crear un nuevo usuario extendido
 - Eliminar un usuario
 - Cambiar password de un usuario
 - Usuario administrador de CKAN
- Configuraciones de andino
 - Modificar el archivo de configuración
 - Cambiar la configuración del SMTP
 - Cambiar el remitente de los correos electrónicos que envía Andino
 - Cambiar el id del container de Google Tag Manager
 - Google Tag Manager
 - Cambiar el id del tag de Google Analytics
 - Indexar datasets con Google Dataset Search
 - Deshabilitar la URL `/catalog.xlsx`
 - Configuración de la llamada de invalidación de caché
 - Caché externa
 - Configuración de CORS
 - Configuración del explorador de series de tiempo
- Acceso a los datos de andino

- Encontrar los volúmenes de mi andino dentro del filesystem del host
- Ver las direcciones IP de mis contenedores
- Ver las variables de entorno que tienen mis contenedores
- Acceder con un cliente de PostgreSQL a las bases de datos
- Eliminar objetos definitivamente
 - Purgar Organizaciones Borradas
 - Purgar Grupos Borrados
 - Purgar Datasets Borrados
 - Listar nombres de los datasets contenidos en Andino
- Backups
 - Backup de la base de datos
 - Realizar un backup del file system
 - Realizar un backup de la configuración
- Comandos de DataPusher
 - Subir todos los recursos al Datastore
- Recomendaciones de Seguridad y Optimizaciones
 - HTTPS
 - Sistema y librerías
 - Firewall
 - SSH
- Optimización de logging
 - Configurar otro `logging driver`
 - Eliminar `logs` antiguos de `Docker`
 - Eliminar logs dentro de Andino

MANTENIMIENTO

EXPLORACIÓN DE LA INSTANCIA DE ANDINO

¿QUÉ ESTÁ CORRIENDO DOCKER?

Para obtener una lista de lo que está corriendo actualmente Docker, podemos usar el siguiente comando:

```
docker ps # Tabla de ejecucion actual
docker ps -q # Listado de IDs de cada contenedor
docker ps -aq # Listado de IDs de todos los contenedores disponibles.
```

UTILIZACIÓN DEL ARCHIVO LATEST.YML EN LOS COMANDOS DE DOCKER-COMPOSE

En múltiples secciones de esta documentación se emplea el uso de comandos que comienzan de la siguiente manera:

```
docker-compose -f latest.yml
```

Es importante recordar que no alcanzaría con especificar el directorio absoluto del archivo `latest.yml`; es necesario ejecutar estos comandos *exactamente en ese mismo directorio*, debido a que ahí también se encuentra el archivo que contiene las variables de entorno (`.env`) ya que es el directorio de instalación de Andino.

INGRESAR AL CONTENDOR PRINCIPAL DE ANDINO

El contenedor principal de andino, donde se ejecuta la aplicación CKAN, es denominado `portal`. Para ingresar en una sesión de consola en el contenedor, ejecutar:

```
docker-compose -f latest.yml exec portal /bin/bash
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

LISTAR TODAS LAS PROPIEDADES DE CADA CONTENEDOR

```
docker-compose -f latest.yml ps -q portal solr db | xargs -n 1 | while read container; do docker inspect $container; done
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

ADMINISTRACIÓN DE USUARIOS

CREAR UN USUARIO ADMIN

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/add_admin.sh mi_nuevo_usuario_admin email_del_usuario_admin
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

El comando solicitará la contraseña del usuario administrador.

LISTAR MIS USUARIOS

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/paster.sh --p lugin=ckan user list
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

VER LOS DATOS DE UN USUARIO

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/paster.sh --p lugin=ckan user nombre-de-usuario
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

CREAR UN NUEVO USUARIO

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/paster.sh --p lugin=ckan user add nombre-de-usuario
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

CREAR UN NUEVO USUARIO EXTENDIDO

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/paster.sh --p lugin=ckan user add nombre [email=mi-usuario@host.com password=mi-contraseña-rara apikey=unsecretomisticonoleible]
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

ELIMINAR UN USUARIO

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/paster.sh --p lugin=ckan user remove nombre-de-usuario
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose`.

CAMBIAR PASSWORD DE UN USUARIO

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/paster.sh --p lugin=ckan user setpass nombre-de-usuario
```

Ver la sección sobre la utilización del archivo latest.yml en los comandos de docker-compose .

USUARIO ADMINISTRADOR DE CKAN

Existe un usuario administrador llamado *default* , el cual es utilizado por la aplicación para la ejecución de ciertas funciones que corren de fondo (por ejemplo, la actualización de la caché del data.json). Es muy importante que dicho usuario **siempre** esté disponible ya que, en caso de no estarlo, provocaría un funcionamiento incorrecto en el portal; por lo tanto, se recomienda muy fuertemente no intentar eliminarlo ni desactivarlo.

CONFIGURACIONES DE ANDINO

MODIFICAR EL ARCHIVO DE CONFIGURACIÓN

El archivo de configuración de andino se llama `production.ini` , y se lo puede encontrar y modificar de la siguiente manera:

```
# Ingresar al contenedor

cd /etc/portal
docker-compose -f latest.yml exec portal /bin/bash

# Una vez adentro, abrimos el archivo production.ini, y buscamos la sección que necesita ser modificada

vim /etc/ckan/default/production.ini

# Editamos y, luego de salir del contenedor, lo reiniciamos

docker-compose -f latest.yml restart portal nginx
```

CAMBIAR LA CONFIGURACIÓN DEL SMTP

Por defecto, andino usará un servidor postfix integrado para el envío de emails. Para usar un servidor SMTP propio, debemos cambiar la configuración del archivo `production.ini` . Para lograrlo, podemos hacerlo de dos formas:

1) Ingresando al contenedor.

Debemos buscar y editar en el archivo `production.ini` la configuración de email que luce como:

```
## Email settings

error_email_from=admin@example.com
smtp.server = postfix
#smtp.starttls = False
smtp.user = portal
smtp.password = portal
smtp.mail_from = administrador
```

Para saber cómo hacerlo, leer la sección que explica cómo modificar el archivo de configuración

2) Ejecutando comandos paster

Suponiendo que nuestro servidor SMTP está en smtp.gmail.com, la dirección de correo del usuario es `smtp_user_mail@gmail.com` , la contraseña de esa dirección de correo `mi_pass` y queremos usar "tls", podemos ejecutar los siguientes comandos:

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "smtp.server=smtp.gmail.com:587";
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "smtp.user=smtp_user_mail@gmail.com";
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "smtp.password=mi_pass";
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "smtp.starttls=True";
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "smtp.mail_from=smtp_user_mail@gmail.com";
```

```
# Finalmente reiniciamos el contenedor
docker-compose -f latest.yml restart portal nginx
```

Tener en cuenta que si se utiliza un servidor SMTP, se debe setear la configuración con **un correo electrónico de @gmail.com**, y que **starttls debe estar en True**.

