**Docker**

**Introduction :**

Docker is one of the most popular container-based platforms attracting the attention of many development teams. More and more companies are switching to Docker due to its reliability, performance, and functionality.

Therefore, it is essential to understand this open-source containerization software and the underlying components powering it.

What is **Docker**?

Docker is an open-source containerization platform used for developing, deploying, and managing applications in lightweight virtualized environments called containers.

What Are **Containers**?

Docker Container is a standardized unit which can be created on the fly to deploy a particular application or environment. It could be an Ubuntu container, CentOs container, etc. to full-fill the requirement from an operating system point of view. Also, it could be an application oriented container like CakePHP container or a Tomcat-Ubuntu container etc.

## What Is Docker Used For?

Docker is used for:

* Running multiple workloads on fewer resources.
* Isolating and segregating applications.
* Standardizing environments to ensure consistency across development and release cycles.
* Streamlining the development lifecycle and supporting CI/CD workflow.
* Developing highly portable workloads that can run on multi-cloud platform.
* A cost-effective alternative to virtual machines.
* A version control system for an application.

## Docker Core Components :

## The tool consists of multiple components, each playing an important role in the platform.

### **Docker Engine :**

The Docker Engine (DE) is installed on the host machine and represents the core of the Docker system. It is a lightweight runtime system and the underlying client-server technology that creates and manages containers.

Docker Engine consists of three components:

* **Server**- the Docker daemon (dockerd), which is responsible for creating and managing containers.
* **Rest API** - establishes communication between programs and Docker and instructs dockerd what to do.
* **Command Line Interface (CLI)** - used for running Docker commands.

### **Docker Images :**

### A Docker image is a file used to execute code in a Docker container. Docker images act as a set of instructions to build a Docker container, like a template. Docker images also act as the starting point when using Docker. An image is comparable to a snapshot in virtual machine (VM) environments.

### **Dockerfile :**

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. This page describes the commands you can use in a Dockerfile.

### **Docker Hub :**

### Docker Hub is a service provided by Docker for finding and sharing container images with your team.

### **Docker Volumes :**

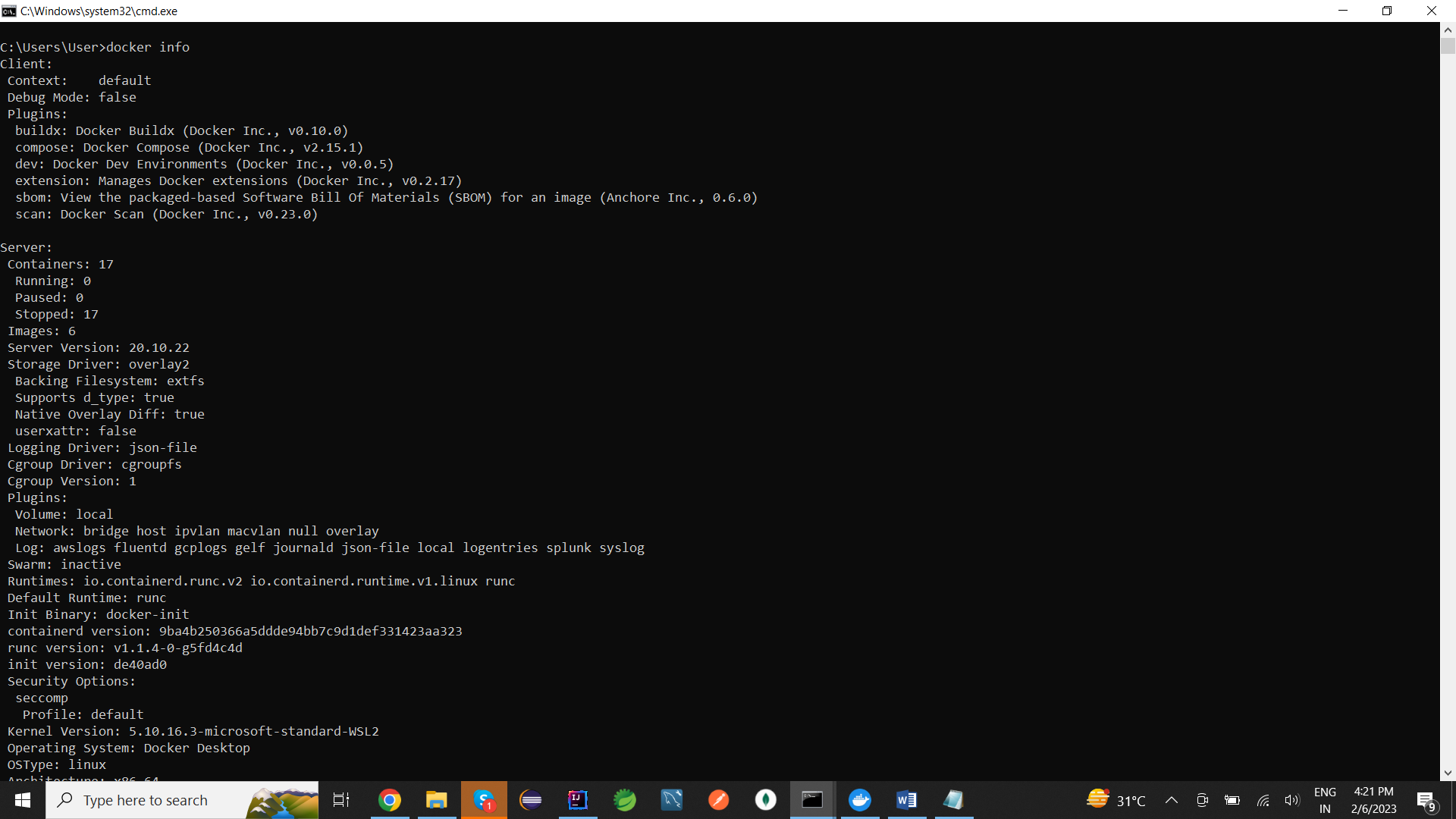
Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker.

