
 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	<p>GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB</p>	Código: GTSI_GU_04
		Versión:1
		Vigente desde: 01/08/2022

TABLA DE CONTENIDO

1.	INTRODUCCIÓN	2
2.	OBJETIVO	2
3.	ALCANCE	2
4.	DEFINICIONES Y CONCEPTOS	2
5.	DOCKER	3
6.	COMPONENTES DEL ECOSITEMA	4
	6.1. Docker Daemon	4
	6.2. Docker client	5
	6.3. Docker machine	5
7.	DOCKER COMPOSE	5
	7.1. Ejecución de aplicaciones con docker-compose	5
8.	ESTRUCTURACIÓN Y DOCKERIZACION	7
9.	VARIABLES DE ENTORNO	11
10.	DOCKERFILE	12
11.	COMPOSE	16
12.	ANEXO	17

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

1. INTRODUCCIÓN

El presente documento es una guía de uso del esquema de dockerización que se debe tener en los desarrollos tecnológicos con el fin de se lograr un proceso de estandarización y un esquema que facilite el despliegue en esquemas de nube bajo Kubernetes o EKS de forma integral o sobre la infraestructura de Parques Nacionales Naturales de Colombia.

2. OBJETIVO

- Establecer el esquema de estructuración de los desarrollos tecnológicos con el fin de poder ser estructurados de forma escalable
- Establecer el paso a paso de cómo pasar la estructura de los desarrollos tecnológicos al repositorio de código fuente.

3. ALCANCE

Esta guía inicia con la estructuración del código y finaliza con la subida del mismo en el repositorio de Parques Nacionales Naturales de Colombia. Aplica para los tres niveles de gestión.

4. DEFINICIONES Y CONCEPTOS

BOILERPLATE:

Código que deben incluirse en muchos lugares con poca o ninguna alteración, lo que lo convierte en una guía o plantilla para crear desarrollos a partir de un esquema base con el fin de guardar una misma estructura y lógica de forma tal que facilite su soporte y mantenimiento en el tiempo.

COMPOSER:


Administrador de paquetes a nivel de aplicación para el lenguaje de programación PHP que proporciona un formato estándar para administrar las dependencias del software PHP y las bibliotecas requeridas.

DOCKERIZACIÓN:

Conjunto de elementos que utiliza la virtualización a nivel del sistema operativo para entregar software en paquetes llamados contenedores. Los contenedores están aislados unos de otros y agrupan su propio software, bibliotecas y archivos de configuración.

DOCKER-COMPOSE:

Herramienta para definir y ejecutar aplicaciones Docker de contenedores múltiples. Se basa en un archivo plano con una sintaxis particular para configurar los servicios de su una o varias aplicaciones.


 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

DOCKERFILE:	Documento de texto que contiene todos los comandos que un usuario podría llamar en la línea de comandos para la construcción de una imagen.
ECS:	Amazon Elastic Container Service (Amazon ECS) es un servicio de administración de contenedores altamente escalable y rápido que facilita la ejecución, detención y administración de contenedores Docker en un clúster.
EKS:	Amazon Elastic Container Service para Kubernetes (Amazon EKS) es un servicio administrado que le facilita la ejecución de Kubernetes en AWS sin necesidad de instalar y operar sus propios clústeres de Kubernetes.
GITLAB:	Herramienta web que proporciona un administrador de repositorio de Git que proporciona wiki, seguimiento de problemas e integración continua / implementación continua.
KUBERNETES:	Sistema de orquestación de contenedores de código abierto para automatizar la implementación, el escalamiento y la administración de aplicaciones.
PULL REQUEST:	Comando que permite informar a otros sobre los cambios que ha introducido en una rama en un repositorio en git. Una vez que se abre una solicitud, puede analizar y revisar los posibles cambios con los colaboradores y agregar confirmaciones de seguimiento antes de que sus cambios se una a una rama determinada.
PHP-FPM:	(FastCGI Process Manager) es una implementación alternativa de PHP FastCGI con algunas características adicionales (principalmente) útiles para sitios con mucha carga.
REPOSITORIO DE CÓDIGO FUENTE:	Lugar donde se almacena un conjunto de líneas de texto que expresan, un lenguaje de programación determinado, de forma tal que puede ser versionado y administrado.

5. DOCKER

Docker es un sistema que permite encapsular todas las dependencias de software, archivos de configuración, código fuente (o binarios ejecutables) y demás componentes necesarios para ejecutar una aplicación dentro de un "contenedor" fácilmente administrable.

Los contenedores Docker son en esencia, una capa de abstracción y automatización del proceso de virtualización a nivel del mismo sistema operativo. El proceso de aislamiento de recursos se realiza utilizando características del Kernel de Linux y sistemas de archivos "unión" capaces de superponer capas a fin de obtener una estructura final de archivos.

