**UNIVERSIDAD TECNOLÓGICA DE SANTIAGO, UTESA.**



**Asignatura:**

INF-025-001 ALGORITMOS PARALELOS

**Tema:**

Actividad Semana 3

**Presentado por:**

Georges de Jesús Gil Pichardo

**Matricula:**

1-18-2363

**Presentado a:**

M.A. IVAN MENDOZA

**Docker y Docker Compose:**

**¿Qué es Docker?**

Docker es una plataforma de código abierto que permite automatizar el despliegue de aplicaciones dentro de contenedores de software. Un contenedor es una unidad estándar que puede contener cualquier aplicación junto con sus dependencias y ejecutarla en cualquier entorno. Docker proporciona una forma de empaquetar aplicaciones en contenedores, lo que facilita su distribución y ejecución en diferentes entornos.

**¿Para qué nos sirve en desarrollo?**

En desarrollo, Docker es invaluable porque:

Portabilidad: Las aplicaciones empacadas en contenedores Docker son portátiles y pueden ejecutarse en cualquier lugar, desde el entorno de desarrollo hasta la producción, sin preocuparse por las diferencias en los entornos.

Aislamiento: Los contenedores proporcionan aislamiento, lo que significa que una aplicación y sus dependencias pueden ejecutarse sin interferir con otras aplicaciones en el mismo sistema.

Rapidez: Docker permite la creación rápida y la eliminación de contenedores, lo que facilita las pruebas y el desarrollo iterativo.

Escalabilidad: Docker facilita la creación y gestión de aplicaciones escalables mediante el uso de contenedores y orquestación de contenedores en clústeres.

**¿Cómo se utiliza Docker?**

Imágenes de Docker:

Una imagen de Docker es como una plantilla que contiene todo lo necesario para ejecutar una aplicación. Puedes buscar imágenes en el Docker Hub, que es un repositorio público de imágenes de Docker, o puedes crear tus propias imágenes personalizadas utilizando un archivo de configuración llamado Dockerfile. Para descargar una imagen, utiliza el comando docker pull nombre\_de\_la\_imagen. Para crear tu propia imagen, crea un Dockerfile que especifique las instrucciones para construir la imagen y utiliza el comando docker build -t nombre\_de\_la\_imagen . en el mismo directorio que el Dockerfile.

Contenedores:

Un contenedor es una instancia en ejecución de una imagen de Docker. Puedes crear y ejecutar un contenedor a partir de una imagen utilizando el comando docker run nombre\_de\_la\_imagen. Puedes especificar opciones adicionales, como puertos, volúmenes y variables de entorno, para personalizar el comportamiento del contenedor.

Comandos Básicos:

* docker pull nombre\_de\_la\_imagen: Descargar una imagen de Docker desde Docker Hub.
* docker run nombre\_de\_la\_imagen: Ejecutar un contenedor a partir de una imagen.
* docker ps -a: Listar todos los contenedores (incluso los detenidos).
* docker stop ID\_del\_contenedor: Detener un contenedor en ejecución.
* docker rm ID\_del\_contenedor: Eliminar un contenedor.
* docker images: Listar las imágenes de Docker en el sistema.
* docker rmi nombre\_de\_la\_imagen: Eliminar una imagen de Docker.

