
Universidad
Tecmilenio®

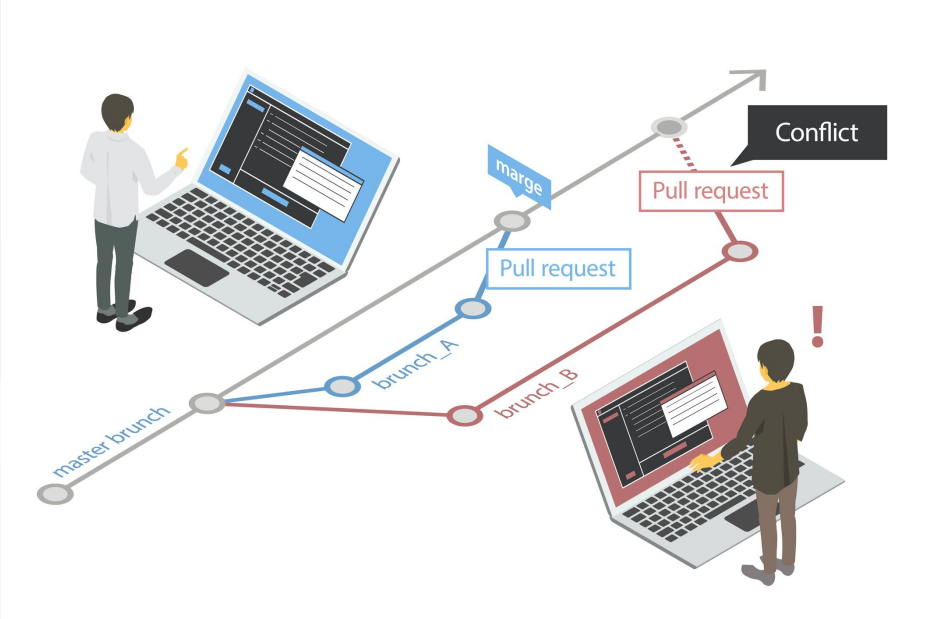
Universidad Tecmilenio

Especialización en DevOps

Git/GitHub

Semana 3

Introducción



Cuando se programa una aplicación, es esencial llevar un control de lo que se está haciendo y publicando cada vez que se edita o se agregan nuevas cosas a un código ya terminado o en desarrollo.

En esos casos, es importante guardar una versión del código por si algo sale mal con el siguiente, se pueda regresar a ese punto de partida. Para eso se creó **Git**, un repositorio que sirve para realizar un seguimiento del proyecto que se realice.

Explicación

Git

Un sistema de control de versiones que contiene la copia del historial completo de un proyecto y no solo del estado actual del mismo (GitHub, 2020).

Modificar o actualizar archivos de un proyecto.

Incluir archivos en el área de preparación (*git.add*).

Generar toma instantánea y almacenar en bases de datos (*git commit*).

GitHub

Una plataforma donde se puede subir, almacenar o respaldar una copia de un repositorio Git.

Trabajo colaborativo.

Gestiona información.

Rastreabilidad de cambios.

Explicación

Repositorios

Ubicación donde se almacenarán los proyectos, los archivos incluidos en ellos, así como los cambios que se realizan.

Repositorio local	Es solo una ubicación de archivo que reside en el sistema. Cuando se confirma (*commit*) algún cambio en un proyecto, se crea una versión, también llamada instantánea, en el repositorio local.
Repositorio remoto	Se encuentra en algún lugar de la nube, en una máquina remota. Esto es muy importante cuando trabajas con varias personas, ya que ahí se compartirán los avances o cambios del proyecto.

Crea un directorio .git en el archivo. → Crea un archivo home.html. → Haz clic en el + para agregar localmente. → Crea en GitHub un repositorio y alójalo en el código local.

Explicación

Comandos básicos

Local

Git init
Git add
Git commit
Git status
Git config
Git Branch
Git merge

Remota

Git remote
Git clone
Git pull
Git push
Git log

```
# Sintaxis
# git pull <rama> <URL/nombre remoto>


# Ejemplo usando el nombre remoto del repositorio
git pull origin staging

From usuario.git.devopstecmi.com:/usuario/devopstecmilocal
 * branch            staging    -> FETCH_HEAD
 * [new branch]      staging    -> origin/staging
Already up-to-date.

# Ejemplo usando el URL
git pull git@usuario.git.devopstecmi.com:/usuario/devopstecmilocal.git staging

From usuario.git.devopstecmi.com:/usuario/devopstecmilocal
 * branch            staging    -> FETCH_HEAD
 * [new branch]      staging    -> origin/staging
Already up-to-date.
```

```
>cd repo_sitio

>dir
 Volume in drive C has no label.
 Volume Serial Number is 8ED7-26EB

 Directory of C:\repo_sitio

23/03/2022  09:23 a. m.    <DIR>          .
23/03/2022  09:23 a. m.    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  751,103,623,168 bytes free

>git init
Initialized empty Git repository in C:/repo_sitio/.git/
```

Explicación

Flujos CI/CD

Este método es perfecto cuando dos o más desarrolladores trabajan juntos en una aplicación y que la fusión del código se realice correctamente sin generar conflictos.