 PARQUES NACIONALES NATURALES DE COLOMBIA	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

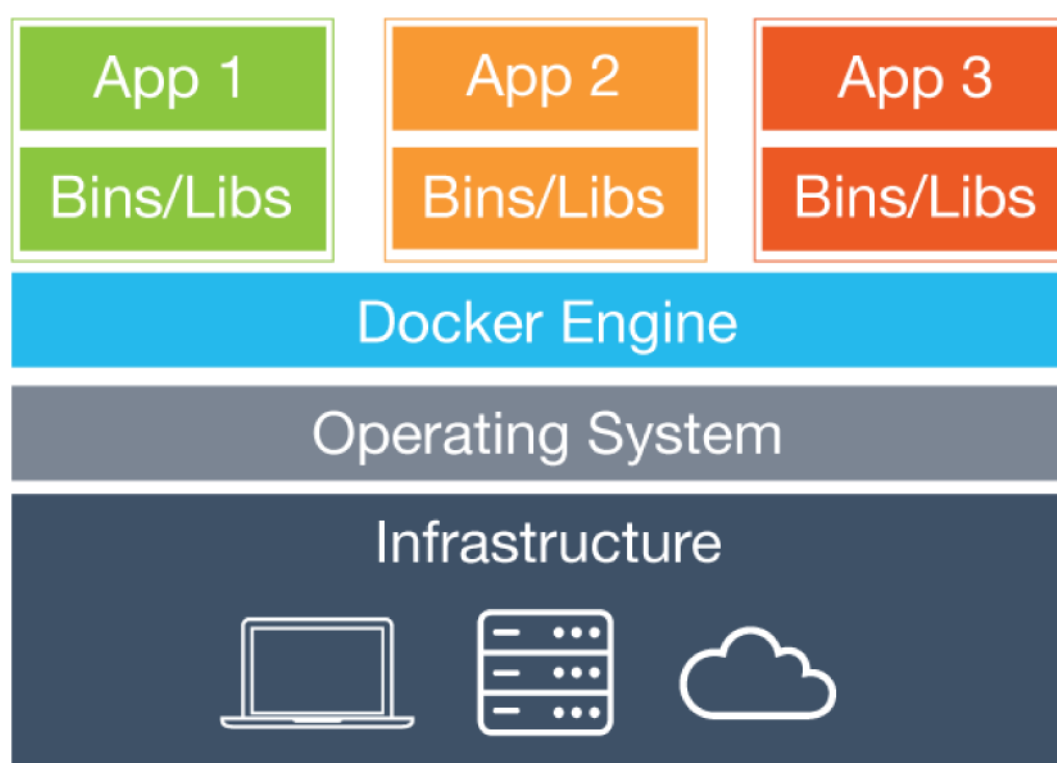


Ilustración 1. Esquema general de docker

El modelo de virtualización que ofrece el ecosistema Docker, resulta muy conveniente para el despliegue de aplicaciones "legacy", dado que la especificación permite declarar de una forma transparente todas las dependencias y el proceso de configuración necesario para ejecutar la aplicación, haciendo que cualquier nuevo miembro del equipo pueda entender el panorama completo de librerías y configuraciones necesarias para la puesta en marcha de la aplicación.

Adicionalmente, una vez "contenerizada", la aplicación puede moverse fácilmente entre servidores en cuestión de minutos, sin necesidad de realizar tareas repetitivas de instalación y configuración.


En el Anexo 1 del documento se encuentra el proceso de instalación de Docker haciendo uso de VirtualBox o con el esquema nativo de Docker dependiendo del tipo de sistema operativo.

6. COMPONENTES DEL ECOSITEMA

El ecosistema de los componentes definidos en docker son principalmente el demonio, el cliente, la máquina y el compose.

6.1. Docker Daemon

El Docker Daemon es un servicio que hace las veces de runtime para la ejecución de las aplicaciones "contenerizadas". Este proceso se encarga de administrar las imágenes y contenedores que se estén ejecutando en una máquina. Este servicio arranca normalmente de la misma forma que arrancan otros servicios en el sistema al momento del arranque.

 PARQUES NACIONALES NATURALES DE COLOMBIA	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

6.2. Docker client

El cliente de docker corresponde a una aplicación de línea de comando que se utiliza para la comunicación con el Docker daemon. Cada vez que se escribe el comando docker ps o docker run se está utilizando el cliente.

6.3. Docker machine

Corresponde a otra aplicación de línea de comando que permite aprovisionar máquinas que ejecutarán instancias de un **docker daemon** y seleccionar a cuál máquina (**docker daemon**) específicamente se quiere "hablar" en un momento específico.

7. DOCKER COMPOSE

Programa de línea de comando que permite ejecutar aplicaciones definidas en archivos docker-compose.yml. Estos archivos permiten declarar la estructura de una o varias aplicaciones y la relación entre ellas, a fin de simplificar la forma en que se lanzan aplicaciones desde la línea de comando.

7.1. Ejecución de aplicaciones con docker-compose

El ecosistema docker permite ejecutar aplicaciones complejas en donde distintos componentes de una arquitectura se dividen en contenedores mediante los cuales puede hacerse "composiciones" de contenedores que prestan servicios entre sí. Por ejemplo, una aplicación que consiste en una base de datos MySQL, y un servidor Nginx o Apache que ejecuta una aplicación PHP, normalmente se configuran como dos contenedores distintos, y se vinculan entre sí mediante "links".

Si bien la línea de comando del **cliente docker** es descriptiva en cuanto a los argumentos que recibe, en ocasiones la cantidad de parámetros necesarios para correr la aplicación es demasiado larga para ser conveniente. Por ejemplo, para lanzar la aplicación una aplicación con características como las descritas, los argumentos se verían de la siguiente forma en la línea de comando:

Aplicación MySQL


```
docker run -d --name MySQL -v /media/data/mysql:/var/lib/mysql -p 3306:3306 mysql:5.6
```

Aplicación Apache + PHP

```
docker run -d --name AppPHP -v /media/data/app:/var/data -p 80:80 --link mysql:mysql php-apache
```

El ejemplo anterior ejecuta en el background (run -d) un contenedor llamado MySQL (--name MySQL) exponiendo el servidor MySQL en el puerto 3306 (-p 3306:3306) de la máquina huésped, montando un directorio de datos (-v /media/data/mysql:/var/lib/mysql) , a partir de la imagen pública mysql:5.6.

El comando anterior descarga del repositorio oficial de docker, la imagen mysql:5.6 y ejecutará la aplicación dentro del contenedor, en un entorno totalmente aislado a manera de proceso en la máquina huésped.

 PARQUES NACIONALES NATURALES DE COLOMBIA	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

A continuación, se crea un contenedor llamado AppPHP que corre la aplicación apache y al que se "vincula" el contenedor MySQL (--link mysql:mysql). Este nuevo contenedor tendrá la capacidad de conectarse al contenedor anterior, utilizando para ello una serie de variables de entorno definidas al momento de establecer el enlace y adicionalmente podrá resolver directamente el nombre mysql a una dirección IP que corresponde con la dirección del contenedor MySQL.

Correr sin embargo varias aplicaciones recordando los detalles de configuración y parámetros de línea de comando de cada una es inconveniente, **docker-compose** resuelve este problema.

