Facultad de Ciencias Exactas y Naturales

Escuela de Informática

Cátedra Ingeniería de Sistemas

Ingeniería Sistemas III

Intercambio Tecnológico

“Docker”

Grupo: 11

Ariel Granda Solano

Brandon Castillo Badilla

Gorki Romero Valerio

Priscila Murillo Quintana

Wendy Carballo Chavarría

Profesora titular: Giselle Víquez Jiménez

Abril 2024

Contenido

1. Introducción. 3

2. Breve descripción de los contenedores. 3

3. ¿Qué es Docker? 3

4. Usos de Docker. 4

Creación y ejecución de arquitecturas de microservicios. 4

Implementación de código con canalizaciones de integración y entrega continua. 4

Creación de plataformas completamente administradas. 4

Creación de sistemas de procesamiento de datos altamente escalables. 4

5. Ventajas de utilizar Docker. 4

6. Desventajas de utilizar Docker. 5

7. Información adicional. 5

8. Conclusiones. 5

9. Recomendaciones. 6

10. Bibliografía 6

1. Introducción.

En el campo del desarrollo de software mantenerse actualizado es una necesidad imperativa. Se requiere una actitud proactiva hacia el aprendizaje y adaptarse al uso de nuevas herramientas diseñadas para mejorar la productividad, optimización de tiempos y mejorar la eficacia en la resolución de problemas complejos. Aprovechar las herramientas y recursos disponibles fomenta la innovación y la calidad de los productos entregados. Una de las herramientas que puede ser de gran utilidad son los contenedores, que permiten empaquetar y distribuir aplicaciones de manera eficiente garantizando que se ejecuten de manera idéntica en cualquier entorno. Esta tecnología facilita la gestión de dependencias, soporta una arquitectura de microservicios para aplicaciones complejas, simplifica el despliegue de aplicaciones y contribuye a una mejor gestión de recursos. Esta tecnología es ejemplificada por Docker, del cual hablaremos a continuación.

2. Breve descripción de los contenedores.

Los contenedores son unidades ejecutables de software en los que se empaqueta el código de la aplicación con sus bibliotecas y dependencias para ejecutarse en cualquier lugar. Para esto utilizan una forma de virtualización del SO en la que las características del Kernel se aprovechan para aislar procesos y controlar la memoria, CPU y disco a los que pueden acceder. Son pequeños y ligeros por lo que los convierte en una buena solución para las arquitecturas de microservicios, permite trabajar en escenarios en la nube y además son una base para equipos que adopten modelos DevOps. (IBM, s.f.) Un concepto importante relacionado con el tema es:

Contenerización.

El software debe diseñarse y empaquetarse incluyendo las variables de entorno relevante, archivos de configuración, bibliotecas y dependencias. (IBM, s.f.)

3. ¿Qué es Docker?

En el 2013 se lanzó la tecnología de contenedores de Docker, Docker Engine, de código abierto desarrollado en Go. Docker aprovechó conceptos existentes de contenedores y primitivas conocidas como cgroups y namespaces de Linux para desarrollar una tecnología que se centra en los requisitos de desarrolladores y operadores de sistemas para separar las dependencias de la aplicación de la infraestructura. La función principal del contenedor es aislar el software de su entorno para garantizar la funcionalidad uniforme aun en diferentes escenarios.

Esta tecnología crea una imagen de contenedor de Docker que es un paquete ligero, independiente y ejecutable que contiene el código, herramientas del sistema, tiempo de ejecución, bibliotecas y configuraciones para ejecutar una aplicación. Las imágenes de contenedor se convierten en contenedores en tiempo de ejecución, es decir, las imágenes de contenedor de Docker se ejecutan en Docker Engine para convertirse en contenedores. Docker está disponible para aplicaciones basadas en Linux y Windows, independientemente de la infraestructura el software contenerizado siempre se ejecutará igual. Los contenedores de Docker que se ejecutan en Docker Engine son estandarizados, para asegurar su portabilidad, y tienen capacidades de aislamiento

predeterminadas para brindar mayor seguridad. Además, al compartir el Kernel del SO del host no requieren otro SO, aumentando la eficiencia del servidor y los costos. (Docker Inc, 2024)

4. Usos de Docker.

Docker Engine resuelve los problemas de las dependencias para los desarrolladores y los equipos de operaciones. Los contenedores de Docker pueden utilizarse para crear aplicaciones y plataformas. (Amazon Web Services, s.f.) Además, puede facilitar tareas como:

Creación y ejecución de arquitecturas de microservicios.