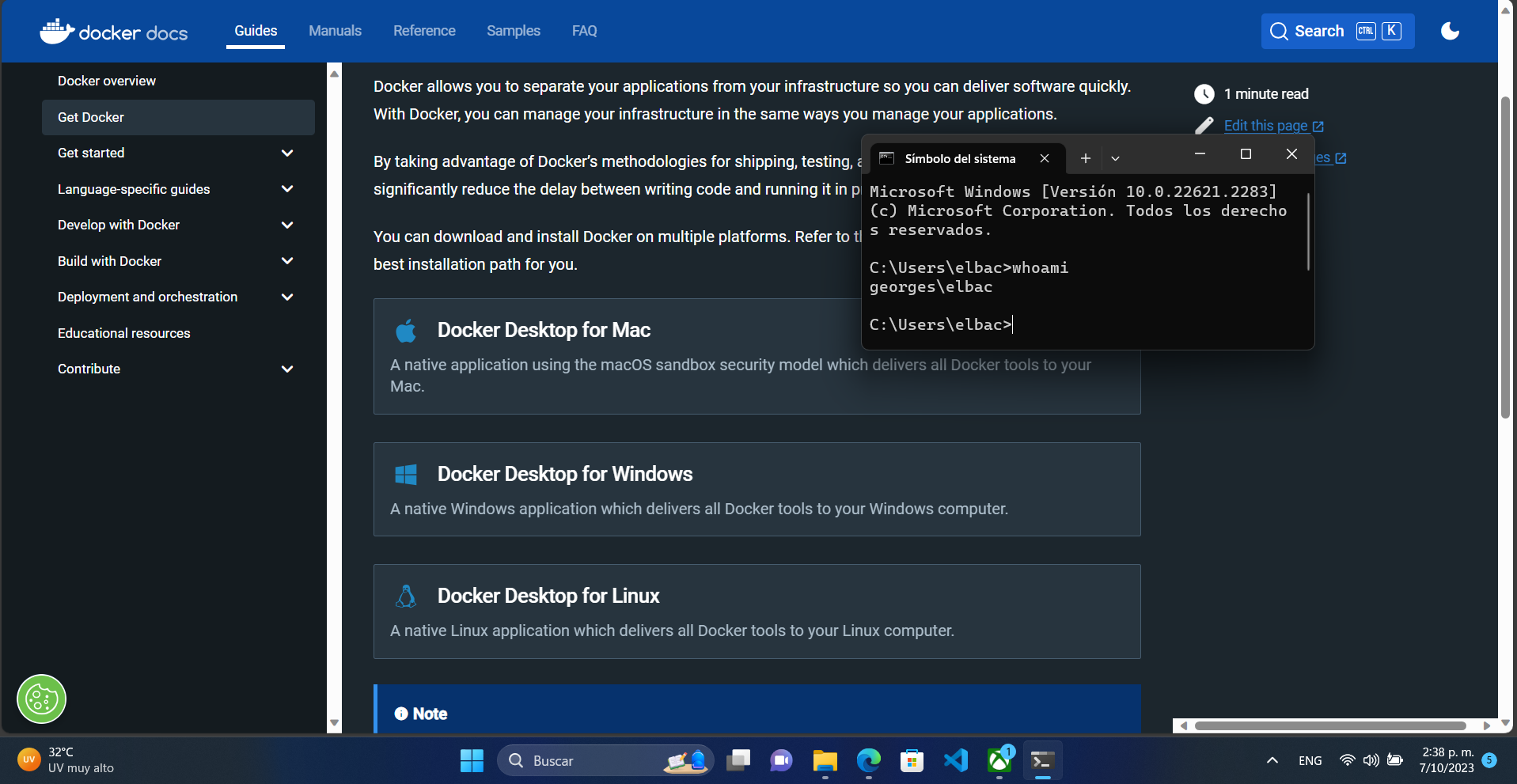
Docker Compose:

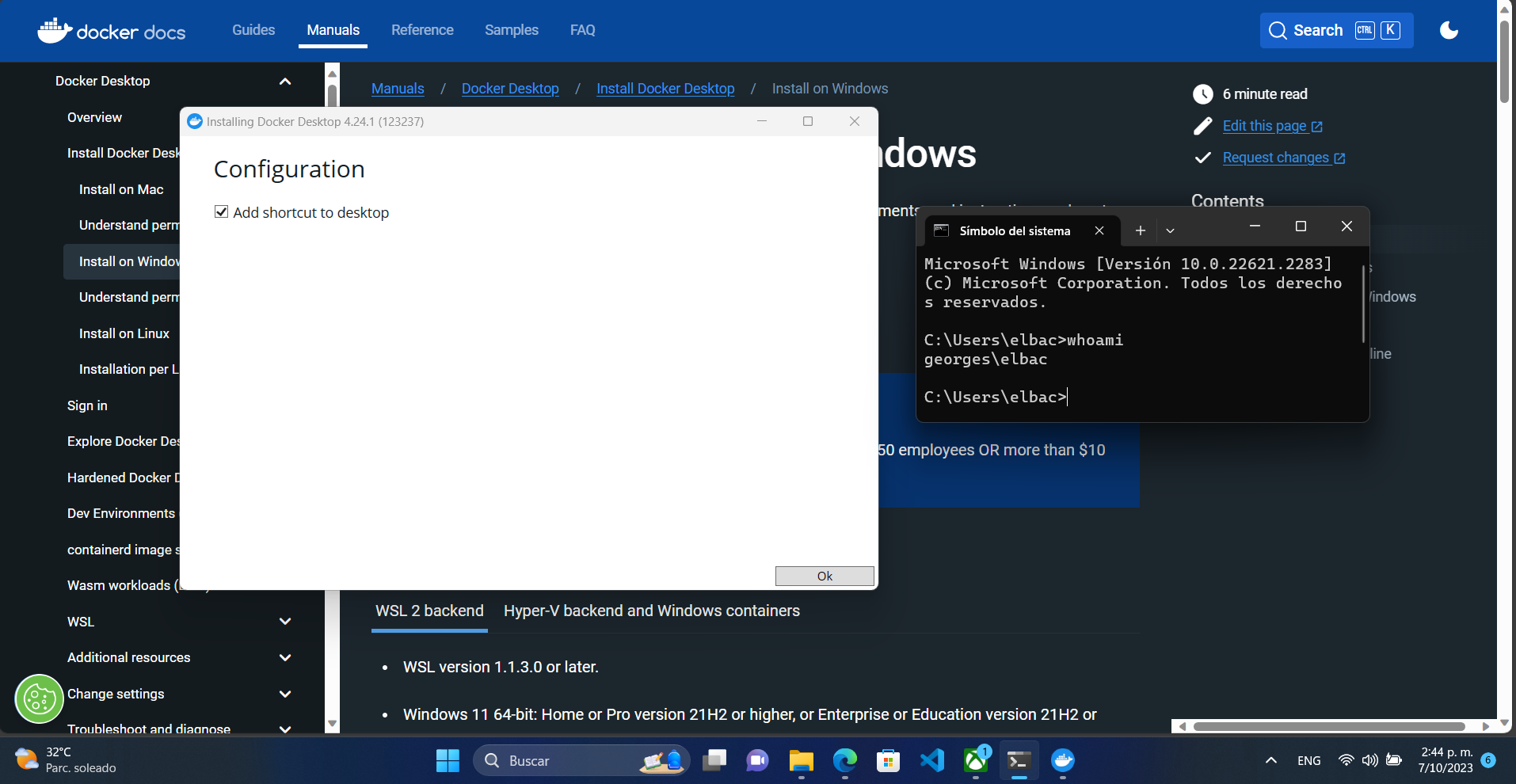
Docker Compose es una herramienta para definir y gestionar aplicaciones multi-contenedor. Permite describir las aplicaciones, sus dependencias, redes y volúmenes en un archivo docker-compose.yml.