A fin de realizar "composiciones" de aplicaciones, con **docker-compose**, basta crear un archivo llamado **docker-compose.yml**. En este archivo se describen en formato YAML las aplicaciones y sus relaciones. **docker-compose** que se encarga de ejecutar cada una de las aplicaciones en orden, de manera que los enlaces o "links" entre ellas sean consistentes. Por ejemplo, el escenario anterior puede describirse en un archivo como el siguiente:

MySQL:

```
image: mysql:5.6
restart: always
expose:
  • 3306
volumes:
  • /media/data/mysql:/var/lib/mysql
```


AppPHP:

```
build: ./myapp
restart: always
expose:
  • 80
links:
  • MySQL:mysql
volumes:
  • /media/data/app:/var/data
```

Con esta configuración, es posible ejecutar lo siguiente:

```
docker-compose up -d AppPHP
```

El comando anterior se encargará de crear un contenedor MySQL y posteriormente un contenedor AppPHP, tal como lo establece AppPHP. El hecho de que MySQL se encuentre bajo la sección links implica que, para crear una instancia consistente de la aplicación, es necesario vincular primero dicha instancia del contenedor a la instancia del contenedor AppPHP.

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

Es importante tener en cuenta que, si se reinicia el contenedor MySQL, es necesario también reiniciar el contenedor AppPHP a fin de que el "link" o vínculo entre las aplicaciones se mantenga consistente. En este caso particular ambos contenedores exponen el puerto declarado en la directiva "expose" sobre un puerto arbitrario de la máquina huésped, ideal para lanzar composiciones de aplicaciones.

8. ESTRUCTURACIÓN Y DOCKERIZACION

La entidad ha definido que todo desarrollo debe estar desplegado en contenedores (Docker) de forma tal que pueda ser utilizado en cualquiera de los esquemas de despliegue en Nube o para el despliegue bajo la infraestructura.

Para casos de desarrollos nuevos a la medida de la entidad se deben estructurar con algunos de los esquema de boilerplate definidos que se encuentran en el repositorio de código de Parques Nacionales para el backend (Laravel framework) y frontend (Angular JS) (<https://gitlab.com/parques-nacionales-naturales-de-colombia/boilerplates>)

1. El desarrollo de aplicaciones nuevas debe estar construidas con base en los repositorios de boilerplate.
2. Todo desarrollo debe mantenerse al día con las actualizaciones que se incluyen en el esquema base del boilerplate.
3. En caso tal de encontrar problemas sobre el funcionamiento del boilerplate este debe ser notificado al administrador del respectivo repositorio.
4. Todo componente que se desee desarrollar que no exista en el boilerplate debe ser incluido en el mismo de forma tal que pueda ser incluido en el desarrollo.
5. Cualquier mejora o sugerencia debe ser notificada al administrador para poder adicionar la funcionalidad al boilerplate.

En caso tal que el desarrollo de la entidad esté en curso se debe tener en cuenta:


1. El grupo TIC debe tener seguimiento y validación del código fuente a través de un esquema de Pull request o en su defecto revisión sobre el código fuente.

En caso tal que el desarrollo de la entidad esté desarrollado por un tercero:

1. Se debe tener un seguimiento y validación del código fuente en caso de ser posible.
2. Se debe hacer uso de las últimas versiones estables para su implementación.
3. Se debe tener esquemas que aseguren la interoperabilidad.
4. Se debe trabajar conjuntamente con el proveedor en la dockerización del desarrollo.

En general para cualquiera de los casos anteriormente mencionados se debe tener en cuenta:

1. El esquema de arquitectura e interoperabilidad debe ser construido conjuntamente con el grupo TIC.

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

2. El desarrollo debe estar estructurado en el esquema de carpetas definidos en el marco con el grupo TIC.
3. El desarrollo debe seguir los lineamientos definido para el grupo TIC y deber contener la lista de chequeo de HandOver que ha sido definida para los Sistemas de Información.

Se debe tener en cuenta la siguiente estructura para la carpeta de desarrollos:

La carpeta /etc contiene toda la información relacionada con la configuración del servidor de aplicaciones, php u otros módulos. En la carpeta src se encuentra el código fuente propio de la aplicación y en la carpeta tmp se manejan los templates que son usados provisionalmente para la configuración del proyecto durante la construcción de la imagen.

Name
etc/apache2/sites-available
src
tmp
.dockerignore
.gitignore
Dockerfile
docker-entrypoint.sh


Ilustración 2 Esquema ejemplo de las carpetas

Para cada uno de los servidores de aplicaciones se tiene una implementación a la fecha de construcción de este manual predefinidos para:

- Tomcat
- Apache
- Nginx

Los esquemas que están contruidos para Parques Nacionales Naturales de Colombia se encuentran disponibles en github de la entidad y puede ser usado de forma abierta por cualquier entidad del sector.

El esquema bajo el cual está estructura se componen los directorio está dado por:

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

Etc: carpeta que tiene la configuración respectiva del código fuente bajo el cual se ejecuta la herramienta. En esta carpeta se tiene información del servidor de aplicaciones, configuración propia del lenguaje y otros. Para este ejemplo se realiza la configuración de un apache2.

Es importante tener en cuenta que cada desarrollo utiliza componentes y servicios propios de cada desarrollo. Este manual actúa como guía pero no define qué se deba desarrollar con ciertos componentes, esta estará sujeta al tipo de desarrollo tecnológico y las necesidades propias del proyecto que se discutirán en la definición de la arquitectura de la herramienta.