Las arquitecturas de microservicios se refieren a un enfoque de diseño de software donde la aplicación se divide en pequeños servicios independientes que se ejecutan en sus propios procesos y se comunican entre sí (APIs). Este enfoque facilita la escalabilidad, pues cada microservicio puede escalar de forma independiente. Los contenedores pueden simplificar la creación, despliegue, y ejecución de estos microservicios al manejar muchos aspectos de la infraestructura subyacente y la comunicación entre servicios. (Intel Corporation, s.f.)

Implementación de código con canalizaciones de integración y entrega continua.

La entrega continua es una parte del desarrollo de software donde el código es construido, probado y desplegado de manera frecuente y automatizada. Esto facilita a los equipos de desarrollo lanzar nuevas características y correcciones de forma rápida y eficiente, reduciendo el tiempo entre el desarrollo de una función y su despliegue en producción. (Amazon Web Services, s.f.)

Creación de plataformas completamente administradas.

Se refieren a servicios en la nube que manejan la infraestructura subyacente, la escalabilidad, el mantenimiento, y la seguridad, permitiendo a los desarrolladores enfocarse exclusivamente en el desarrollo de aplicaciones. (Microsoft, s.f.)

Creación de sistemas de procesamiento de datos altamente escalables.

Estos sistemas se necesitan para manejar grandes volúmenes de datos rápidamente. Son clave para implementar soluciones de big data y pueden integrarse dentro de arquitecturas de microservicios para procesar datos en tiempo real o en lotes. (Amazon Web Services, s.f.)

5. Ventajas de utilizar Docker.

Portabilidad. Docker permite empaquetar una aplicación con todas sus dependencias en contenedores, lo que garantiza que se ejecute de la misma manera en cualquier entorno, desde el desarrollo hasta la producción.

Consistencia del entorno. Al definir las dependencias y configuraciones del entorno dentro de un contenedor, se asegura que todos los miembros del equipo trabajen con el mismo entorno, evitando que a algunos les funcione y a otros no.

Facilidad de implementación. Los contenedores Docker pueden implementarse fácilmente en diferentes entornos, ya sea en la nube, en servidores locales o en máquinas virtuales, simplificando el proceso de implementación y escalado.

Aislamiento. Docker utiliza el concepto de contenedores para aislar aplicaciones y sus dependencias del sistema operativo subyacente, lo que brinda seguridad y evita conflictos entre diferentes aplicaciones y sus dependencias.

Eficiencia de recursos. Los contenedores Docker comparten el mismo kernel del sistema operativo, lo que reduce la sobrecarga en comparación con las máquinas virtuales tradicionales, lo que permite una mejor utilización de los recursos del sistema.

6. Desventajas de utilizar Docker.

Complejidad inicial. Puede haber una curva de aprendizaje pronunciada para comprender los conceptos y la configuración inicial de Docker.

Gestión de imágenes. Mantener un repositorio de imágenes Docker puede requerir esfuerzos adicionales para garantizar la integridad y seguridad de las imágenes utilizadas en el desarrollo.

Rendimiento. Aunque ofrece eficiencia de recursos, en algunos casos puede haber una pequeña sobrecarga de rendimiento debido al uso de contenedores.

Volumen de almacenamiento. Los contenedores pueden aumentar el consumo de almacenamiento en comparación con las aplicaciones tradicionales debido a la duplicación de sistemas de archivos en cada contenedor.

Seguridad. Si no se configura correctamente, podría presentar riesgos de seguridad, especialmente si se ejecutan contenedores con privilegios elevados o si no se aplican adecuadamente las prácticas de seguridad recomendadas.

7. Información adicional.

Documentación oficial de Docker. El mejor lugar para comenzar es la documentación oficial de Docker en Docker Documentation. Aquí se encuentran guías detalladas, tutoriales y referencias para comenzar con Docker y profundizar en sus características y funcionalidades.

Tutoriales en línea. Hay muchos tutoriales disponibles en línea en plataformas como YouTube, Udemy, Coursera y Pluralsight. Estos tutoriales van desde introducciones básicas hasta temas más avanzados como la orquestación de contenedores con Docker Swarm y Kubernetes.

Libros. Hay varios libros disponibles que cubren Docker en profundidad. Algunos títulos populares incluyen "Docker Deep Dive" de Nigel Poulton y "Docker in Action" de Jeff Nickoloff.

Cursos en línea. Plataformas como Udemy, Coursera y edX ofrecen una variedad de cursos en línea sobre Docker, desde cursos introductorios hasta programas más avanzados de certificación.