Integración continua	Permite que diferentes desarrolladores carguen y combinen cambios de código en la misma rama del repositorio de manera frecuente.
Entrega continua	El nuevo código introducido y que ha pasado el proceso de CI se publica automáticamente en un ambiente de producción.
Despliegue continuo	Todos los cambios en el código que han pasado por las dos etapas anteriores se implementan automáticamente en producción.

Explicación

Archivos .md

Los archivos markdown se generan con un lenguaje de secuencias de comandos para dar formatos a textos sin formato en un diseño en específico.

Estos archivos se utilizan en Git para escribir el Readme.md (IONOS, 2020).

Información básica del programa
Guías
Referencias
Instrucciones
Código

Universidad Tecmilenio

Ejercicio

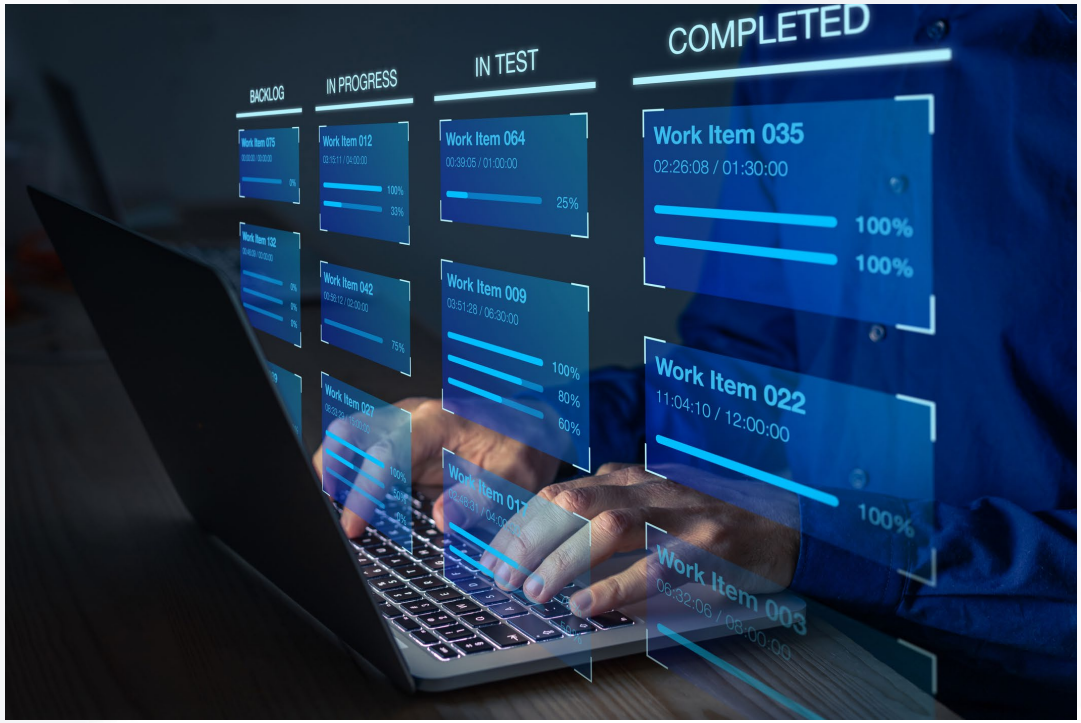
Utilizando Git y GitHub, realiza lo que se pide:

1. Identifica un repositorio público en GitHub, escribe la URL y el tipo de información compartida.
2. Inicializa un nuevo repositorio localmente que recibirá la información del repositorio del paso anterior.
3. Clona el repositorio del paso 1 en el repositorio del paso 2.
4. Configura Git con tus datos de usuario y correo electrónico, y muestra la configuración del entorno antes y después de agregar los datos solicitados.
5. En tu repositorio local, agrega un archivo readme.md, un archivo de nombre ToDo y otro de nombre Lista_Desarrollo.
6. Agrega el archivo ToDo al área de preparación (staging).
7. Verifica el estado del repositorio.
8. Agrega los archivos readme.md y Lista_Desarrollo al área de preparación (staging).
9. Confirma los cambios.
10. Renombra el archivo Lista_Desarrollo a Autores y confirma los cambios.
11. Dentro del archivo Autores, agrega tu nombre.
12. Cancela los cambios hechos al archivo Autores (git status te dirá cómo hacerlo).