En caso de tener un esquema de apache en la carpeta de sites-enabled

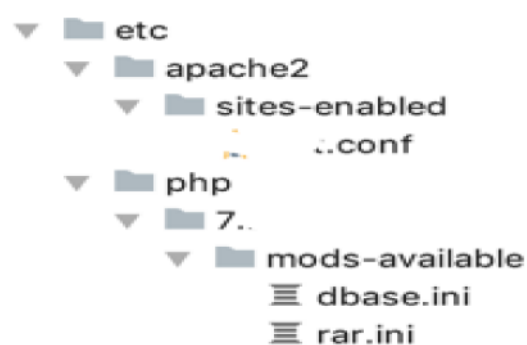


Ilustración 3 Ejemplo del esquema de directorio de etc

En la siguiente carpeta se tiene el esquema general de la definición del VirtualHost, cualquier información que se quiera colocar se debe realizar el documento interno:

```

ServerName ${SERVER_NAME}
<Directory "${DATA_DIRECTORY}">
# Important for security, prevents someone from
# uploading a malicious .htaccess
AllowOverride None

SetHandler none
SetHandler default-handler

Options -ExecCGI
RemoveHandler .cgi .php .php3 .php4 .php5 .phtml .pl .py .pyc .pyo
<Files *>
SetHandler none
SetHandler default-handler

Options -ExecCGI
# @TODO: deprecated
#php_flag engine off
RemoveHandler .cgi .php .php3 .php4 .php5 .phtml .pl .py .pyc .pyo
</Files>
</Directory>
<VirtualHost *:80>
ServerName ${SERVER_NAME}
DocumentRoot /var/www
ServerAdmin ${SERVER_ADMIN_EMAIL}


ErrorLog /dev/stdout
CustomLog /dev/null combined
LogLevel ${LOG_LEVEL}

Header unset ETag
FileETag None

<Directory /var/www >
AllowOverride None
DirectoryIndex index.php
Require all granted
RewriteEngine On
RewriteCond "%{REQUEST_FILENAME}" -s [OR]
RewriteCond "%{REQUEST_FILENAME}" -l [OR]
RewriteCond "%{REQUEST_FILENAME}" -d
RewriteRule "\.(js|ico|css|gz|html|xml)$" "index.php" [NC,L]
</Directory>
</VirtualHost>

```

Ilustración 4 Virtualhost

 PARQUES NACIONALES NATURALES DE COLOMBIA	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

Es importante tener en cuenta que todo debe ser definido en términos de variables de entorno las cuales deben estar definidas en el docker-compose del proyecto. Por defecto el proyecto debe tener el DocumentRoot¹ /var/www es la única ruta que se debe dejar de forma explícita. El tema de docker-compose será objeto del manual para más adelante.

El esquema de errores de las aplicaciones debe estar en un log en el directorio particular pero en lo esquema de producción debe estar en un esquema de STDOUT y/o STDERR para poder integrarlo a esquema de seguimiento de errores.

Importante que en caso de necesitar incluir los archivos .ini se deben incluir en el mods-available del php

En la carpeta src/ se tiene todo el código fuente de la herramienta

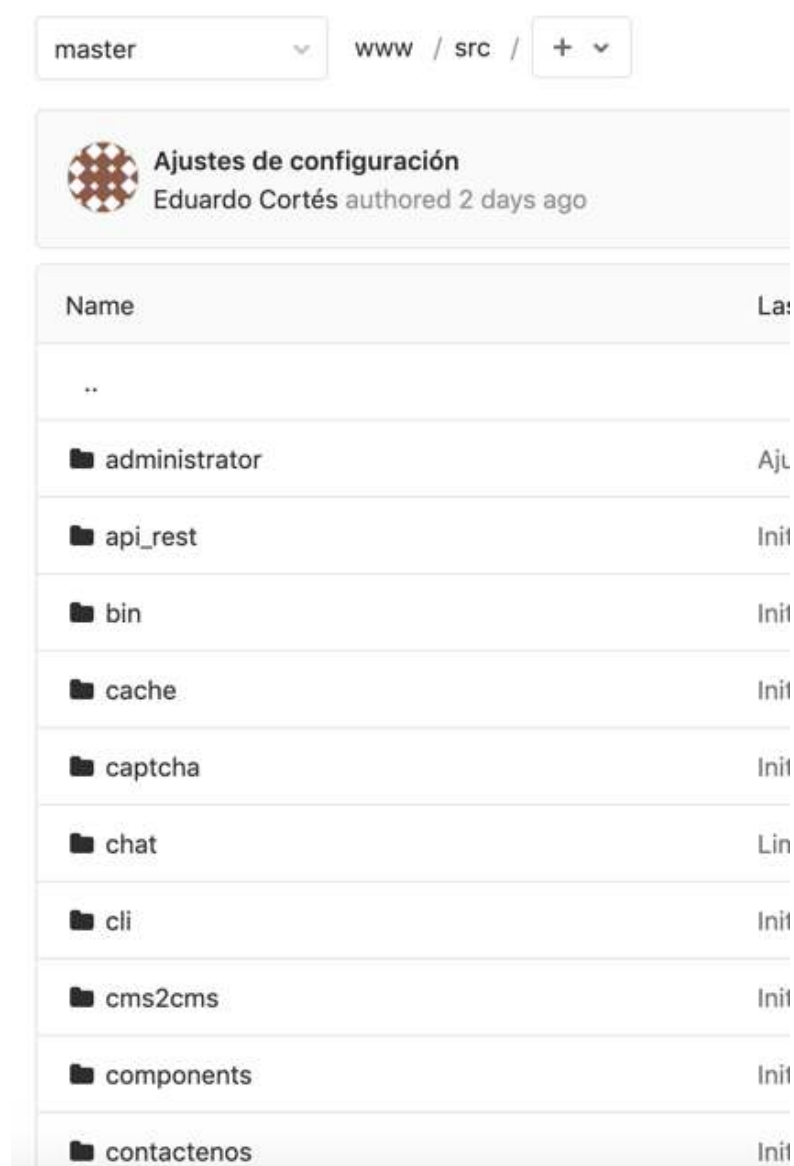



Ilustración 5 Código que va en la carpeta src

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

9. VARIABLES DE ENTORNO

En la carpeta tmp/ se tiene un archivo donde están definidas todas las variables de entorno, teniendo en cuenta que se hace uso de php-fpm es necesario realizar el cargue de estas variables de entorno al esquema del php-fpm.