CAMBIAR EL REMITENTE DE LOS CORREOS ELECTRÓNICOS QUE ENVÍA ANDINO

Para modificar el remitente de los correos electrónicos que el sistema envía (por ejemplo los de creación de usuarios nuevos o los de olvido de contraseña), se deben seguir los pasos de la sección Cambiar la configuración del SMTP pero modificando el atributo de configuración `smtp.mail_from`.

CAMBIAR EL ID DEL CONTAINER DE GOOGLE TAG MANAGER

Será necesario modificar la configuración en el archivo `production.ini`.

Para saber cómo hacerlo, leer la sección que explica cómo modificar el archivo de configuración.

Esta vez, buscaremos la configuración debajo de la sección `[app:main]` (vas a encontrar campos como `"superThemeTaxonomy"` y `"ckan.site.title"`).

El campo que estamos buscando es `ckan.google_tag_manager.gtm_container_id`.

En caso de no encontrar el campo mencionado, lo podemos agregar:

```
ckan.google_tag_manager.gtm_container_id = { id que necesitás guardar }
```

GOOGLE TAG MANAGER

Para configurar el código de seguimiento de Google Tag Manager ejecutar el siguiente comando:

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "ckan.google_tag_manager.gtm_container_id=<tu código de seguimiento GTM>";
```

```
# Finalmente reiniciamos el contenedor
docker-compose -f latest.yml restart portal nginx
```

CAMBIAR EL ID DEL TAG DE GOOGLE ANALYTICS

Será necesario modificar el archivo de configuración `production.ini`.

Para saber cómo hacerlo, leer la sección que explica cómo modificar el archivo de configuración.

La sección a buscar luce de esta manera:

```
## Google Analytics
googleanalytics.id = { un id }
googleanalytics_resource_prefix = { un prefix }
googleanalytics.domain = { un dominio }
```

Lo que se debe modificar es el campo `googleanalytics.id`.

INDEXAR DATASETS CON GOOGLE DATASET SEARCH

Andino cuenta con la posibilidad de utilizar Google Dataset Search para que éste indexe tus datasets. Para el mejor entendimiento de la herramienta, recomendamos que entres a <https://search.google.com/search-console/about>.

Por default, esta configuración se encuentra desactivada. Para activarla, podés ir a *Configuración -> Configuración avanzada -> Google Dataset Search* -> Clickear el checkbox y guardar el cambio.

DESHABILITAR LA URL /CATALOG.XLSX

En caso de desear deshabilitar la URL `/catalog.xlsx`, se puede ejecutar el siguiente comando:

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "andino.disable_catalog_xlsx_url=True";
```

En caso de querer restaurarlo, se debe configurar el atributo `andino.disable_catalog_xlsx_url` con el valor `False`.

CONFIGURACIÓN DE LA LLAMADA DE INVALIDACIÓN DE CACHE

La aplicación puede ser configurada para hacer una llamada HTTP ante cada cambio en los metadatos del portal. Esta llamada (A.K.A. "hook") puede configurarse para ser a cualquier URL, y usando cualquier método HTTP. Se deberá utilizar un campo llamado `andino.cache_clean_hook`, que tendrá asignada la URL a la cual queremos enviarle requests HTTP que lograrán ese efecto.

Además, el paquete "Andino" provee una configuración de *nginx* que permite recibir esta llamada e invalidar la caché.

Para configurar internamente *nginx* y *andino*, sólo es necesario pasar la opción `--nginx-extended-cache` al momento de usar el script de instalación.

Si nuestra aplicación ya está instalada, podemos seguir los siguientes pasos:

1. Actualizar a la última versión de la aplicación, con el script de actualización.
2. Ir al directorio de instalación `cd /etc/portal`
3. Editar el archivo `.env`
4. Agregar una línea nueva que sea: `NGINX_CONFIG_FILE=nginx_extended.conf`
5. Reiniciar el contenedor de *nginx* `docker-compose -f latest.yml up -d nginx`

Luego configuramos el hook de invalidación:

1. Entramos al contenedor del portal: `docker-compose -f latest.yml exec portal bash`
2. Configuramos el hook: `/etc/ckan_init.d/update_conf.sh`
`andino.cache_clean_hook=http://nginx/meta/cache/purge`
3. Salimos `exit`
4. Reiniciamos el portal: `docker-compose -f latest.yml restart portal nginx`

Nota: tener en cuenta que, por defecto, se emplea el método PURGE para disparar el hook, lo cual se puede cambiar editando el campo `andino.cache_clean_hook_method` dentro del archivo de configuración `production.ini`. Para saber cómo hacerlo, leer la sección que explica cómo modificar el archivo de configuración.

CACHÉ EXTERNA

Es posible implementar la caché externa por fuera del paquete *andino*. Para esto, en el servidor que servirá de caché, necesitamos instalar *openresty*. Esta plataforma web nos permite correr **nginx** y modificar su comportamiento usando *lua*.

Luego de instalar **openresty**, debemos activarlo para que empiece cada vez que se prenda el servidor:

```
systemctl enable openresty
systemctl restart openresty
```

Luego de instalar *openresty*, debemos agregar los archivos de configuración. Primero borramos la configuración de *nginx* que viene por defecto en `/etc/openresty/nginx.conf` y agregamos la nuestra:

```
#user nobody;
worker_processes 1;

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

#pid logs/nginx.pid;

events {
    worker_connections 1024;
}
```

```

http {
    include      mime.types;
    default_type application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #              '$status $body_bytes_sent "$http_referer" '
    #              '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile      on;
    #tcp_nopush   on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}

```

Luego, creamos el directorio donde pondremos la configuración de nuestra caché y creamos el archivo.

```

mkdir -p /etc/nginx/conf.d/
touch /etc/nginx/conf.d/000-andino-cache.conf

```

El archivo `000-andino-cache.conf` contendrá lo siguiente. Es necesario cambiar la palabra `IP_A_ANDINO` por la IP donde está andino.

```

proxy_cache_path /tmp/nginx_cache/ levels=1:2 keys_zone=cache:30m max_size=250m;
proxy_temp_path /tmp/nginx_proxy 1 2;

```

```

server_tokens off;