Universidad Tecmilenio

Cierre

Utilizar git en los proyectos de desarrollo ahorra mucho tiempo, da seguimiento a los cambios en el código fuente y permite que varios desarrolladores trabajen a la vez. Junto con la herramienta CI/CD, permite que los enlaces al código fuente sean más rápidos y sin errores, generando proyectos más estables y automáticos.



Bibliografía

• GitHub. (2020). *GitHub.* Recuperado de https://github.com/

• IONOS. (2020). *Archivo readme: resumen con plantilla*. Recuperado de https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/archivo-readme/

Universidad Tecmilenio

Especialización DevOps

Docker Deep Dive

Semana 3

En el pasado era difícil subir una aplicación final al entorno en donde se iba a desplegar, pues al mudar las configuraciones y las codificaciones siempre pasaban errores.

Para solucionar este problema, se creó Docker, una herramienta que te asegura minimizar los errores al migrar e implementar en diferentes plataformas.

Explicación

Docker es una plataforma abierta para desarrollar, trasladar y ejecutar aplicaciones en un entorno seguro (Diez, s.f.).

Se enfoca en administrar y ejecutar contenedores de aplicaciones. Puede instalarse en cualquier plataforma.

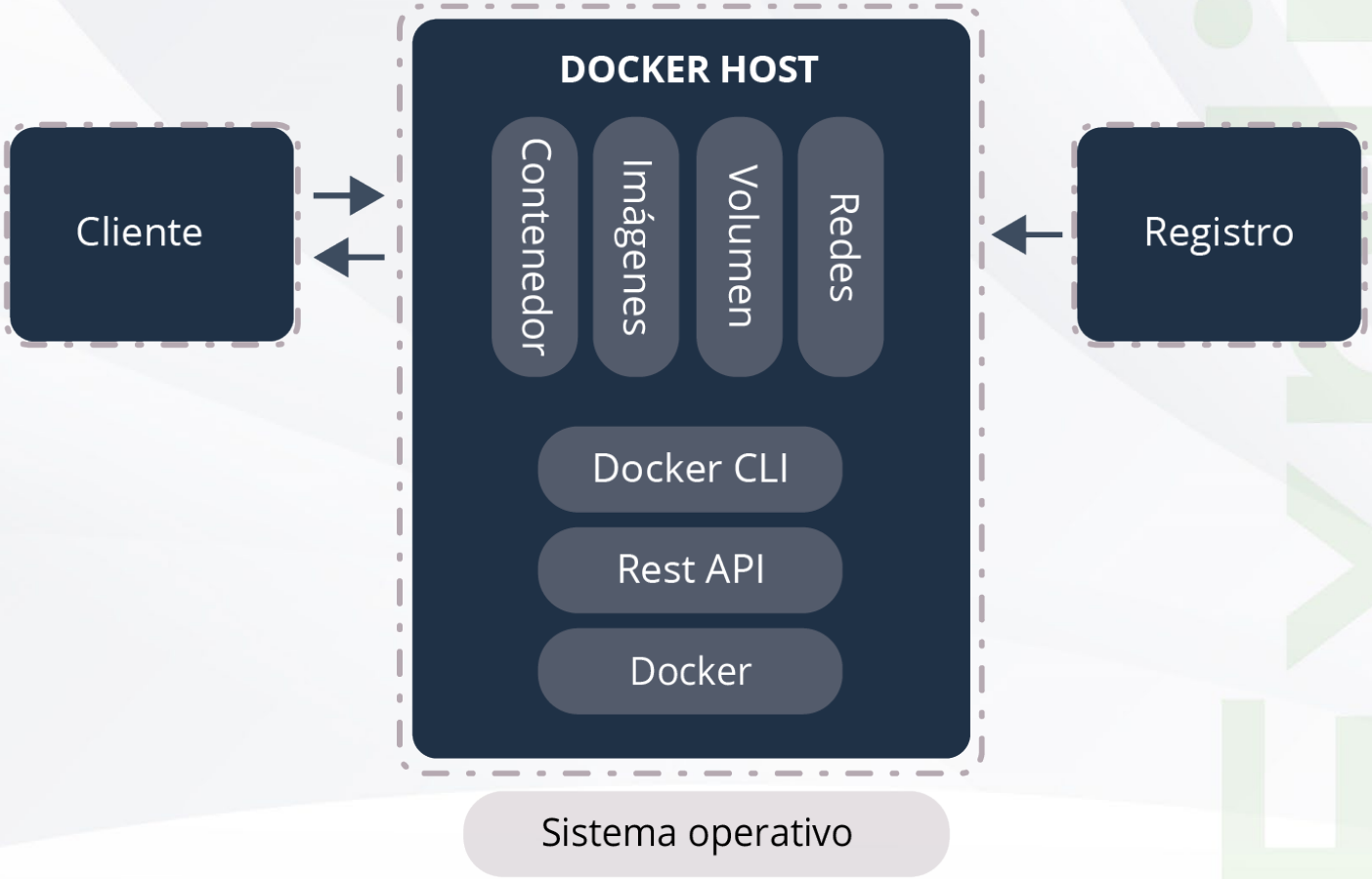
Contenedores en la nube.
Ligereza.
Ejecución sencilla.
Compartir.

