Instalación y Práctica con herramientas

En esta sección realizaremos la instalación de Docker mediante el aprovisionamiento que nos brinda Vagrant, software el cual hemos instalado y configurado en la unidad n°1. Para esto reutilizaremos la imagen de Ubuntu que hemos creado en VirtualBox y también parte de su configuración para lograr mediante Docker una arquitectura basada en contenedores, en la que tendremos dos de ellos. Uno corresponde a una imagen de php con el servidor web apache, la cual en el archivo Dockerfile la podremos identificar con el nombre de “php:7.0-apache”, y la otra corresponde a una imagen del servidor de base de datos MySQL.

Tal como hemos mencionado anteriormente, los contenedores Docker no persisten datos nuevos, cualquier cambio generado en el contenedor ya sea modificar algún archivo o generan nuevos, los cambios se perderán si se reinicia el servicio o se reconstruye la imagen, ya veremos de qué se trata esto. Para dar una solución a esto Docker nos permite persistir los datos generados en el contenedor por fuera del mismo mediante el montado de un volumen o un directorio de la máquina host (en nuestro caso es la imagen Ubuntu en Virtualbox) con un directorio del contenedor. Podemos observar en el archivo de configuración docker-compose.yml que ese directorio, para nuestro ejemplo, corresponde al directorio "/var/www/utn-devops-app” para la imagen de Ubuntu y el directorio “/var/www/html" para el contenedor Docker. Este tipo de configuración corresponde a la sección “volumes:” del archivo anteriormente mencionado.

Los contenedores de Docker nos permiten tener una infraestructura inmutable dado que no hay persistencia de datos, sin embargo si deseamos modificar alguna configuración específica, nos permite realizarlo de distintos modos. El alcance de este curso no cubre estos aspectos pero para que se tenga conocimiento de estos métodos los mencionaremos, los comandos que permiten realizar estas variantes son: “docker export” para exportar el contenido de un contenedor (sin incluir los volúmenes), “docker save” para guardar los datos de una nueva imagen y “docker commit” para crear una versión nueva de la imagen a partir del estado actual de un contenedor (esto permite observar las diferencias entre imágenes tal como si fuera el código versionado de una aplicación). A su vez, la versatilidad que nos brinda el uso de las distintas imágenes se puede amplificar mediante el uso de Docker Hub. Lo que nos permite esta herramienta es subir nuestras propias imágenes a un repositorio en la nube, tal como si fuera un repositorio de una aplicación y permitir descargarlas y mantener el versionado desde un lugar centralizado

1. Abrir una terminal de comandos (presionar las tecla “Windows” + r, escribir “cmd” y luego hacer clic en “Ok”). Luego de cada comando debe presionar la tecla “Enter” para que se ejecute”

2. cd UTN-DevOps/utn-devops
3. git pull

4. git checkout unidad-2-docker
5. vagrant up --provision
6. set PATH=%PATH%;C:\Program Files\Git\usr\bin
7. vagrant ssh
8. cd /vagrant/docker/
9. ls -l
    a. En este directorio posee el siguiente contenido:
        i. Directorio “configs”: aquí se alojaremos todas las configuraciones necesarias para nuestros contenedores
        ii. Archivos “Dockerfile” y “docker-compose.yml”: cada uno contiene las instrucciones necesarias para configurar los contenedores.
10. sudo docker-compose up -d
    a. Este comando lo que hace es revisar primeramente el archivo Dockerfile que se encuentra en el directorio, aplicar las directivas que se encuentran en él y luego pasa a ejecutar las directivas que se encuentran en el archivo docker-compose.yml
    b. El archivo Dockerfile contiene instrucciones para configurar un sólo contenedor. En nuestro caso será el contenedor que contiene la imagen de php y apache (servidor web)
    c. El archivo docker-compose.yml se utiliza para configurar varios contenedores en una sóla máquina host (nuestra imagen Ubuntu). Este simple archivo de configuración nos va a proporcionar las directivas necesarias para que podamos definir la configuración de múltiples contenedores, por ejemplo establecer dependencias entre ellos, la imagen requerida para cada contenedor., los directorios a los cuales se podrán acceder entre contenedores (recordemos que una de las ventajas de los contenedores y los microservicios es la posibilidad de aislamiento), el mapeo entre puertos que se ejecutan dentro del contenedor y el puerto que se dará accesibilidad por fuera, y variables que se utilizarán para la configuración interna de cada contenedor (en el ejemplo definiremos el nombre de la base de datos y la clave de administrador)
    d. La ejecución del comando revisará si las imágenes especificadas en los archivos antes mencionados están descargadas, en caso de no estarlo las descargara mediante la ejecución interna del comando “docker pull [nombre imagen]”. Hará una compilación de la imagen, esto quiere decir que se creará un caché con todas las especificaciones para que cuando se requiera volver a iniciar los contenedores sea de lo más rápido posible e iniciará todos los contenedor definidor en docker-compose.yml. La primera vez que se ejecuta este comando demora un tiempo hasta que finalice.
11. sudo docker-compose down

    a. Este comando detiene todos los contenedores. En el caso de querer iniciarlos nuevamente ejecute el comando anterior “sudo docker-compose up -d”
12. sudo docker-compose up -d
    a. Levantamos nuevamente todos los contenedores
13. sudo docker ps
    a. Con este comando veremos todos los contenedores creados
14. sudo docker exec -it apache2_php /bin/bash
15. cd /var/www/html/myapp
    a. En este directorio se aloja una aplicación de ejemplo para la práctica de este curso. La descarga de la aplicación y la instalación del software necesario se especificó en el archivo Vagrant.bootstrap.sh
16. composer update --no-scripts --prefer-dist
    a. Configuramos la aplicación
    b. Este comando suele demorar bastante tiempo, esperar hasta que finalice
17. chmod -R 777 storage bootstrap/cache
    a. Configuramos la aplicación
18. exit
19. sudo docker exec -i -t mysql /bin/bash
    a. Con este comando ingresaremos en el contenedor llamado mysql. Si se quisiera, también podríamos ingresar mediante la identificación “CONTAINER ID” que nos brinda el comando “sudo docker ps”, en lugar de la palabra “mysql”
20. mysql -uroot -proot devops_app < /tmp/script.sql
    a. Este comando creará una tabla con algunos registros necesarios para el ejemplo de esta unidad.
21. exit
    a. Con esto salimos del contenedor y volveremos a estar dentro de la máquina Ubuntu
22. Ingresar la siguiente url en la barra de su navegador: http://127.0.0.1:8081/ o http://utn-devops.localhost:8081/
    a. Con esto se verá la aplicación montada sobre el contenedor Docker que tiene un servidor web escuchando en el puerto 8081 la cual se conectará al servidor de base de datos MySQL que se encuentra en un contenedor distinto
23. exit
24. vagrant halt
    a. Ejecute este comando cuando ya no desee continuar trabajando con la máquina virtual y/o desee apagar su equipo.