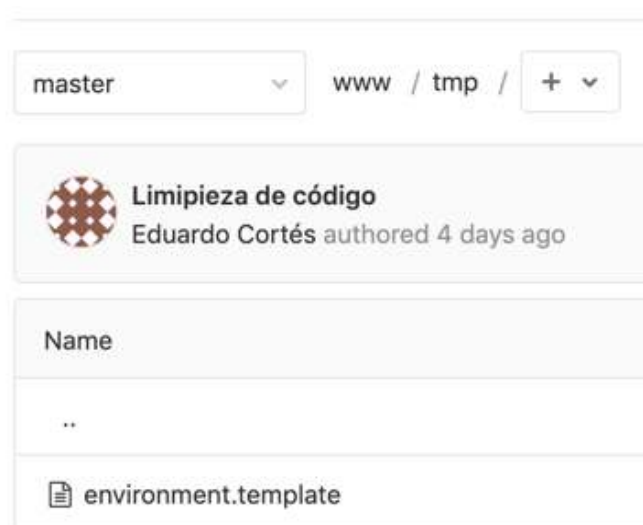


Ilustración 6 Template variable para php-fpm


El esquema de template debe contener las variables definidas en el docker-compose. El tema de docker-compose será objeto del manual para más adelante.

```

environment.template 873 Bytes
1 ; Dockerfile
2
3 env[STORAGE_PATH] = ${DOLLAR}STORAGE_PATH
4 env[TMP_STORAGE_PATH] = ${DOLLAR}TMP_STORAGE_PATH
5
6
7 ; docker-compose.yml
8
9 env[VIRTUAL_HOST] = ${DOLLAR}VIRTUAL_HOST
10
11 env[DEBUG_MODE] = ${DOLLAR}DEBUG_MODE
12 env[LOG_LEVEL] = ${DOLLAR}LOG_LEVEL
13
14 env[PRINCIPAL_DB_HOST] = ${DOLLAR}PRINCIPAL_DB_HOST
15 env[PRINCIPAL_DB_USERNAME] = ${DOLLAR}PRINCIPAL_DB_USERNAME
16 env[PRINCIPAL_DB_PASSWORD] = ${DOLLAR}PRINCIPAL_DB_PASSWORD
17 env[PRINCIPAL_DB_DATABASE] = ${DOLLAR}PRINCIPAL_DB_DATABASE
18
19 env[CHAT_DB_HOST] = ${DOLLAR}CHAT_DB_HOST
20 env[CHAT_DB_USERNAME] = ${DOLLAR}CHAT_DB_USERNAME
21 env[CHAT_DB_PASSWORD] = ${DOLLAR}CHAT_DB_PASSWORD
22 env[CHAT_DB_DATABASE] = ${DOLLAR}CHAT_DB_DATABASE
23
24 env[SINAP_DB_HOST] = ${DOLLAR}SINAP_DB_HOST
25 env[SINAP_DB_USERNAME] = ${DOLLAR}SINAP_DB_USERNAME
26 env[SINAP_DB_PASSWORD] = ${DOLLAR}SINAP_DB_PASSWORD
27 env[SINAP_DB_DATABASE] = ${DOLLAR}SINAP_DB_DATABASE

```

Ilustración 7 Archivo template

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

10. DOCKERFILE

El archivo de Dockerfile, es el archivo que define cómo se construye la imagen del esquema de docker, es el set de instrucciones que permite que se ensamble el proyecto de forma tal que luego a partir de este ensamble (build) a partir del cual se crean las instancias del proyecto.

```

1 FROM servicioswebminambiente/apache2_php:7.4
2
3 ENV STORAGE_PATH /var/storage
4 ENV TMP_STORAGE_PATH /var/tmp_storage
5 ENV SHARED_TMP_STORAGE_PATH /var/shared_tmp_storage
6
7 # Instala paquetes del SO
8 RUN apt-get update \
9   && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
10  php-mbstring php-xml php-zip \
11  # Hace limpieza
12  && apt-get clean && rm -rf /var/tmp/* /var/lib/apt/lists/* /tmp/*
13
14 # Instala paquetes de Composer
15 #RUN mkdir -p /var/www
16 #COPY src/composer.json /var/www/
17 #WORKDIR /var/www
18 #RUN composer install
19
20 # Habilita módulos de Apache2
21 #RUN a2enmod xxx
22
23 # Configura módulos de PHP
24 RUN rm -f /etc/php/7.4/fpm/conf.d/*opcache* /etc/php/7.4/fpm/conf.d/*xdebug* \
25  # Configuración de PHP
26  && sed -i 's/output_buffering = 4096/output_buffering = 0ff/' /etc/php/7.4/fpm/php.ini \
27  && sed -i 's;session\.save_path = "/var/lib/php/sessions"@session.save_path = "/var/shared_tmp_storage/sessions"@' /etc/php/7.4/fpm/php.ini \
28  && sed -i 's;/date\.timezone = /date.timezone = America/Bogota/' /etc/php/7.4/fpm/php.ini \
29  && sed -i 's;/max_execution_time = ./max_execution_time = 60/' /etc/php/7.4/fpm/php.ini \
30  && sed -i 's;/max_input_time = ./max_input_time = 60/' /etc/php/7.4/fpm/php.ini \
31  && sed -i 's;/memory_limit = ../memory_limit = 1024M/' /etc/php/7.4/fpm/php.ini \
32  && sed -i 's;/post_max_size = .M/post_max_size = 50M/' /etc/php/7.4/fpm/php.ini \
33  && sed -i 's;/upload_max_filesize = .M/upload_max_filesize = 50M/' /etc/php/7.4/fpm/php.ini \
34  && sed -i 's;/short_open_tag = 0ff/short_open_tag = 0n/' /etc/php/7.4/fpm/php.ini \
35  && sed -i 's;/short_open_tag = 0ff/short_open_tag = 0n/' /etc/php/7.4/cli/php.ini
36
37 # Copia archivos del sistema de archivos local al contenedor
38 COPY etc /etc
39 COPY tmp /tmp
40 COPY src /var/www
41
42 WORKDIR /var/www
43
44 # Ejecución de contenedor
45 COPY docker-entrypoint.sh /
46 RUN chmod u+x /docker-entrypoint.sh
47
48 VOLUME ["/storage"]
49 ENTRYPOINT ["/docker-entrypoint.sh"]
50 CMD ["supervisord", "-c", "/etc/supervisord.conf"]


```

Ilustración 8 Archivo dockerfile

El esquema del DockerFile (<https://docs.docker.com/engine/reference/builder/>) el esquema comienza con el FROM, es a partir de que imagen se construye la imagen actual, uno puede basar su imagen en imágenes que tienen predefinidos esquemas de instalación, en este caso se basa en las imágenes que han sido creadas en dockerhub por Parques Nacionales.