Universidad Tecmilenio

Explicación

Arquitectura

Docker usa una arquitectura cliente-servidor.



Explicación

Contenedor

- Es una instancia ejecutable de una imagen. Se puede crear, iniciar, mover o destruir mediante el cliente o una API de Docker.
- Se ejecuta en máquinas locales, virtuales o en la nube
- Es portable (se puede ejecutar en cualquier plataforma).
- Está aislado de otros contenedores funcionando con sus software, configuraciones y ejecutables propios.

El propósito de los contenedores es ejecutar una sola aplicación o servicio, por ende, las imágenes deben ser ligeras de manera intencional, quitando todas las partes no esenciales (Rickerd, 2019).

Imágenes

- Es una plantilla de solo lectura, con instrucciones para crear un contenedor de Docker.
- Imagen empacada para que una aplicación funcione.
- Se pueden crear de forma personalizada y con características propias.

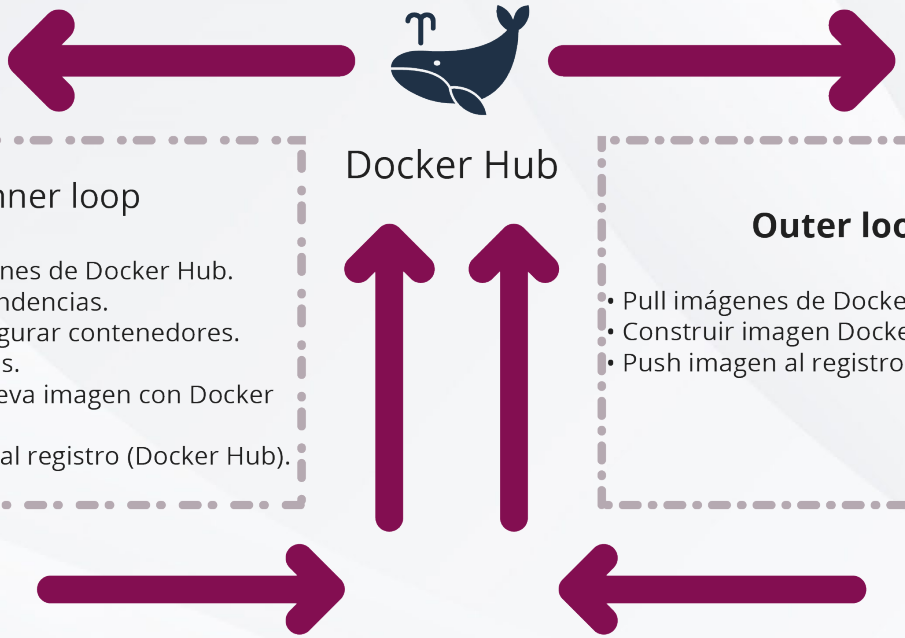
Explicación

Conceptos de Docker

Volúmenes de datos Host Anónimos Nombrados	Objeto separado del ciclo de vida del contenedor.
Redes en Docker Bridge Host None	Abstracción creada para facilitar la administración de la comunicación entre contenedores.
Registro Docker Administra Controla Integra	Lugar donde se almacenan las imágenes de Docker.

Explicación

CI/CD Docker



Ejercicio

Instala Docker en el sistema operativo de tu preferencia. Para conocer el proceso, los archivos binarios y/o fuente de Docker, visita las páginas:

Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones.
https://docs.docker.com/get-docker/
https://hub.docker.com

1. Visita el registro oficial de Docker. Para proceder, deberás crear una cuenta de Docker (registrarte en el sitio), o bien, si tienes ya una cuenta existente, iniciar sesión con ella.
Busca las imágenes de Apache y PHP para conocer la manera de descargar las variables de entorno.

2. Crea un contenedor con la imagen de Apache con PHP.

3. Coloca restricciones de consumo de memoria. Puedes conocer más acerca de las limitaciones de memoria y de procesador en la siguiente liga:

Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones.
https://docs.docker.com/config/containers/resource_constraints/

Las limitaciones de memoria son las siguientes:
A. 100 MB límites de RAM.
B. Solo podrá acceder al CPU 0.