### **Docker Desktop :**

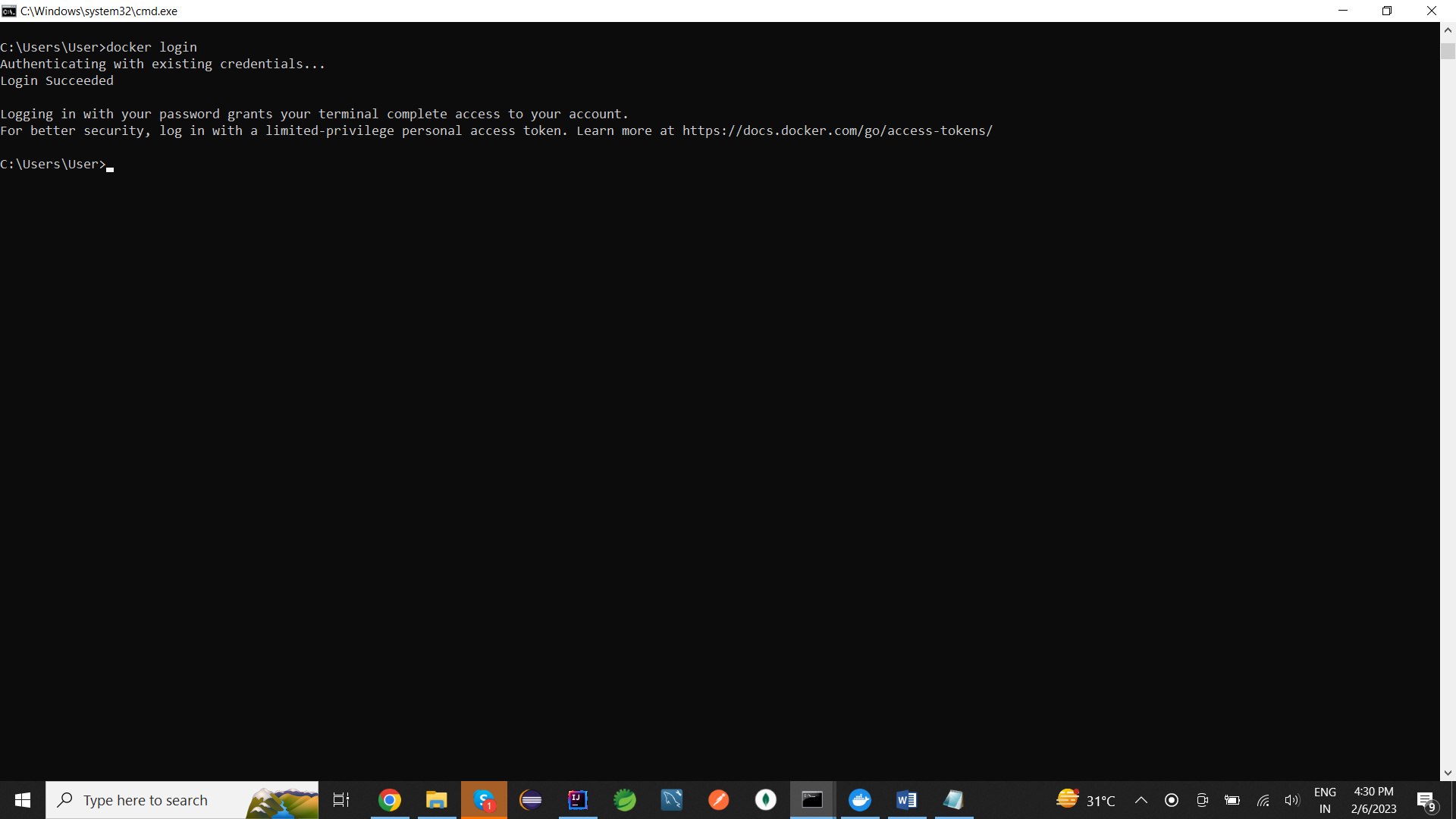
Docker Desktop is a one-click-install application for your Mac, Linux, or Windows environment that enables you to build and share containerized applications and microservices