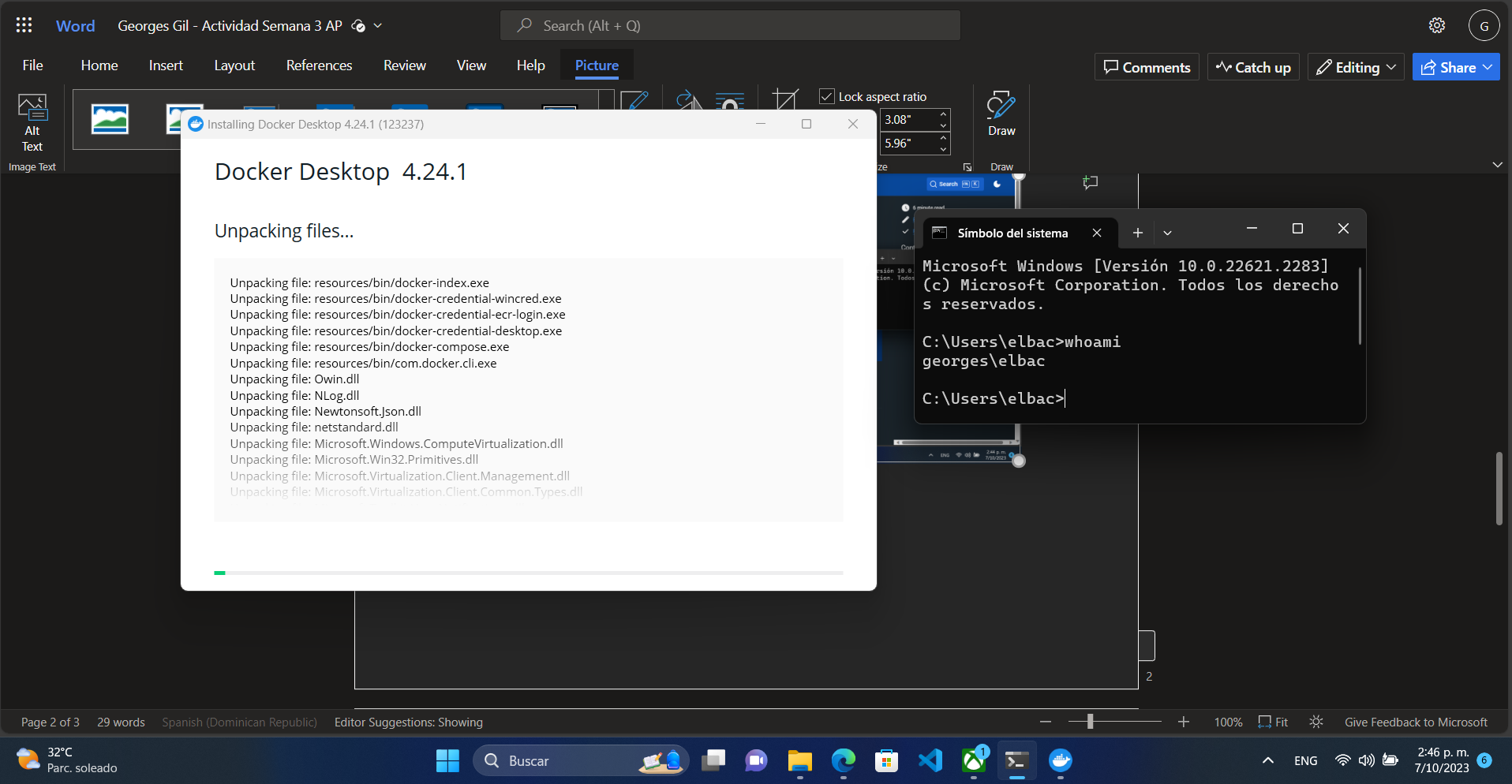
**Cluster o Balanceo de carga**

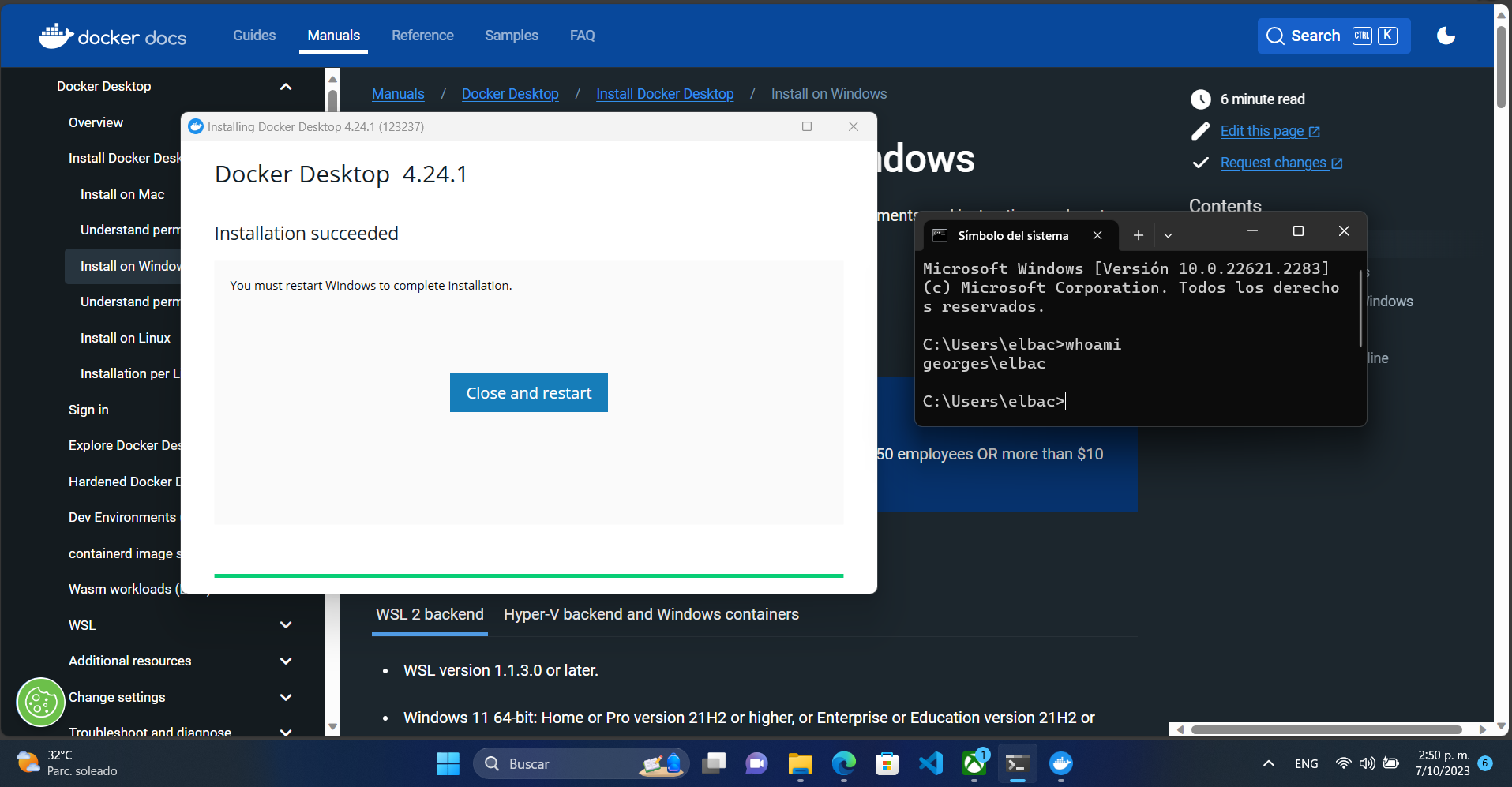
Consiste de un Front-1 (balanceador de carga), tres servidores web y un servidor GFS (Global File System) donde se almacenará la información. Para esto, se usa Swarm, un protocolo nativo de Docker. Convierte un conjunto de hosts en un solo host virtualizado de Docker. Algunas de sus características son: • Alta disponibilidad. • Balanceo de carga. • Descentralizado. • Escalabilidad

Instalación Docker









**Ejemplos de Docker-Compose**

version: '3'

services:

web:

image: nginx

ports:

- "80:80"

app:

image: node:alpine

volumes:

- ./app:/app

working\_dir: /app

command: "npm start"

depends\_on:

- db

db:

image: mariadb

environment:

MYSQL\_ROOT\_PASSWORD: example

MYSQL\_DATABASE: mydb

Node.js, Nginx y MariaDB con Docker Compose

Node.js: Este servicio utiliza una imagen de Node.js para ejecutar una aplicación Node.js. Monta el código fuente de la aplicación desde el directorio local ./app al contenedor y ejecuta el comando npm start.

Nginx: El servicio de Nginx utiliza la imagen oficial de Nginx y redirige el tráfico del puerto 80 del host al puerto 80 del contenedor.

MariaDB: Utiliza la imagen oficial de MariaDB y configura las variables de entorno para establecer la contraseña de root y el nombre de la base de datos.

**Ejercicio:**

Crear una imagen de una aplicación que se conecte a una base de datos(Otra imagen de docker), con lo siguiente criterios de tecnologías por rango de matrículas:

**Matriculas (1-18-\*)**

* GOLANG (APP)
* NGINX (Alojamiento)
* POSTGRESQL (BD)

1. Aplicación que haga uso de un **CRUD** de una tabla en su BD.
2. Alojamiento del servicio.
3. Base de datos con su usuario y contraseña es su matricula y nombre del alumno.
4. Subir código de su app a Github.
5. Mostrar aplicación en clases.

**Nota**: Uso de variables de entornos con Docker, uso de docker-compose y crear una imagen de su aplicación.

**Link Github:** [**AlgoritmosParalelos/Semana-03 at main · GeorgesGil/AlgoritmosParalelos (github.com)**](https://github.com/GeorgesGil/AlgoritmosParalelos/tree/main/Semana-03)

