Universidad Autónoma De Chiapas

Licenciatura en ingeniería en desarrollo y tecnologías de software

Arquitectura Basada en Microservicios

Taller De Desarrollo 4

Dr. Luis Gutiérrez Alfaro

Alumno: Manuel Antonio Nuñez Sánchez

Matricula: A200884 "6°M"

INTRODUCCIÓN:

En esta investigación aprenderemos Los sistemas de información basados en arquitectura de microservicios ya que son una forma de diseñar y desarrollar aplicaciones de software en la que una aplicación se divide en múltiples servicios independientes y pequeños, cada uno de los cuales realiza una función específica.
Estos servicios se comunican entre sí a través de API y pueden ser desarrollados, implementados y escalados de manera independiente.

ApI rest:

Las API es un estilo de arquitectura de software que utiliza los métodos HTTP para acceder y manipular recursos en un sistema

Google Maps API:

Descripción: Proporciona acceso a los servicios de Google Maps, como la geolocalización, la búsqueda de lugares, las direcciones y las imágenes de mapas.
Uso común: Integración de mapas en sitios web y aplicaciones móviles, búsqueda de ubicaciones cercanas, rutas de navegación, seguimiento de ubicaciones en tiempo real, etc.

GitHub API:

Descripción: Permite a los desarrolladores interactuar con la plataforma de desarrollo de software de GitHub, lo que les permite acceder a repositorios, issues, pull requests, usuarios y más.
Uso común: Integración de GitHub en flujos de trabajo de desarrollo, automatización de tareas relacionadas con repositorios, seguimiento de problemas y colaboración en proyectos de código abierto.

Docker:

Docker es una plataforma de código abierto diseñada para desarrollar, enviar y ejecutar aplicaciones en contenedores. Los contenedores son entornos ligeros y portátiles que incluyen todo lo necesario para que una aplicación se ejecute, como el código, las bibliotecas y las dependencias, y se ejecutan de manera consistente en cualquier entorno compatible con Docker, ya sea una computadora local, un servidor en la nube o un centro de datos.
Docker se ha convertido en una tecnología fundamental en el desarrollo de aplicaciones modernas, especialmente en entornos de microservicios y en la creación de pipelines de CI/CD (Integración Continua/Entrega Continua). Permite a los desarrolladores empacar sus aplicaciones y todas sus dependencias en contenedores, lo que simplifica el despliegue y la administración de aplicaciones en una variedad de entornos.

Componentes Básicos.

Docker es una plataforma de contenedores que consta de varios componentes clave que trabajan juntos para permitir la creación, administración y ejecución de contenedores.
Algunos componentes básicos de Docker son:

Docker Daemon (dockerd): Es un servicio que se ejecuta en el host de la máquina. Es responsable de administrar los contenedores, imágenes, redes y volúmenes de Docker.
El Daemon escucha las solicitudes de la CLI de Docker y realiza acciones como la creación, ejecución o eliminación de contenedores.

Docker Client (docker): El cliente de Docker es la interfaz de línea de comandos que permite a los usuarios interactuar con el Docker Daemon.
Los usuarios envían comandos al cliente, que luego comunica estas solicitudes al Daemon.
Los comandos de Docker se utilizan para crear, configurar, administrar y supervisar contenedores y recursos relacionados.

Imágenes de Docker (Docker Images): Son plantillas de solo lectura que contienen el sistema de archivos y la configuración de una aplicación.
Estas imágenes se utilizan como base para crear contenedores.
Las imágenes se almacenan en el registro de imágenes de Docker y se pueden compartir y reutilizar.

Contenedores de Docker (Docker Containers): Los contenedores de Docker son instancias en tiempo de ejecución de imágenes de Docker.
Cada contenedor es una aplicación aislada que se ejecuta en un entorno independiente.
Los contenedores comparten el núcleo del sistema operativo del host, pero tienen su propio espacio de usuario y recursos aislados.

Componenentes Básicos:

Docker Registry: Un registro de Docker es un repositorio de imágenes de Docker donde se pueden almacenar y compartir imágenes públicas y privadas.
Docker Hub es un registro público popular.
Puedes configurar tu propio registro privado si necesitas mantener el control sobre tus imágenes.

Redes de Docker (Docker Networks): Docker permite crear redes virtuales que permiten la comunicación entre contenedores.
Los contenedores que se conectan a la misma red pueden comunicarse entre sí de manera eficiente.
Esto es útil para aplicaciones distribuidas y microservicios, donde diferentes contenedores deben interactuar.

Volúmenes de Docker (Docker Volumes): Los volúmenes de Docker son directorios o sistemas de archivos montados en un contenedor desde el host.
Se utilizan para persistir datos y compartir información entre contenedores.
Los volúmenes son útiles para garantizar que los datos no se pierdan cuando un contenedor se detiene o se elimina.

Arquitectura Docker:

La arquitectura de Docker se basa en una arquitectura cliente-servidor que permite la creación, administración y ejecución de contenedores.

Cliente de Docker (Docker Client): Es la interfaz de línea de comandos que utilizan los usuarios para interactuar con Docker.
Puede ser utilizado desde la línea de comandos o a través de herramientas gráficas que se conectan al Daemon de Docker.
Los comandos de Docker Client se utilizan para crear, configurar, administrar y supervisar contenedores, imágenes, redes y volúmenes.

Docker Daemon (dockerd): Es un servicio que se ejecuta en el host de la máquina.
Es responsable de administrar los recursos de Docker, incluyendo los contenedores, imágenes, redes y volúmenes.
El Daemon escucha las solicitudes del Docker Client y ejecuta acciones como la creación, ejecución o eliminación de contenedores.
También se encarga de gestionar la interacción con el núcleo del sistema operativo del host.

REST API de Docker: El Docker Daemon expone una API REST a través de la cual el Docker Client se comunica con él.
Esto permite que el Docker Client envíe solicitudes HTTP para realizar operaciones en contenedores y otros recursos de Docker.
La API de Docker también es utilizada por herramientas de gestión y orquestación, como Docker Compose, Docker Swarm y Kubernetes.
garantizar que los datos no se pierdan cuando un contenedor se detiene o se elimina.

Conclusión:

Podríamos decir que la arquitectura de Docker es fundamental para comprender cómo funcionan los contenedores y cómo se administran en un entorno Docker.
La arquitectura de Docker nos proporciona una forma eficiente y escalable de empaquetar, distribuir y ejecutar aplicaciones en contenedores, lo que ha revolucionado la forma en que se desarrollan y despliegan aplicaciones en entornos de desarrollo y producción.
Su enfoque en la portabilidad, el aislamiento y la eficiencia de recursos ha hecho que Docker sea una herramienta esencial en la construcción de aplicaciones modernas y en la adopción de prácticas de desarrollo ágiles.

Bibliografias

*https://www.oracle.com/mx/cloud/cloud-native/container-registry/what-isdocker/

*https://www.ibm.com/mx-es/topics/Docker

*https://learn.microsoft.com/es-es/dotnet/architecture/microservices/container-docker-introduction/dockerdefined