Facilitador(a): Aris Castillo Asignatura: Sistemas Operativos

Estudiante: Gabriel Díaz Fecha: 16-11-2020 Grupo: 1IF131

**Descripción general de Docker**

Docker es una plataforma abierta (licencia de código abierto Apache 2.0) para desarrollar, enviar y ejecutar aplicaciones. Docker le permite separar sus aplicaciones de su infraestructura para que pueda entregar software rápidamente.

La plataforma de Docker usa el kernel de Linux permite crear contenedores ligeros y portables para las aplicaciones software que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues.

Docker utiliza el concepto de contenedores los cuales permiten empaquetar y ejecutar aplicaciones en un entorno aislado y seguro. Docker no utiliza la plataforma de hypervisor para ejecutarse en el host instalado. Docker se ejecutan directamente dentro del kernel de la máquina host. Esto significa que puede ejecutar más contenedores en una combinación de hardware determinada que si estuviera utilizando máquinas virtuales. ¡Incluso puede ejecutar contenedores Docker dentro de máquinas host que en realidad son máquinas virtuales!

Las herramientas del contenedor, como Docker, ofrecen un modelo de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos. Docker también automatiza la implementación de la aplicación (o conjuntos combinados de procesos que constituyen una aplicación) en este entorno de contenedores.

Estas herramientas desarrolladas a partir de los contenedores de Linux, lo que hace a Docker fácil de usar y único, otorgan a los usuarios un acceso sin precedentes a las aplicaciones, la capacidad de implementar rápidamente y control sobre las versiones y su distribución.

**Ventajas de los contenedores Docker**

Modularidad: El enfoque Docker para la creación de contenedores se centra en la capacidad de tomar una parte de una aplicación, para actualizarla o repararla, sin necesidad de tomar la aplicación completa.

Control de versiones de imágenes y capas: Cada archivo de imagen de Docker se compone de una serie de capas. Estas capas se combinan en una sola imagen. Una capa se crea cuando la imagen cambia. Cada vez que un usuario especifica un comando, como ejecutar o copiar, se crea una nueva capa.

Docker reutiliza estas capas para construir nuevos contenedores, lo cual hace mucho más rápido el proceso de construcción. Los cambios intermedios se comparten entre imágenes, mejorando aún más la velocidad, el tamaño y la eficiencia -Keep it small: a closer look at Docker image sizing. El control de versiones es inherente a la creación de capas. Cada vez que se produce un cambio nuevo, básicamente, usted tiene un registro de cambios incorporado: control completo de sus imágenes de contenedor.

Restauración: Probablemente la mejor parte de la creación de capas es la capacidad de restaurar. Toda imagen tiene capas. ¿No le gusta la iteración actual de una imagen? Restáurela a la versión anterior. Esto es compatible con un enfoque de desarrollo ágil y permite hacer realidad la integración e implementación continuas (CI/CD) desde una perspectiva de las herramientas.

Implementación rápida: Solía demorar días desarrollar un nuevo hardware, ejecutarlo, proveerlo y facilitarlo. Y el nivel de esfuerzo y sobrecarga era extenuante. Los contenedores basados en Docker pueden reducir el tiempo de implementación a segundos. Al crear un contenedor para cada proceso, puede compartir rápidamente los procesos similares con nuevas aplicaciones. Y, debido a que un SO no necesita iniciarse para agregar o mover un contenedor, los tiempos de implementación son sustancialmente inferiores. Además, con la velocidad de implementación, puede crear y destruir la información creada por sus contenedores sin preocupación, de forma fácil y rentable.

**Usos de Docker**

El uso de docker para las empresas se ha popularizado por su agilidad para crear proyectos, desde el diseño hasta la implantación, pasando por un testeo del producto y el mantenimiento del mismo. Docker viene a solucionar cada uno de esos puntos utilizando una metodología de desarrollo ágil que le permite a los involucrados en proyecto tener una mejor administración de su producto, algunas características que se pueden mencionar:

**Aislamiento de aplicaciones:** Docker ofrecerá el mismo sistema base para desarrollar o testear aplicaciones o servicios. De igual forma es un sistema aislado del sistema anfitrión, por lo que la ejecución de una aplicación en esa máquina no afectará al puesto en el que estamos trabajando.

Un claro ejemplo de aislamiento de aplicaciones es cuando queremos desplegar dos servidores, pero cada uno de ellos tiene diferentes dependencias que causan conflictos con las que necesita el otro. Desplegando cada servidor en contenedores distintos, solventamos este problema de forma fácil, pudiendo aislar los servidores y sus dependencias de posibles errores por dichos conflictos.

**Ahorro de costes en servidores:** De igual forma en la que creamos imágenes de sistemas para virtualizar equipos que desplegar para trabajar, podemos generar contenedores que desplieguen un servidor sobre el que ejecutar servicios.

Virtualizando estos servidores con Docker, el ahorro en hardware y el aprovechamiento o rendimiento del existente será considerable, sin un consumo de memoria tan alto y con la capacidad de gestionar más eficientemente la memoria disponible entre los servicios que lo requieran; a diferencia de las máquinas virtuales tradicionales que acapararán parte de esa memoria para desplegar el sistema de virtualización sobre el que irá el servidor que queramos virtualizar.

**Control de versiones y Depuración:** Una de las características de Docker que normalmente no se engloba en virtualización, es el sistema de control de versiones que nos ofrece, pudiendo regresar a la versión que queramos en caso de necesitarlo.

Muy útil por ejemplo en el caso de un servidor web, donde tras una actualización de seguridad nos hemos dejado una abertura por la que nos han tirado el servicio. Cerramos el contenedor, regresamos a una versión anterior y lo ponemos de nuevo en producción. Desplegamos otro contenedor aislado y trabajamos en la actualización de seguridad corrigiendo el error explotado para tirar el servicio. Una vez lo hemos resuelto, no tenemos más que volver a desplegar el contenedor corregido en el entorno de producción. Esto último es cuestión de minutos, contando así con cortes de servicio mínimos para la corrección de errores o implementación de novedades.

**Gestión de proyectos:** Uno de los mayores problemas a los que se enfrentan los equipos de desarrollo, es el tener que trabajar todos bajo el mismo entorno. Cada equipo sobre el que se va a poner a prueba la aplicación o servicio siempre tendrá algo diferente al resto, una actualización de menos (o de más), una librería de la que otros no dispongan, o directamente un sistema operativo u otro.