```

```

server {
    client_max_body_size 100M;
    location / {
        proxy_pass http://IP_A_ANDINO:80/;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_cache cache;

        # Disable cache for logged-in users
        proxy_cache_bypass $cookie_auth_tkt;
        proxy_no_cache $cookie_auth_tkt;
        proxy_cache_valid 30m;
        proxy_cache_key $host$scheme$proxy_host$request_uri;

        # Ignore apache "Cache-Control" header
        # See https://lists.okfn.org/pipermail/ckan-dev/2016-March/0098
64.html
        proxy_ignore_headers Cache-Control;
        # In emergency comment out line to force caching
        # proxy_ignore_headers X-Accel-Expires Expires;

        # Show cache status
        add_header X-Cache-Status $upstream_cache_status;
    }

    location /meta/cache/purge {

```



```

allow 192.168.0.0/16;
allow 172.16.0.0/12;
allow 10.0.0.0/8;
allow 127.0.0.1;
deny all;
if ($request_method = PURGE ) {
    content_by_lua_block {
        filename = "/tmp/nginx_cache/"
        local f = io.open(filename, "r")
        if (f~=nil) then
            io.close(f)
            os.execute('rm -rf "'..filename..'")
        end
        ngx.say("OK")
        ngx.exit(ngx.OK)
    }
}
}
}

```

Finalmente, reiniciamos **openresty** : `systemctl restart openresty` .

Ahora que tenemos la caché configurada, necesitamos configurar la llamada, o hook, de invalidación de caché. Para esto, entramos al servidor donde está corriendo andino y corremos:

`IP_INTERNA_CACHE=<ip interna del servidor de caché>`

```

cd /etc/portal
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh
h "andino.cache_clean_hook=http://$IP_INTERNA_CACHE/meta/cache/purge";
docker-compose -f latest.yml restart portal nginx

```

Si queremos probar la integración, podemos entrar al contenedor de andino y probar invalidar la caché:

```

IP_INTERNA_CACHE=<ip interna del servidor de caché>
cd /etc/portal
docker-compose -f latest.yml exec portal curl -X PURGE "http://$IP_INTERNA_CACHE/meta/cache/purge";
# => OK

```

NOTA: Si estamos usando nuestro andino con un IP y *no* con un dominio, tendremos que cambiar la configuración `ckan.site_url` para que use la IP del servidor donde se encuentra la caché externa.

`IP_PUBLICA_CACHE=<ip publica del servidor caché>`

```

cd /etc/portal
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh
h "ckan.site_url=http://$IP_PUBLICA_CACHE";
docker-compose -f latest.yml restart portal nginx

```

CONFIGURACIÓN DE CORS

Cuando es necesario acceder a Andino desde URLs distintas que apuntan a una misma instancia (ej: accediendo a través de un gateway/caché o directamente a la instancia de Andino o usando la IP pública del servidor *host*) es necesario, para el correcto funcionamiento de Andino, configurar parámetros para habilitar CORS (*Cross-Origin Resource Sharing*). Esto se debe a que un Andino debe tener una URL canónica, por lo tanto, las demás URLs utilizadas deben estar en el *whitelist* de CORS de Andino.

Para poder navegar tu Andino usando como URL una que no es la canónica de tu instancia, tenés que realizar dos acciones (los comandos deben ser ejecutados desde el directorio de instalación de Andino; por *default* , `/etc/portal`):

1. Habilitar el comportamiento CORS: `docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "ckan.cors.origin_allow_all = false"` (si bien el parámetro de configuración tiene el valor `false` , esto habilita el control de URLs contra el *whitelist*).

2. Agregar las URLs al *whitelist*: `docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh "ckan.cors.origin_whitelist=http://localhost:8080 http://127.0.0.1 http://127.0.0.1:8080" (en el ejemplo se habilitan las URLs http://localhost:8080 , http://127.0.0.1 y http://127.0.0.1:8080).`

Luego reiniciá los contenedores `portal` y `nginx` : `docker-compose -f latest.yml restart nginx portal` .

Si deseás habilitar **todas** las URLs para CORS (no recomendado), en el paso 1 debés pasar el valor `true` para el atributo de configuración `ckan.cors.origin_allow_all` .

Para ver más acerca del funcionamiento de CORS en CKAN ver la documentación oficial de CKAN (en inglés) .

CONFIGURACIÓN DEL EXPLORADOR DE SERIES DE TIEMPO

Andino tiene instalado el plugin `ckanext-seriestiempoexplorer` , pero no se encuentra presente entre los plugins activos. Para activarlo, debemos entrar al contenedor de andino, editar el archivo `/etc/ckan/default/production.ini` y agregar el `seriestiempoexplorer` a la lista de plugins. Luego de agregarlo, debemos reiniciar el servidor.

```
cd /etc/portal
docker-compose -f latest.yml exec portal bash;
vim /etc/ckan/default/production.ini
# ...
apachectl restart
```

Luego, si vamos a la configuración del sitio, podremos apreciar que se agrego una nueva sección "Series" en el apartado "Otras secciones del portal".

ACCESO A LOS DATOS DE ANDINO

ENCONTRAR LOS VOLÚMENES DE MI ANDINO DENTRO DEL FILESYSTEM DEL HOST

```
docker-compose -f latest.yml ps -q andino solr db | xargs -n 1 | while
read container; do docker inspect -f ' {{.Name}}: {{range .Mounts}}{{.S
ource}}: {{.Destination}} {{end}} ' $container; done
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose` .

VER LAS DIRECCIONES IP DE MIS CONTENEDORES

```
docker-compose -f latest.yml ps -q andino solr db | xargs -n 1 | while
read container; do docker inspect -f '{{.Name}}: {{range .NetworkSettin
gs.Networks}}{{.IPAddress}}{{end}}' $container; done
```

VER LAS VARIABLES DE ENTORNO QUE TIENEN MIS CONTENEDORES

```
docker-compose -f latest.yml ps -q andino solr db | xargs -n 1 | while
read container; do docker inspect -f '{{range $index, $value := .Config
.Env}}export {{$.value}}{{println}}{{end}}' $container; done
```

Ver la sección sobre la utilización del archivo `latest.yml` en los comandos de `docker-compose` .

ACCEDER CON UN CLIENTE DE POSTGRESQL A LAS BASES DE DATOS

```
docker-compose -f dev.yml exec db psql -U postgres
# psql \c ckan db default CKAN
# psql \c datastore_default db datastore CKAN
```

ELIMINAR OBJETOS DEFINITIVAMENTE

Es bien sabido que, dentro de CKAN, cada vez que borramos algun elemento, en verdad no se borra, sino que pasa a estar inactivo; por lo tanto, tener alguna forma de eliminar elementos de manera definitiva resulta altamente necesario.

PURGAR ORGANIZACIONES BORRADAS

```
curl -X POST http://tu-host/api/3/action/organization_purge -H "Authorization:tu-api-key" -F id=id-o-nombre-que-se-desea-borrar
```

PURGAR GRUPOS BORRADOS

```
curl -X POST http://tu-host/api/3/action/group_purge -H "Authorization:tu-api-key" -F id=id-o-nombre-que-se-desea-borrar
```

PURGAR DATASETS BORRADOS

```
curl -X POST http://tu-host/api/3/action/dataset_purge -H "Authorization:tu-api-key" -F id=id-o-nombre-que-se-desea-borrar
```

LISTAR NOMBRES DE LOS DATASETS CONTENIDOS EN ANDINO

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/paster.sh --plugin=ckan dataset list | grep -v DEBUG | grep -v count | grep -v Datasets | xargs -n2 | while read id name; do echo $name; done
```

Ver la sección sobre la utilización del archivo latest.yml en los comandos de docker-compose .