Cada una de estas imágenes tiene un tag que es la versión específica que se instala dependiendo de las necesidades.

En esa medida se define el FROM y la versión seguida de los “:” dependiendo del tag respectivo

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

Luego de eso se realiza una definición del donde va a quedar el esquema de STORAGE se tiene por lo general

```
ENV STORAGE_PATH /var/storage
ENV TMP_STORAGE_PATH /var/tmp_storage
ENV SHARED_TMP_STORAGE_PATH /var/shared_tmp_storage
```

Ilustración 11 Almacenamiento

1. Activos de información, es decir aquellos archivos donde se tiene los pdf y demás archivos que son cargados por los usuarios (en una fase posterior este esquema debería estar implementado con Google Storage)
2. Esquema sobre el cual reposan las sesiones y otros archivos temporales. En esquemas de alta disponibilidad estos deben estar en un volumen compartidos
3. Esquema sobre los cuales se tienen archivos temporales que usa la herramienta para mejorar desempeño, estos archivos no deben estar compartidos en un esquema de alta disponibilidad simplemente debe estar en un disco de alta velocidad para tener buenos desempeños en la herramienta

```
# Instala paquetes del SO
RUN apt-get update \
  && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
  php-mbstring php-xml php-zip \
  # Hace limpieza
  && apt-get clean && rm -rf /var/tmp/* /var/lib/apt/lists/* /tmp/*

# Instala paquetes de Composer
#RUN mkdir -p /var/www
#COPY src/composer.json /var/www/
#WORKDIR /var/www
#RUN composer install
```


Ilustración 12 Instalación de paquetes

Luego se realiza la instalación de aquellos paquetes que no están instalados por defecto en la versión del FROM de la imagen.

En la mayoría de los desarrollos se debería tener un esquema de composer, para el manejo de paquetes de PHP, el composer estaría a cargo de la descarga de las librerías necesarias para el funcionamiento de la herramienta.

```
# Instala paquetes de Composer
WORKDIR /var/www/html/composer.json
RUN composer install \
  && rm -f composer.*
```

Ilustración 13 Instalación del esquema de composer

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

Se recomienda para desarrollos que tengan más de 3 años tener una copia de la carpeta vendor generada en el proceso de instalación con el fin de asegurar que las librerías tiene una copia en el repositorio local de la entidad.

La herramienta no debe hacer copiar del archivo php.ini se debe hacer la configuración respectiva a partir de SED

```
# Configura modulos de PHP
RUN rm -f /etc/php/7.4/fpm/conf.d/*opcache* /etc/php/7.4/fpm/conf.d/*xdebug* \
# Configuración de PHP
&& sed -i 's/output_buffering = 4096/output_buffering = Off/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's;session\.save_path = "/var/lib/php/sessions"@session.save_path = "/var/shared_tmp_storage/sessions"@' /etc/php/7.4/fpm/php.ini \
&& sed -i 's;/date\.timezone = /date.timezone = America/Bogota/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's/max_execution_time = ../max_execution_time = 60/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's/max_input_time = ../max_input_time = 60/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's/memory_limit = ..M/memory_limit = 1024M/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's/post_max_size = .M/post_max_size = 50M/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's/upload_max_filesize = .M/upload_max_filesize = 50M/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's/short_open_tag = Off/short_open_tag = On/' /etc/php/7.4/fpm/php.ini \
&& sed -i 's/short_open_tag = Off/short_open_tag = On/' /etc/php/7.4/cli/php.ini

# Copia archivos del sistema de archivos local al contenedor
```

Ilustración 14 Actualización de variables de PHP del .ini

Finalmente se realiza

```
# Copia archivos del sistema de archivos local al contenedor
COPY etc /etc
COPY tmp /tmp
COPY src /var/www

WORKDIR /var/www


# Ejecución de contenedor
COPY docker-entrypoint.sh /
RUN chmod u+x /docker-entrypoint.sh

VOLUME ["${STORAGE_PATH}"]
ENTRYPOINT ["/docker-entrypoint.sh"]
CMD ["supervisord", "-c", "/etc/supervisor/supervisord.conf"]
```

Ilustración 15 Copia de carpetas y entrypoint

Sin embargo, cada proyecto es particular y no significa que se deba hacer literalmente cada desarrollo tiene sus respectivas implementaciones para lo cual cada caso debe ser evaluado.

El archivo del entrypoint define todo lo que se deben pasar de la imagen del contenedor a la instancia que se ejecute sobre el desarrollo tecnológico, este archivo debe validar que todas las variables de entorno mínimas se encuentren definidas, que se validen la creación de los directorios y se coloquen los permisos respectivos para el usuario. Finalmente para apoyar el desarrollo se da la opción de incluir el esquema de XDEBUG para los desarrolladores