8. Conclusiones.

Profesionalmente, hemos aprendido que mantenerse actualizado en el campo del desarrollo de software es crucial para mejorar la productividad y la eficacia en la resolución de problemas. Adoptar herramientas como Docker puede ser fundamental para simplificar el desarrollo, despliegue y gestión de aplicaciones, especialmente en entornos de microservicios y DevOps.

Personalmente, hemos ampliado nuestro conocimiento sobre la tecnología de contenedores y cómo Docker ha revolucionado la forma en que desarrollamos y desplegamos software. Esta experiencia nos ha permitido entender mejor los beneficios y desafíos asociados con el uso de Docker, así como explorar recursos adicionales para seguir aprendiendo y mejorando nuestras habilidades.

Técnicamente, hemos comprendido cómo Docker utiliza la virtualización a nivel de sistema operativo para crear entornos aislados y portables para las aplicaciones. También hemos identificado las ventajas de Docker, como la portabilidad, la consistencia del entorno y la eficiencia de recursos, así como las posibles desventajas, como la complejidad inicial y los riesgos de seguridad si no se configura correctamente.

En resumen, hemos reconocido la importancia de mantenernos actualizados en tecnologías emergentes como Docker para seguir siendo competitivos en el campo del desarrollo de software, mientras exploramos recursos adicionales como la documentación oficial, tutoriales en línea, libros y cursos para continuar mejorando nuestras habilidades técnicas.

9. Recomendaciones.

Como se mencionó en el documento, queda claro que Docker es fundamental para las empresas y programadores. Como resultado, al utilizar buenas prácticas se garantiza un entorno seguro y optimizado. Estas recomendaciones facilitarán un uso adecuado para evitarán errores y minimizar vulnerabilidades:

1. **Actualice Regularmente las Dependencias del Contenedor**: Mantenga las dependencias actualizadas dentro de los contenedores en todo momento para evitar fallas o ataques debido a las vulnerabilidades conocidas.
2. **Evitar Dependencias sin Uso**: Evite instalar y mantener dependencias que no son esenciales para la aplicación. Aumentará el tamaño de la imagen innecesariamente.
3. **Utilizar Imágenes Oficiales**: El uso de imágenes oficiales de Docker es vital para la seguridad y la optimización y luego se pueden realizar las modificaciones necesarias sobre una base confiable.
4. **Restringir Privilegios del Contenedor**: En la medida de lo posible, no ejecute contenedores que puedan ejecutarse con privilegios de superusuario. Reduce el nivel de permisos si es posible.
5. **Un Proceso por Contenedo**r: Limitar cada contenedor a una única tarea asignada, evitando ejecutar múltiples procesos en un mismo contenedor para mantener la claridad en su uso.
6. **Firmar y Verificar Imágenes**: Implementar la firma y verificación de imágenes para asegurar su autenticidad y protegerse contra la inyección de código malicioso.

Estas prácticas no solo buscan optimizar la seguridad y el rendimiento del entorno de Docker, sino también promover un desarrollo y despliegue de aplicaciones más eficiente y libre de complicaciones. (AssureSoft, 2018)

10. Bibliografía

Amazon Web Services. (s.f.). *AWS*. Obtenido de ¿Qué es la integración continua y la entrega/implementación continua?: https://docs.aws.amazon.com/es_es/whitepapers/latest/practicing-continuous-

integration-continuous-delivery/what-is-continuous-integration-and-continuous-deliverydeployment.html

Amazon Web Services. (s.f.). *AWS*. Obtenido de ¿Qué es Docker?: https://aws.amazon.com/es/docker/#:~:text=Docker%20le%20permite%20entregar%20c%C3%B3digo,mejorar%20el%20uso%20de%20recursos.

Amazon Web Services. (s.f.). *AWS*. Obtenido de ¿Qué es el procesamiento por lotes?: https://aws.amazon.com/es/what-is/batch-processing/

AssureSoft. (25 de septiembre de 2018). *AssureSoft*. Obtenido de https://www.assuresoft.com/es/blog/errores-que-debe-evitar-al-utilizar-docker

Docker Inc. (2024). *Docker*. Obtenido de https://www.docker.com/resources/what-container/

IBM. (s.f.). *IBM*. Obtenido de ¿Qué son los contenedores?: https://www.ibm.com/mx-es/topics/containers

Intel Corporation. (s.f.). *Intel*. Obtenido de https://www.intel.la/content/www/xl/es/cloud-computing/microservices.html#:~:text=Una%20arquitectura%20de%20microservicios%20es,aplicaciones%20basadas%20en%20la%20nube.

Microsoft. (s.f.). *Azure*. Obtenido de https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-paas