BACKUPS

Es altamente recomendable hacer copias de seguridad de los datos de la aplicación, tanto la base de datos como los archivos de configuración y subidos por los usuarios.

BACKUP DE LA BASE DE DATOS

Un ejemplo fácil de hacer un backup de la base de datos sería:

```
container=$(docker-compose -f latest.yml ps -q db)
today=`date +%Y-%m-%d.%H:%M:%S`
filename="backup-$today.gz"

# Creo un directorio temporal y defino dónde generaré el backup
backupdir=$(mktemp -d)
backupfile="$backupdir/$filename"

# Exporto la base de datos
docker exec $container pg_dumpall -c -U postgres | gzip > "$backupfile"

# Copio el archivo al directorio actual y borro el original
# Podría reemplazar $PWD con mi directorio de backups, como /etc/portal/backups
mv "$backupfile" $PWD
```

Y para los demás archivos de la aplicación (requiere `jq`):

```
backupdir=$(mktemp -d)
today=`date +%Y-%m-%d.%H:%M:%S`
appbackupdir="$backupdir/application/"
mkdir $appbackupdir
container=$(docker-compose -f latest.yml ps -q portal)

docker inspect --format '{{json .Mounts}}' $container | jq -r '.[][.Name, .Source, .Destination] | @tsv' |
while IFS=$'\t' read -r name source destination; do

    if ls $source/* 1>> /dev/null 2>&1; then
        dest="$appbackupdir$name"
        mkdir -p $dest
```

```

        tar -C "$source" -zcvf "$dest/backup_$today.tar.gz" $(ls $source)
    else
        echo "No file at $source"
    fi
done

tar -C "$appbackupdir../" -zcvf backup.tar.gz "application/"

```

Podría colocarse esos scripts en el directorio donde se instaló la aplicación (ejemplo : `/etc/portal/backup.sh`) y luego agregar un `cron` :

Para correr el script cada domingo, podríamos usar la configuración `0 0 * * 0` (ver cron para más información), correr el comando `crontab -e` y agregar la línea:

```
0 0 * * 0 cd /etc/portal/ && bash /etc/portal/backup.sh
```

REALIZAR UN BACKUP DEL FILE SYSTEM

```
# Exporto el path al almacenamiento del volumen
export CKAN_FS_STORAGE=$(docker inspect --format '{{ range .Mounts }}{{ if eq .Destination "/var/lib/ckan" }}{{ .Source }}{{ end }}{{ end }}' andino)
```

```
# Creo un tar.gz con la info.
tar -C "$(dirname "$CKAN_FS_STORAGE")" -zcvf /ruta/para/guardar/mis/bkps/mi_andino.fs-data_$(date +%F).tar.gz "$(basename "$CKAN_FS_STORAGE")"
```

REALIZAR UN BACKUP DE LA CONFIGURACIÓN

```
# Exporto el path al almacenamiento del volumen
export ANDINO_CONFIG=$(docker inspect --format '{{ range .Mounts }}{{ if eq .Destination "/etc/ckan/default" }}{{ .Source }}{{ end }}{{ end }}' andino)
```

```
# Creo un tar.gz con la info.
tar -C "$(dirname "$ANDINO_CONFIG")" -zcvf /ruta/para/guardar/mis/bkps/mi_andino.config-data_$(date +%F).tar.gz "$(basename "$ANDINO_CONFIG")"
```

COMANDOS DE DATAPUSHER

Deben ser corridos en el directorio de instalación de Andino.

SUBIR TODOS LOS RECURSOS AL DATASTORE

Es posible que existan recursos que no hayan sido subidos al Datastore. Para buscar e intentar subir dichos recursos, ejecutar:

```
docker-compose -f latest.yml exec portal /usr/lib/ckan/default/bin/paster --plugin=ckan datapusher submit_all -c /etc/ckan/default/production.ini
```

Se preguntará si se desea proceder. Al escribir que sí (`y`), iniciar la subida de recursos. Para cada uno de ellos, se escribirá su id y luego el status: si subió correctamente, aparecerá " `OK` "; en caso contrario, " `FAIL` ".

RECOMENDACIONES DE SEGURIDAD Y OPTIMIZACIONES

Mantener seguro un servidor web puede ser una tarea ardua, pero sobre todo es *constante*, ya que *constantemente* se detectan nuevas vulnerabilidades en los distintos softwares. Y un servidor web no es la excepción! En este breve apartado, se darán pequeñas recomendaciones para mantener seguro el servidor, no solo antes posibles atacantes, sino tambien ante posibles fallos del sistema y como efectuar mitigaciones.

Las siguientes recomendaciones pueden ser implementadas fácilmente en un sistema Ubuntu 16.04, el cual es el recomendado (a la fecha), para correr la aplicación.

HTTPS

HTTPS permite que la conexión entre el *browser* y el servidor sea encriptada y de esta manera segura. Es altamente recomendable usar HTTPS, para mantener la privacidad de los usuarios. El portal de documentación para desarrolladores de Google provee buena información sobre esto: <https://developers.google.com/web/fundamentals/security/encrypt-in-transit/why-https>

SISTEMA Y LIBRERÍAS

Es *altamente recomendable* mantener el sistema operativo y las aplicaciones que usemos actualizadas. Constantemente se están subiendo *fixes* de seguridad y posibles intrusos podrían aprovechar que las aplicaciones o el mismo sistema operativo estén desactualizados. Periódicamente, podríamos constatar las nuevas versiones de nuestro software y actualizar dentro de lo posible. Como ejemplo, podemos ver que para Ubuntu 16.04 salió Ubuntu 16.04.2, con algunas correcciones de seguridad. Ver .

FIREWALL

Todo servidor debe tener activado el firewall. El firewall permitirá denegar (o permitir) el acceso a la red. En un servidor web, el puerto abierto al público deberían ser sólo el 80 (http) y el 443 (https). Además de ese puerto, si la máquina es accedida remotamente mediante un servidor SSH, deberíamos abrir este puerto también, pero con un límite de acceso. La solución es fácilmente implementable con el programa `ufw` .