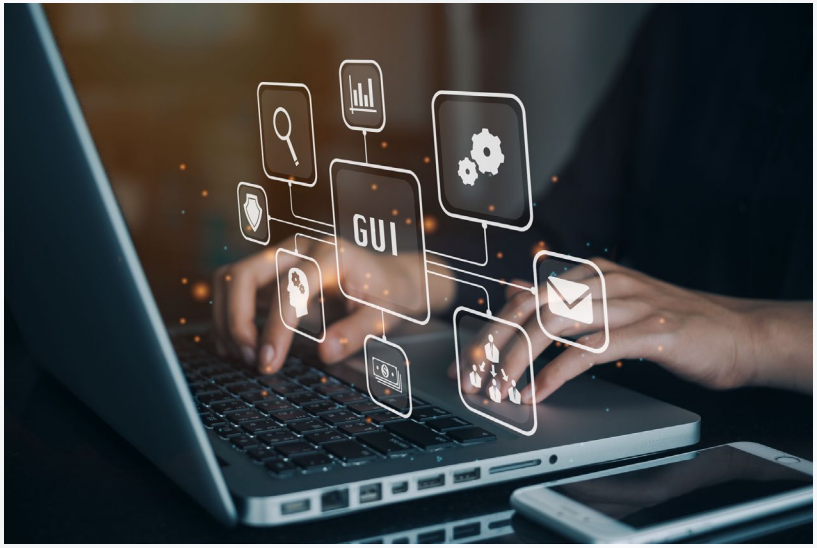
4. Trabaja con dos variables de entorno (verifica las características del entorno en el Docker Hub en las imágenes antes mencionadas).
A. ENV = dev.
B. VIRTUALIZATION = docker.

5. Haz accesible al contenedor vía puerto 5555 en el navegador.

6. Crea un directorio en el Docker Host (máquina local) en la ruta: /opt/source1 (crea el directorio en tu máquina local). Debe persistir el código que se incluya en el webserver. En este caso, para pruebas, utilizarás un phpinfo que debe sobrevivir a la eliminación del contenedor.

7. Haz las capturas del funcionamiento del contenedor con el webserver en el puerto antes mencionado.

Cierre



Docker es una poderosa herramienta que permite agilizar el diseño, codificación, pruebas y despliegues de desarrollos e implementaciones específicas. Es importante practicar con los comandos para realizar despliegues más rápidos y automáticos de manera limpia.

Bibliografía

• Diez, B. (s.f.). *Hola Docker CI / CD - GitHub Actions*. Recuperado de https://lemoncode.net/lemoncode-blog/2020/2/12/hola-docker-ci-cd-github-actions

• Rickerd, G. (2019). *Docker Volumes: Why, When, and Which Ones?* Recuperado de https://spin.atomicobject.com/2019/07/11/docker-volumes-explained/

Universidad Tecmilenio

Especialización DevOps

Docker Deep Dive. Parte 2

Semana 3

Docker es una herramienta de bajo consumo de recursos y actualmente cuenta con más de ocho millones de imágenes de sistemas operativos, entornos de desarrollo de base de datos, herramientas de monitoreo de DevOps, etc.

Sin embargo, para sacar el mejor provecho posible es importante aprender a crear imágenes personalizadas según los requerimientos de los proyectos de desarrollo que se realicen y crear configuraciones propias para los contenedores según el requerimiento.

Explicación

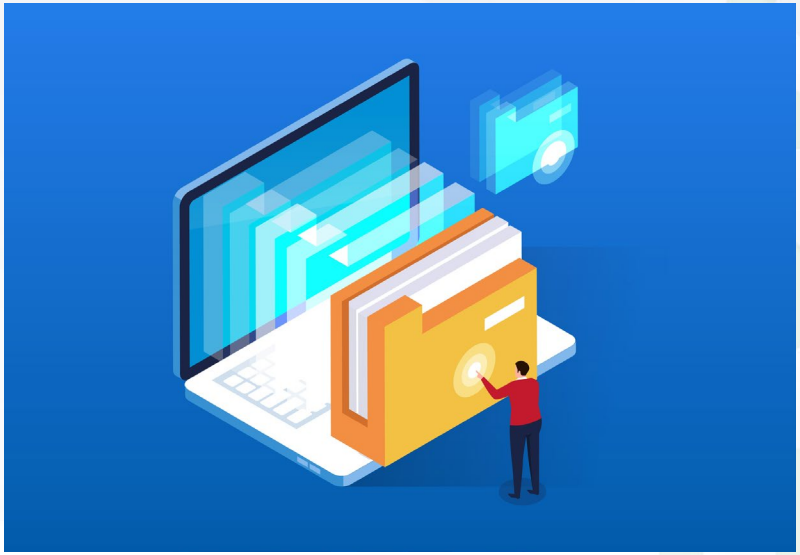
Dockerfile

Docker crea imágenes de manera autónoma en un archivo de texto que se denomina Dockerfile (Docker, s.f.).

- Detener
- Destruir
- Reconstruir

Todos los contenidos recursivos, archivos y subdirectorios que se encuentran en el directorio donde está el Dockerfile se considerarán al momento de construir la imagen.

Por ello, es importante colocar el Dockerfile en un directorio vacío y agregar archivos, componentes o dependencias necesarias, en caso de ser preciso.