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	<p>GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB</p>	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

```
#!/bin/bash
set -e

if [ "$1" = 'supervisord' ]; then
# Variables de entorno obligatorias
: ${VIRTUAL_HOST:?"debe estar definido para ejecutar esta aplicacion"}

: ${DEBUG_MODE:?"debe estar definido para ejecutar esta aplicacion"}
: ${LOG_LEVEL:?"debe estar definido para ejecutar esta aplicacion"}

: ${PRINCIPAL_DB_HOST:?"debe estar definido para ejecutar esta aplicacion"}
: ${PRINCIPAL_DB_USERNAME:?"debe estar definido para ejecutar esta aplicacion"}
: ${PRINCIPAL_DB_PASSWORD:?"debe estar definido para ejecutar esta aplicacion"}
: ${PRINCIPAL_DB_DATABASE:?"debe estar definido para ejecutar esta aplicacion"}

: ${CHAT_DB_HOST:?"debe estar definido para ejecutar esta aplicacion"}
: ${CHAT_DB_USERNAME:?"debe estar definido para ejecutar esta aplicacion"}
: ${CHAT_DB_PASSWORD:?"debe estar definido para ejecutar esta aplicacion"}
: ${CHAT_DB_DATABASE:?"debe estar definido para ejecutar esta aplicacion"}

: ${SINAP_DB_HOST:?"debe estar definido para ejecutar esta aplicacion"}
: ${SINAP_DB_USERNAME:?"debe estar definido para ejecutar esta aplicacion"}
: ${SINAP_DB_PASSWORD:?"debe estar definido para ejecutar esta aplicacion"}
: ${SINAP_DB_DATABASE:?"debe estar definido para ejecutar esta aplicacion"}

# Crea los directorios iniciales
mkdir -p "${TMP_STORAGE_PATH}/logs"
mkdir -p "${SHARED_TMP_STORAGE_PATH}/sessions"
mkdir -p "${TMP_STORAGE_PATH}/joomla/logs"
mkdir -p "${TMP_STORAGE_PATH}/joomla/tmp"
mkdir -p "${TMP_STORAGE_PATH}/sinap/logs"
mkdir -p "${TMP_STORAGE_PATH}/sinap/tmp"

# Establece permisos sobre los directorios
chown www-data:www-data "${TMP_STORAGE_PATH}/logs"
chown www-data:www-data "${SHARED_TMP_STORAGE_PATH}/sessions"
chown www-data:www-data "${TMP_STORAGE_PATH}/joomla/logs"
chown www-data:www-data "${TMP_STORAGE_PATH}/joomla/cache"
chown www-data:www-data "${TMP_STORAGE_PATH}/joomla/tmp"
chown www-data:www-data "${TMP_STORAGE_PATH}/joomla/administrator/cache"
chown www-data:www-data "${TMP_STORAGE_PATH}/sinap/logs"
chown www-data:www-data "${TMP_STORAGE_PATH}/sinap/cache"
chown www-data:www-data "${TMP_STORAGE_PATH}/sinap/tmp"
chown www-data:www-data "${TMP_STORAGE_PATH}/sinap/administrator/cache"

chown www-data:www-data "/var/www/chat/_log"
chown www-data:www-data "/var/www/directorio_entidades/app/cache"
chown www-data:www-data "/var/www/directorio_entidades/app/logs"
chown www-data:www-data "/var/www/nchat/cache"
chown www-data:www-data "/var/www/nchat_2/cache"

if [ ${DEBUG_MODE} = '1' ]; then
# Modo de desarrollo
# Habilita el modulo de xdebug
phpenmod xdebug
else
# Modo produccion
# Habilita el modulo de opcache
phpenmod opcache
fi

export DOLLAR='$'


# Configuración de VirtualHosts
envsubst < /etc/apache2/sites-available/sitio.template > /etc/apache2/sites-enabled/sitio.conf

# Variables de entorno
echo "" >> /etc/php/7.4/fpm/pool.d/www.conf
envsubst < /tmp/environment.template >> /etc/php/7.4/fpm/pool.d/www.conf
rm -f /tmp/environment.template

# @TODO: descomentar esta linea para que salgan los logs en Google
#tail -Fn 0 ${TMP_STORAGE_PATH}/logs/* &
exec supervisord -c /etc/supervisor/supervisord.conf
else
exec "$@"
fi
```

Ilustración 16 Archivo de entrypoint

Para poder realizar el esquema de instanciación y construcción de la imagen se debe tener un archivo compose (<https://gitlab.com/otic1/dockercompose1>), cada desarrollo tiene un archivo que se encuentra en el repositorio.

 PARQUES NACIONALES NATURALES DE COLOMBIA	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

11. COMPOSE

Los desarrollos web que estén bajo docker deben estar siempre acompañados de un docker-compose este archivo permite a los desarrolladores poder ejecutar en un ambiente local el desarrollo tecnológico sobre el cual estén trabajando. Para esto es necesario 1) contar con acceso al repositorio <https://gitlab.com/otic1/docker-compose/develop> este repositorio contiene la configuración de cada proyecto. 2) en caso de no tener acceso se debe solicitar acceso al administrador 3) una vez se cuente con acceso al repositorio se deben colocar todas las variables de entorno que sean necesarias para poder ejecutar el contenedor, la información asociada a cada variable de entorno son de ejemplo y debe tener en cuenta que las mismas no necesariamente serán las mismas del ambiente de producción, este debe realizar los ajustes que sean necesarios a su código fuente 4) una vez finalice la actualización de la variables debe hacer push de los cambios para poder incluirlas en el branch master.

Para este proceso es importante que se realice un branch propio y solicite un pull request de tal forma que el administrador valide que está colocando las variables de entorno necesarias con respecto a lo que tiene definido en el código fuente del proyecto.