SSH

Los servidores ssh permiten el acceso al servidor remotamente. **No debe permitirse el acceso por ssh mediante usuario y password** . Sólo debe permitirse el acceso mediante clave publica. DigitalOcean tiene una buena guía de cómo configurar las claves públicas Ver .

OPTIMIZACIÓN DE LOGGING

CONFIGURAR OTRO LOGGING DRIVER

Por default, docker escribe a un archivo con formato `json` , lo cual puede llevar a que se acumulen los logs de la aplicación, y estos archivos crezcan indefinidamente. Para evitar esto, se puede configurar el `logging driver` de docker. La recomendación es usar `journald` y configurarlo para que los logs sean persistentes.

ELIMINAR LOGS ANTIGUOS DE DOCKER

Dentro del normal funcionamiento de la plataforma, se genera una gran cantidad de logs, los cuales, ante un incidencia, son sumamente útiles. Pero, luego de un tiempo, y sabiendo que los mismos se almacenan internamente en Andino, podría ser necesario eliminarlos.

```
sudo su -c "ls /var/lib/docker/containers/ | xargs -n1 | while read docker_id; do truncate -s 0 /var/lib/docker/containers/${docker_id%/*}/${docker_id%/*}-json.log; done"
```

ELIMINAR LOGS DENTRO DE ANDINO

```
docker-compose -f latest.yml exec portal truncate -s 0 /var/log/apache2/*.log
```

Ver la sección sobre la utilización del archivo latest.yml en los comandos de docker-compose .

INDICE

- Configuración HTTPS
- Certificado SSL
- Configuración de SSL
 - Modificar el puerto
 - Realizar cambios en un Andino instalado
- Probar la configuración

- Renovar certificados SSL
- Deshabilitar SSL

CONFIGURACIÓN HTTPS

CERTIFICADO SSL

Andino cuenta con soporte *builtin* de certificados SSL. Es posible instalar Andino con SSL, actualizar una versión de Andino sin SSL y agregarle el soporte para SSL, e inclusive deshabilitar SSL a una instancia con SSL.

Lo único que un administrador de Andino necesita para habilitar SSL es contar con los certificados `.key` y `.crt` para nuestra aplicación y la elección de un puerto libre del host para usar SSL.

Cómo obtener estos archivos está fuera del "scope" de esta documentación.

CONFIGURACIÓN DE SSL

Tanto la instalación como la actualización de un Andino emplean el uso de un parámetro llamado `nginx_ssl`, el cual puede ser utilizado para especificar que se desea utilizar SSL.

Para especificar el path de los archivos del certificado, se debe utilizar los parámetros `ssl_key_path` y `ssl crt_path`. Los archivos dentro del contenedor `nginx` se llamarán `andino.key` y `andino.crt` respectivamente, y el proceso de instalación o actualización los copiará en el directorio `/etc/nginx/ssl`. En caso de que al menos uno de estos archivos no esté, *no se podrá utilizar el archivo de configuración para SSL* y se elegirá el default en su lugar. Hay que asegurarse de que el path de cada archivo sea válido (exista en el host), y que estén especificados en la instalación/actualización.

Un navegador web *no debería* mostrarnos ninguna advertencia si los certificados son correctos.

Un ejemplo de uso dentro del comando de instalación de Andino para los parámetros mencionados:

```
--nginx_ssl --ssl_key_path="/home/miusuario/Desktop/mi_archivo_key.key"
--ssl crt_path="/home/miusuario/Desktop/mi_archivo.crt"
```

Estos parámetros de configuración deben ser especificados en cada actualización de Andino, para mantener SSL habilitado.

MODIFICAR EL PUERTO

Para la instalación de Andino, el puerto a ser utilizado por default es el 443, pero éste puede ser cambiado mediante el parámetro `nginx_ssl_port` y un valor a elección.

Ejemplo:

```
--nginx_ssl_port=8443
```

Es importante para los administradores saber que Andino tomará el puerto especificado (o el default) ya sea que el portal use o no use HTTPS. En caso de no querer usar HTTPS y que el host tenga el puerto 443 tomado por un servidor web, es requisito especificar un puerto distinto (ejemplo: 8443) que será reservado por Andino, pero no utilizado.

REALIZAR CAMBIOS EN UN ANDINO INSTALADO

Para lograr que Andino implemente la configuración HTTPS, es necesario realizar una actualización de andino y especificar las opciones detalladas en la sección Configuración de SSL.

Estas opciones solo son válidas a partir de Andino `release-2.5.2`.

PROBAR LA CONFIGURACIÓN

Para asegurarse de que Nginx esté utilizando la configuración HTTPS, ejecutar el siguiente comando debería mostrar `nginx_ssl.conf`:

```
docker exec -it andino-nginx bash -c 'echo $NGINX_CONFIG_FILE'.
```

Si se está implementando la configuración HTTPS y los certificados fueron creados correctamente, el explorador debería redirigir cualquier llamada HTTP a HTTPS.

Si se especificó un puerto para SSL, el portal debería permitir el ingreso si el puerto es parte de la URL.

También deberías poder navegar el portal en el puerto SSL seleccionado.

RENOVAR CERTIFICADOS SSL

Para renovar los certificados SSL de tu instancia de Andino es tan sencillo como ejecutar una actualización de Andino. Para llevarlo a cabo es necesario que subas los dos archivos que componen el certificado (`.cer` y `.key`) y que ejecutes el comando de actualización de Andino, especificando la opción `--nginx_ssl` y las opciones que permiten configurar los archivos del certificado como estáá especificado en la sección Configuración de SSL .

Si deseás mantener la versión de Andino que tenés, debés especificar la opción `--andino_version` con la versión de tu instancia de Andino.

DESHABILITAR SSL

El proceso de deshabilitación de SSL se puede lograr mediante la ejecución del proceso de actualización de Andino `update.py` especificando el parámetro `--andino_version` a una versión igual a la del portal a configurar (es decir, se mantendrá la misma versión), pero no especificando el parámetro `--nginx_ssl` .

De esta manera el script de actualización usará la configuración de Andino que no habilita HTTPS y se habilitará el acceso por el puerto 80.