Explicación

Sintaxis Dockerfile

Basado en líneas de comandos, se considera una línea por instrucción o para comentarios (#).

Editores de texto como:

Gedit

Nano

Vim

FROM

ARG

RUN

CMD

ENTRYPOINT

COPY/ADD

COPY

ENV

WORKDIR

EXPOSE

LABEL

USER

VOLUME

Explicación

Docker ignore

El principal objetivo es no enviar una gran cantidad de archivos o directorios y evitar redundancias o excesos de tamaño. El cliente interpreta el archivo .dockeringnore como un script con una lista de instrucciones.

La instrucción empieza con el operador numeral “#”, el cual incluye lo siguiente:

'*': empata cualquier secuencia de caracteres no separadores.

'?': empata con cualquier carácter separador.

'[' ['^'] { rango de caracteres } ']’.

'**’: considera todos los subdirectorios y archivos, incluyendo aquellos de tamaño cero.

'!': las líneas con signos de admiración sirven para establecer exclusiones.

'#': las líneas que inician con el numeral que son ignoradas se consideran como comentarios.

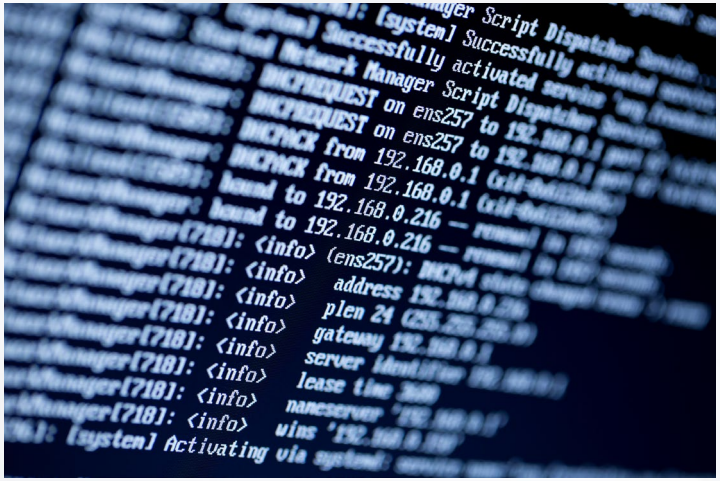
Ejercicio

1. Crea una carpeta en la raíz de Docker llamada imagenweb.
2. Crea un archivo de llamado Dockerfile. Puedes usar cualquier editor de consola como Vim, Nano, etc.
3. Agrega el siguiente contenido al Dockerfile:

```
FROM debian:latest
RUN apt-get update
RUN apt-get -y install apache2
EXPOSE 80
CMD /usr/sbin/apache2ctl -D FOREGROUND
```

4. Guarda el archivo Dockerfile.
5. Ejecuta, desde la línea de comandos, el comando “sudo docker build -t werbserver” para construir la imagen personalizada.
6. Verifica si se armó correctamente el contenedor mediante el comando: “docker images”.
7. Arranca un nuevo contenedor en el puerto 8085 leyendo el puerto 80.
8. Verifica su funcionamiento mediante un navegador abriendo el localhost:8085.
9. Crea una nueva imagen a tu gusto. Puedes buscar algunas ideas en los foros de la comunidad de Docker.
10. Repite los puntos del 1 al 6 con el contenido de tu imagen y configuraciones seleccionadas.

Cierre



El uso correcto de Dockerfile es necesario para sacar el mayor provecho posible a las imágenes personalizadas y con las configuraciones apropiadas para el desarrollo de proyectos, implementar funcionalidades y optimizar el manejo de los contadores con las características apropiadas y la sintaxis correcta para la definición de comandos en el archivo.

Docker es una herramienta poderosa, pero es importante practicar las fases para su máximo rendimiento.

Bibliografía

• Docker. (s.f.). *Dockerfile reference*. Recuperado de https://docs.docker.com/engine/reference/builder/ /