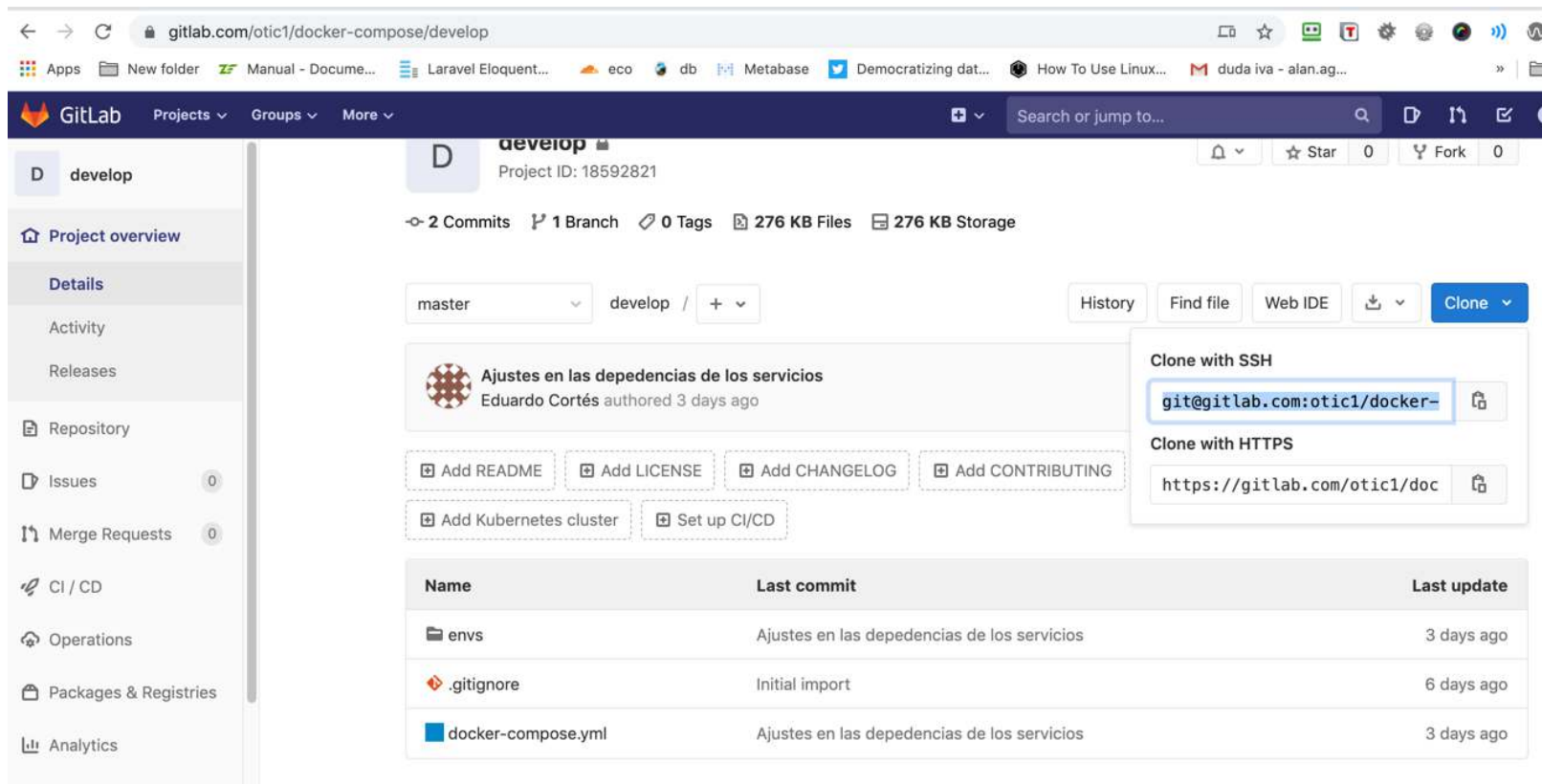



Ilustración 17 Repositorio de docker-compose

Una vez finalice la inclusión de las variables de entorno el archivo debe quedar con algo similar a:

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

```

env_file: ./envs/cronjobs.dev.env

educacion-ambiental:
  build: ./educacion-ambiental
  depends_on:
    - mysql56
  expose:
    - "80"
  networks:
    - mads-net
  volumes:
    - ./storage/educacion-ambiental:/var/storage
    - ./tmp_storage/educacion-ambiental:/var/tmp_storage
    - ./educacion-ambiental/src:/var/www
  environment:
    - VIRTUAL_HOST=educacion-ambiental.local
  env_file: ./envs/educacion-ambiental.dev.env

entrenamiento:
  build: ./entrenamiento
  depends_on:
    - mysql56
  ports:
    - "80:80"
  networks:
    - mads-net
  volumes:
    - ./storage/entrenamiento:/var/storage
    - ./tmp_storage/entrenamiento:/var/tmp_storage
    - ./entrenamiento/src:/var/www
  environment:
    - VIRTUAL_HOST=entrenamiento.minambiente.gov.co
  env_file: ./envs/entrenamiento.dev.env

escueladeformacion:
  build: ./escueladeformacion
  depends_on:
    - mysql56
  expose:
    - "80"
  networks:

```

Ilustración 18 Variables de entorno

12. ANEXO


Este es un procedimiento de ejemplo de cómo realizar el proceso de instalación en un ambiente local de docker

INSTALACIÓN NATIVA DEL DOCKER

Para la instalación nativa de docker en el sistema operativo se deben cumplir con los requerimientos mínimos de docker.

En el siguiente link se establecen los sistemas operativos para los cuales es compatible el uso de docker <https://docs.docker.com/engine/install/>

Para realizar el proceso de instalación para ambiente Mac siga los pasos definidos en el siguiente link <https://docs.docker.com/docker-for-mac/install/>

 <p>PARQUES NACIONALES NATURALES DE COLOMBIA</p>	GUIA ESQUEMA DE DOCKERIZACIÓN DE LOS DESARROLLOS WEB	Código: GTSI_GU_04
		Versión: 1
		Vigente desde: 01/08/2022

FECHA DE VIGENCIA VERSIÓN ANTERIOR	VERSIÓN ANTERIOR	MOTIVO DE LA ACTUALIZACIÓN

CRÉDITOS		
Elaboró	Nombre	Adriana Lorena Bernal – Sandra Milena Gómez - Claudia Patricia Berrocal - Alan Agua
	Cargo	Profesional Contratista Grupo de Tecnologías de la Información y las Comunicaciones
	Fecha	11/07/2022
Revisó	Nombre	Carlos Arturo Sáenz Barón
	Cargo	Coordinador Grupo de Tecnologías de la Información y las Comunicaciones
	Fecha:	11/07/2022
Aprobó	Nombre	Carlos Arturo Sáenz Barón
	Cargo	Coordinador Grupo de Tecnologías de la Información y las Comunicaciones
	Fecha:	22/07/2022