Ejemplo de uso:

```
sudo wget https://raw.githubusercontent.com/datosgobar/portal-andino/master/install/update.py
sudo python update.py --andino_version=<versión del andino del portal>
```

INDICE

- Configuración de DNS
- Introducción
- Cómo diagnosticar si tu andino tiene el problema
- Cómo resolver el problema
 - Configuración de DNS públicos
 - Configurar andino con el nuevo nombre de dominio
 - Configurar un alias en la red de Docker para el contenedor nginx
 - Verificando que el problema fue resuelto

CONFIGURACIÓN DE DNS

INTRODUCCIÓN

Algunas funcionalidades del Portal Andino requieren que la aplicación web o procesos externos (ej: DataPusher) puedan navegar sin restricciones desde el proceso en el cual corre la aplicación Python a la URL donde está publicado el sitio (definida por el setting `ckan.site_url`).

Esta URL debe poder ser accedida sin problemas de resolución de nombres o de ruteo de IPs desde el mismo contenedor *portal* para el correcto funcionamiento del sitio.

Algunas instancias de Andino han reportado problemas de funcionamiento debido a que la infraestructura donde están alojados no permite esta resolución de nombres de manera correcta. Este artículo describe cómo diagnosticar el problema y propone soluciones al mismo.

Recordá que *todos los comandos de este artículo deben ser ejecutados en el directorio donde instalaste andino, por ejemplo /etc/portal* .

CÓMO DIAGNOSTICAR SI TU ANDINO TIENE EL PROBLEMA

Para saber si es necesario realizar las configuraciones detalladas en este artículo, podés realizar los siguientes pasos detallados abajo.

1. Obtener el valor de `ckan.site_url` : el valor se puede obtener ejecutando el siguiente comando:
`docker-compose -f latest.yml exec portal grep ckan\.site_url /etc/ckan/default/production.ini` . Ese comando debería devolver `ckan.site_url = <URL de tu Andino>` .
2. Evaluar si tu andino puede navegar hasta la URL del sitio. Ejecutá el siguiente comando: `docker-compose -f latest.yml exec portal curl <URL de tu Andino>/data.json` .

Si el segundo paso no devuelve una respuesta en formato *json* con la información del catálogo de tu instancia idéntica a la que obtendrías navegando desde tu navegador a `<URL de tu Andino>/data.json` (por ejemplo, luego de un rato obtenés `curl: (7) Failed to connect to <URL de tu Andino> port 80: Connection timed out`), entonces *debés aplicar la configuración recomendada en este artículo* .

CÓMO RESOLVER EL PROBLEMA

La solución del problema se basa en asegurar que dentro de la red de Docker el nombre de dominio de Andino pueda resolverse correctamente a la IP pública del host, a la IP privada dentro de la red en la que está o a la IP interna dentro de la red de Docker.

Para lograrlo, planteamos distintas alternativas.

CONFIGURACIÓN DE DNS PÚBLICOS

Este paso no está cubierto por esta documentación, ya que puede ser resuelto mediante formas diversas, dependiendo de la infraestructura con la que contás.

Lo que debés hacer es contactarte con tu administrador de infraestructura y solicitar una IP pública fija y un nombre de dominio para la instancia de tu andino.

Probablemente ya tengas un nombre de dominio asignado a tu instancia de andino, con lo cual podés saltar este paso.

CONFIGURAR ANDINO CON EL NUEVO NOMBRE DE DOMINIO

Si ya tenés un nombre de dominio asignado para acceder a tu andino, y cuando lo instalaste lo configuraste usando ese nombre de dominio, podés saltar este paso.

Para configurar el nuevo nombre de dominio es necesario actualizar el setting `ckan.site_url` de la instancia de Andino. Esto lo podés lograr con el siguiente comando:

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/update_conf.sh
"ckan.site_url=http://<tu nombre de dominio>" .
```

Podés verificar que haya quedado bien configurado ejecutando:

```
docker-compose -f latest.yml exec portal grep ckan\.site_url
/etc/ckan/default/production.ini .
```

Para reflejar los cambios, es necesario reiniciar la aplicación web del contenedor `portal` :

```
docker-compose -f latest.yml exec portal apachectl restart .
```

Finalmente, si ya tenías datos cargados en tu andino, necesitás regenerar el índice de búsqueda, usando el siguiente comando:

```
docker-compose -f latest.yml exec portal /etc/ckan_init.d/run_rebuild_search.sh
```

CONFIGURAR UN ALIAS EN LA RED DE DOCKER PARA EL CONTENEDOR NGINX

Si, aún teniendo un nombre de dominio asignado, tu portal no puede resolver el mismo a la IP

pública del servidor, podés modificar la configuración de la red de Docker usada por Andino para mapear el nombre de dominio de tu instancia al contenedor `nginx`.

Aclaración: Esta configuración se realiza por defecto para todas las instancias de Andino desde la versión 2.5. Si tu instancia de Andino fue creada *antes de la versión 2.5*, seguramente quieras realizar estos pasos.

Para asegurarte de que la red interna de los contenedores de Docker que conforman Andino tienen la configuración necesaria para la correcta resolución de nombres, podés seguir los siguientes pasos (todos en el directorio de instalación de Andino, por ejemplo `/etc/portal`):

1. Editá el archivo `.env` y asegurate que el valor del atributo `SITE_HOST` tenga el valor del `hostname` de tu instancia de Andino, sin `http`. Si no encontrás una entrada para `SITE_HOST` en tu archivo `.env`, agregala al final. Por ejemplo debería ser `SITE_HOST=mi-andino.mi-ministerio.gob.ar`.
2. Descargá la última versión de `latest.yml`: `mv latest.yml latest.yml.bak && wget https://raw.githubusercontent.com/datosgobar/portal-andino/master/latest.yml`. Probablemente ya tengas esta misma versión si actualizaste a Andino 2.5, pero para asegurarnos de que tengas los últimos cambios necesarios para esta configuración es necesario realizar este paso.
3. Recreá el contenedor `nginx`: `docker-compose -f latest.yml up -d nginx`. Recordá que este paso puede generar algo de *downtime*, por lo que quizás sea prudente realizarlo en algún horario con poco tráfico en Andino.

VERIFICANDO QUE EL PROBLEMA FUE RESUELTO

Si el problema fue resuelto, ahora podrías realizar el procedimiento detallado en [Cómo diagnosticar si tu andino tiene el problema y deberías obtener un json como respuesta](#).

GENERACIÓN DE IMÁGENES DOCKER

- Generación de imágenes Docker

Para generar las imágenes, ejecutar la aplicación y levantarla, previamente es necesario instalar Docker y `docker-compose`:

- Instalar Docker
- Instalar docker-compose

Actualmente, el repositorio contiene 1 archivo `Dockerfile` y 2 archivos `docker-compose`

- `Dockerfile`: Se usa para generar la imagen de la aplicación
- `dev.yml`: Archivo de docker-compose para levantar los servicios necesarios y generar la imagen de la aplicación.
- `latest.yml`: Archivo de docker-compose para levantar la aplicación a su última version. (ver instalación)

Para levantar toda la aplicación, se puede correr:

```
$ docker-compose -f dev.yml up -d nginx
```

Si es la primera vez que se corre este comando, puede llegar a tardar bastante en descargar las imágenes. Una vez terminado, dejar en el puerto `localhost:80` la aplicación ejecutándose, pero antes se debe correr un comando para inicializar el desarrollo:

```
$ docker exec -it andino /etc/ckan_init.d/init_dev.sh
```

También se pueden levantar los servicios por separado de la aplicación:

- Los servicios:

```
$ docker-compose -f dev.yml up --build --abort-on-container-exit db sol redis
```

postfix

- La aplicación andino que tendrá el puerto 8080 y en el 8800 el datapusher.