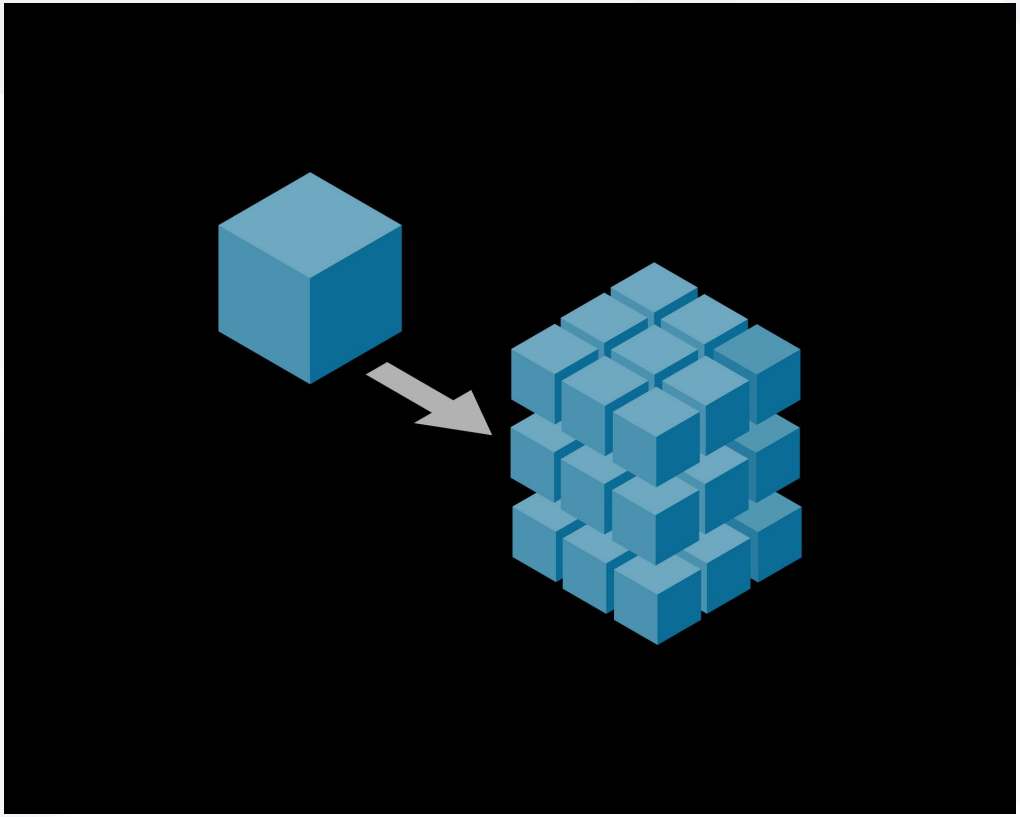
Universidad Tecmilenio

Especialización DevOps

Docker Deep Dive. Parte 3

Semana 3

Docker ofrece el manejo de varios contenedores en un proyecto, un conjunto orquestado para integrar todos los servicios de una aplicación completa en contenedores con una utilización mínima de recursos.



Universidad Tecmilenio

Explicación

Docker compose

Una aplicación se compone de varios microservicios que a la vez forman un proceso en conjunto (Barrios, 2020).

Esta unión de microservicios, Docker compose lo describe en un solo archivo de configuración YAML.

Instrucciones con líneas de comandos “docker-compose”. Usa subconjuntos de JSON.

```yaml
version: "3.9"  #cometario
services:
  web:
     container_name: ngninxejemplo
      ports:
        - "8080-80"
      image: nginx
      volumes:
        - .:/src
networks:
  counter-net:

volumes:
  conter-vol:
```

Universidad Tecmilenio

Explicación

Nodos de Docker Compose

- Version
- Services
- Volumes
- Networks

Docker trabaja a través de nodos que permiten establecer las configuraciones (Poulton, 2020).

Universidad Tecmilenio

Explicación

Variables y archivos de entorno

Las variables y archivos de entorno hacen que la configuración sea más dinámica y automatizable.

Cuando se utiliza una variable desde el Shell, a través de un contenedor de servicio, en el nodo “environment” no se asigna valor.

```
web:
  enviroment:
    - DEBUG=1
```

Cuando se utilizan múltiples variables se pueden declarar en un archivo auxiliar al Docker Compose.

```
web:
  environment:
    - DEBUG
```

```
web:
  env_file:
   - mi_archivo.env
```

Universidad Tecmilenio

Explicación

Perfiles de servicio

Ajustan el modelo de aplicación de un Docker Compose a varios usos y entornos habilitando selectivamente servicios.

Asignando de 0 a n perfiles.

Si un servicio es asignado a un perfil, este se ejecutará únicamente cuando el perfil sea activado.

```yaml
version: "3.9"
services:
  web:
    image: httpd
    profiles: ["web"]

  phpmyadmin:
    image: phpmyadmin
    dpends_on:
      - db
    profiles:
      - debug

  support:
    image: guacamole

  db:
    image: mariadb
```

Universidad Tecmilenio

Explicación

Trabajo en red en compose

Docker crea una red simple para cada aplicación y todos los servicios se comunican por medio de esta red.

Los contenedores se pueden localizar por su nombre, pues Docker genera un host idéntico al nombre del contenedor.

```
version: "3.9"
services:
  web:
    image: httpd
    ports:
      - "8080:8080"
  db:
    image: mariadb
    ports: "3306:3306"
```

Cuando se corre el comando "docker-compose up":
Se crea una red llamada miapp_default.

Un contenedor se crea usando la configuración en el nodo del servicio "web".

Un contenedor se crea usando la configuración en el nodo de servicio "db".