```
$ docker-compose -f dev.yml up --abort-on-container-exit --build --no-deps portal
```

- Nginx que estará en el puerto 80 :

```
$ docker-compose -f dev.yml up -d --no-deps nginx
```

Eso levantará la aplicación con el directorio actual (`$PWD`) disponible dentro del directorio `/dev-app` del container.

Para acceder a la aplicación, hacer modificaciones en `runtime` , basta con correr el comando:

```
$ docker-compose -f dev.yml exec andino /bin/bash
```

INDICE

- Desarrollo
- Instalar un nuevo requerimiento en la imagen base
 - Instalar nuevo requerimiento
 - Configuración
 - Configuración propia
 - Inicio automático
 - Generación del nuevo contenedor
- Probar modificaciones y nuevas funcionalidades
 - Instalando Andino
 - Actualizando Andino
- Desarrollo
 - Instalar un nuevo requerimiento en la imagen base
 - Instalar nuevo requerimiento
 - Configuración
 - Configuración propia
 - Inicio automático
 - Generación del nuevo contenedor
 - Probar modificaciones y nuevas funcionalidades
 - Instalando Andino
 - Actualizando Andino

DESARROLLO

INSTALAR UN NUEVO REQUERIMIENTO EN LA IMAGEN BASE

Si necesitamos instalar y configurar un nuevo requerimiento para andino, lo recomendable es instalarlo en la imagen de datosgobar/portal-base. Esto permitirá mantener el build de datosgobar/portal-andino más pequeño y rápido.

Para ejemplificar esto, supondremos que queremos levantar los workers de RQ usando supervisor. Esta implementación, presente en can 2.7 , requiere levantar supervisor con una configuración especial para levantar los workers.

INSTALAR NUEVO REQUERIMIENTO

Para instalar `supervisor` en el portal base, podemos hacer uso de la tarea de `ansible` encargada de instalar los requerimientos para la aplicación. A la fecha, este código se encuentra en `dependencies.yml`

La parte que instala los requerimientos quedaría así:

```
- name: Install dependencies
  apt:
    name: "{{ item }}"
    state: present
  with_items:
    - libevent-dev
    # Otras dependencias
    - gettext
    - supervisor
```

Al agregar `supervisor`, éste será instalado cuando la generación de la imagen termine. Ahora, lo siguiente sólo sería agregar la configuración para levantar las colas de `rq`, con la configuración que trae `ckan`.

CONFIGURACIÓN

El mejor punto para agregar esta configuración es en el archivo `configure.yml`. El mismo es utilizado *después* de que `ckan` es instalado, por lo que el archivo de configuración ya está presente en el sistema.

Para eso, añadiremos una tarea más de `ansible`, usaremos el módulo `copy`. Al final de `configure.yml`, agregaremos algo como:

```
# Otras tareas

- name: Copy supervisor configuration
  copy:
    src: /usr/lib/ckan/default/src/ckan/ckan/config/supervisor-ckan-worker.conf
    dest: /etc/supervisor/conf.d/
    remote_src: yes
```

CONFIGURACIÓN PROPIA

Algo que podríamos desear es tener nuestra propia configuración de `supervisor`, que se basa en la que nos provee `ckan`. Para esto, copiamos el archivo `supervisor-ckan-worker.conf` desde dentro del contenedor y lo colocamos al lado de `production.ini.j2`, dentro del directorio `templates/ckan`. Luego, cambiamos la tarea de `ansible` para que copie nuestro archivo:

```
# Otras tareas

- name: Copy supervisor configuration
  copy:
    src: ckan/supervisor-ckan-worker.conf
    dest: /etc/supervisor/conf.d/
```

INICIO AUTOMÁTICO

Para lograr que `supervisor` se inicie automáticamente cada vez que se levante el portal, debemos correr `service supervisor restart` mientras esta se levanta, en "runtime". El mejor lugar para esto es el script que se corre por defecto: `start_ckan.sh.j2`.

En este script, agregamos los comandos que queremos *antes* de que la aplicación sea iniciada:

```
# otros comandos ...

echo "Iniciando Supervisor"
service supervisor restart

service apache2 stop
```

```
exec {{ CKAN_INIT }}/run_andino.sh
```

GENERACIÓN DEL NUEVO CONTENEDOR

Ahora, para probar estos cambios, generamos una nueva imagen de `datosgobar/portal-base` :

```
base_image="datosgobar/portal-base:test"
```

```
cd portal-base/;
```

```
docker build base_portal -t "$base_image";
```

Esto generará una imagen con el tag (o nombre) `"datosgobar/portal-base:test"`.

Para hacer una prueba rápida, podemos correr:

```
docker run --rm -it datosgobar/portal-base:test supervisord -n
```

Este comando iniciará un contenedor con supervisor corriendo. Deberíamos ver en consola los siguiente mensajes (podemos salir del contenedor usando `Ctrl+c`):

```
2018-06-22 18:55:46,593 CRIT Supervisor running as root (no user in con
fig file)
2018-06-22 18:55:46,593 WARN Included extra file "/etc/supervisor/conf.
d/supervisor-ckan-worker.conf" during parsing
2018-06-22 18:55:46,615 INFO RPC interface 'supervisor' initialized
```

Para probar esto directamente en el contenedor de `portal-andino`, debemos **generar una nueva imagen del contenedor en base a la imagen del portal base**. Para esto, en el archivo `Dockerfile` del portal, cambiamos el `FROM`, utilizando la imagen temporal que acabamos de crear:

```
FROM datosgobar/portal-base:test
```

```
# ... otros comandos
```

Ahora, generamos la nueva imagen:

```
cd portal-andino/
```

```
docker build -t portal-andino:test .
```

Luego, iniciamos la aplicación en modo desarrollo:

```
cd portal-andino/
```

```
./dev.sh setup;
```

Una vez iniciada la aplicación, entramos al contenedor y verificamos que supervisor esté corriendo:

```
cd portal-andino/
```

```
./dev.sh console
```

```
# Una vez en el contenedor
supervisorctl
```

En este momento, deberíamos ver el estado de supervisor. Para salir, usamos `Ctrl + C`.

Si comprobamos que el nuevo requerimiento está instalado y configurado, volvemos el `Dockerfile` a su estado anterior (deshaciendo el cambio en el `FROM`). Luego, debemos sacar una nueva imagen del **portal-base** y, finalmente, una nueva de **portal-andino** que se base en la anterior.

PROBAR MODIFICACIONES Y NUEVAS FUNCIONALIDADES

INSTALANDO ANDINO

Se utilizará el archivo `dev.sh` de `portal-andino` ejecutando la función `complete_up`, la cual permite levantar una instancia en base a los parámetros recibidos. Dicha función puede ser usada para

testear cambios en portal-andino, portal-andino-theme, y/o portal-base.

Los parámetros a utilizar son:

-h --help		mostrar ayuda
-a --andino_branch	VALUE	nombre del branch de portal-andino (default: master)
-t --theme_branch	VALUE	nombre del branch de portal-andino-theme (default: master o el ya utilizado)
-b --base_branch	VALUE	nombre del branch de portal-base
--nginx_ssl		activar la configuración de SSL (requiere los archivos del certificado SSL)
--nginx_host_port	VALUE	puerto a usar para HTTP
--nginx_ssl_port	VALUE	puerto a usar para HTTPS
--nginx-extended-cache		activar la configuración de caché extendida de Nginx
--ssl_key_path	VALUE	path a la clave privada del certificado SSL
--ssl crt_path	VALUE	path al certificado SSL

Todos los parámetros son opcionales. Para portal-andino y portal-andino-theme, el branch a utilizar, por default, será master en ambos casos. Para portal-base, se defaulteará a la versión que figure en el Dockerfile de portal-andino.

*Nota: Si se utiliza el nombre 'localhost' para la variable `site_host`, es posible que ocurra un error al intentar subir un archivo perteneciente a un recurso al Datastore: `Error: Proceso completo pero no se pudo enviar a result_url`. Para evitar este problema, se puede utilizar un hostname local diferente; seguir los siguientes pasos para utilizar uno nuevo: Ejecutar `vi /etc/hosts` en el host. Puede existir la línea `127.0.0.1 localhost`, y también otras parecidas; hay que escribir una nueva (es necesario asegurarse de que la IP y el hostname de la que queremos escribir sean distintos a todos los anteriores). * Escribir una línea nueva `127.0.X.1 nuevo_hostname`, reemplazando la 'X' por algún número que no esté en una de las IPs de arriba, y 'nuevo_hostname' por el que se quiera utilizar*

ACTUALIZANDO ANDINO

Para el mismo archivo, también existe una función `complete_update`, cuyo objetivo es actualizar una instancia ya levantada.

Los parámetros que recibe son exactamente los mismos que para la función de instalación.

Nota: si se desea mantener la configuración de SSL y/o de la caché extendida, es necesario especificarlo utilizando los parámetros correspondientes. No ocurre lo mismo para los archivos del certificado, puesto que son persistidos al instalar Andino.

TESTS

INDICE

- Tests de Ckan
- Probar la instalación con SSL en Vagrant (Instalando nginx)
- Instalación de andino
- Instalar y configurar nginx

Para correr los tests de la aplicación, se deben levantar todos los servicios, y luego inicializar la configuración de test.

TESTS DE CKAN

```
$ docker-compose -f dev.yml up --build -d portal
$ docker exec andino bash /etc/ckan_init.d/tests/install_solr4.sh
$ docker exec andino bash /etc/ckan_init.d/tests/install_nodejs.sh
```

```
$ docker exec andino bash -c 'su -c "bash /etc/ckan_init.d/tests/run_all_tests.sh" -l $USER'
```

PROBAR LA INSTALACIÓN CON SSL EN VAGRANT (INSTALANDO NGINX)

INSTALACIÓN DE ANDINO

Primero, debemos evitar instalar la aplicación en vagrant, ya que pasaremos un parámetro mas. Para eso, modificamos el `Vagrantfile` cambiando las líneas:

```
-INSTALL_APP = true
-UPDATE_APP = !INSTALL_APP
+INSTALL_APP = false
+UPDATE_APP = false
```

Y luego levantar la VM con `vagrant up`.

Luego entramos a la aplicación y corremos el siguiente comando:

```
sudo -E python ./install.py --error_email admin@example.com --site_host 192.168.23.10 --database_user db_user --database_password db_pass --datastore_user data_db_user --datastore_password data_db_pass --nginx_port 127.0.0.1:8000
```

Esto instalará la aplicación, pero solo la hará accesible desde `localhost:8000`.

INSTALAR Y CONFIGURAR NGINX

Luego, instalamos `nginx` en el host `sudo apt install nginx`.

Generamos los certificados locales:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/andino.key -out /etc/nginx/ssl/andino.crt
```

Creamos la configuración de nginx (*no apta para producción*) en `/etc/nginx/sites-available/001-andino.conf`:

```
upstream wsgi_andino {
    # fail_timeout=0 means we always retry an upstream even if it failed
    # to return a good HTTP response (in case the gunicorn master nukes a
    # single worker for timing out).

    server 127.0.0.1:8000 fail_timeout=0;
}

server {
    listen 80;
    #example: server_name www.datos.gob.ar datos.gob.ar;
    server_name dev.andino.gob.ar dev.andino.gob.ar;
    return 301 https://$host$request_uri;
}

server {
    listen 443;
    server_name dev.andino.gob.ar;

    ssl on;
    ssl_certificate /etc/nginx/ssl/andino.crt;
    ssl_certificate_key /etc/nginx/ssl/andino.key;

    client_max_body_size 4G;
    keepalive_timeout 5;
    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
```

```
proxy_set_header X-Forwarded-Protocol https;
proxy_redirect off;

if (!-f $request_filename) {
    proxy_pass http://wsgi_andino;
    break;
}

}
gzip on;
gzip_disable "msie6";
gzip_comp_level 6;
gzip_types text/plain text/css application/json application/x-javas
cript text/xml application/xml application/xml+rss text/javascript;

}
```

Activamos el sitio:

```
sudo rm /etc/nginx/sites-enabled/default -rf
sudo ln -s /etc/nginx/sites-available/001-andino.conf /etc/nginx/sites-
enabled/001-andino.conf

sudo systemctl restart nginx
```

Finalmente, accedemos al sitio en <http://192.168.23.10:80>, y el mismo nos debería redireccionar a la versión *https*. El explorador nos mostrará una advertencia sobre el sitio, ya que no podrá validar los certificados.