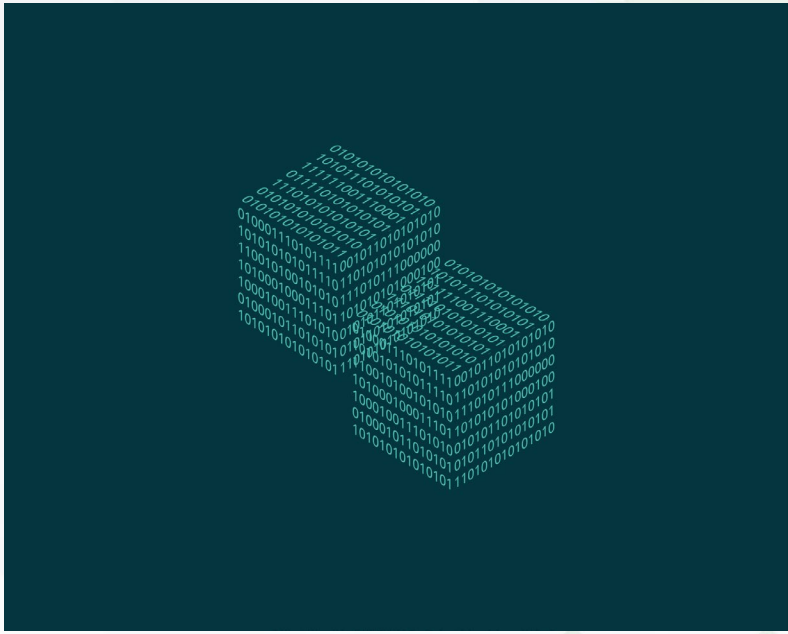
```
version: "3.9"
services:
  web:
    build:
    links:
      - "db:database"
  db:
    image: mariadb
    ports: "3306:3306"
```

Ejercicio

1. Instala Docker Compose. Puedes ir a la documentación oficial de Docker para obtener detalles de cómo realizar la instalación de Docker Compose. Si estás trabajando con Docker Desktop, puedes omitir este paso, ya que, por defecto, en Docker Desktop se encontrará ya instalado.
2. Busca la imagen más reciente de WordPress que incluya el servidor http (Apache) y que tenga ya integrado PHP. Puedes navegar en el Docker Hub para encontrarla y para hacer uso de ella en el Docker Compose.
3. Busca la imagen más reciente de MySQL para hacer uso de ella en el Docker Compose.
4. Usa un editor de texto para crear tu Docker Compose. Recuerda nombrar tu archivo como "docker-compose.yml". En el archivo, asegúrate de:
a. Crea un servicio llamado wordpress, el cual activa la imagen de WordPress con Apache y PHP. Este servicio activa el acceso por el puerto 8080 al 80 (8080:80). Coloca las siguientes variables de entorno para inicializar la configuración de WordPress. Las variables serán las siguientes:
   WORDPRESS_DB_HOST: mysql
        WORDPRESS_DB_USER: root
        WORDPRESS_DB_PASSWORD: root
   WORDPRESS_DB_NAME: wordpress
b. Crea un servicio llamado mysql, en donde actives la imagen de mysql. Crea un volumen para almacenar de manera persistente el contenido de la ruta: /var/lib/mysql. Utiliza las siguientes variables de entorno para la configuración de la base de datos y la contraseña del usuario root:
   MYSQL_DATABASE: wordpress
   MYSQL_ROOT_PASSWORD: root
5. Guarda tu Docker Compose.
6. Ejecuta el comando "docker-compose up" para lanzar tus contenedores según la configuración solicitada. En caso de error, corrige de acuerdo con los mensajes de advertencia.
7. Realiza pruebas para asegurar que los contenedores estén funcionando correctamente.
8. Documenta los pasos seguidos en caso de haber encontrado errores en la implementación. Agrega los pasos para solucionar las problemáticas y junta todo en un documento. Puedes colocar capturas de pantalla de la consola conforme se realiza el despliegue como evidencias de tu proceso.

Cierre

Docker no solo es una herramienta para la conjunción de código y despliegue automatizado, sino que también ayuda a desplegar y administrar una aplicación multicontenedor que necesite de varios microservicios para ser más rápida en la administración, al otorgar un archivo YAML para la configuración de los contenedores, imágenes, redes, variables de entorno, etc.



Bibliografía

• Docker. (s.f.). *Using profiles with Compose*. Recuperado de https://docs.docker.com/compose/profiles/

• Barrios, E. (2020). *¿Qué demonios es Docker y Docker-Compose? y cómo Dockerizar Dotnet Core WebApi y SQL Server en un ambiente de desarrollo ideal.* Recuperado de https://dev.to/ebarrioscode/que-demonios-es-docker-docker-compose-y-como-dockerizar-dotnet-core-webapi-y-sql-server-en-un-ambiente-de-desarrollo-ideal-95a

• Poulton, N. (2020). *Docker Deep Dive*. Estados Unidos: Amazon.