### **Docker Basic Commands :**

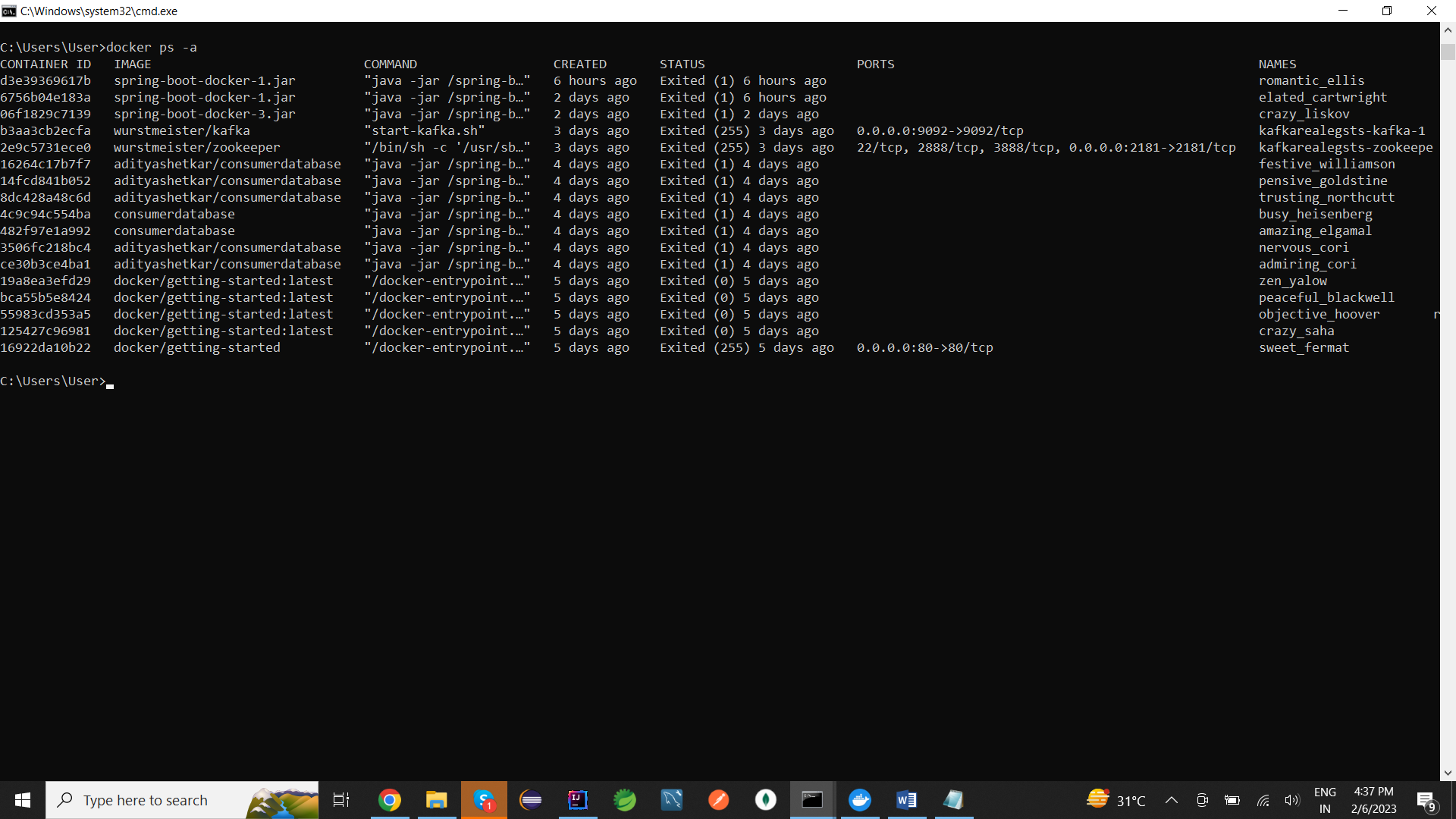
* docker info : To get all Docker related information from your machine.



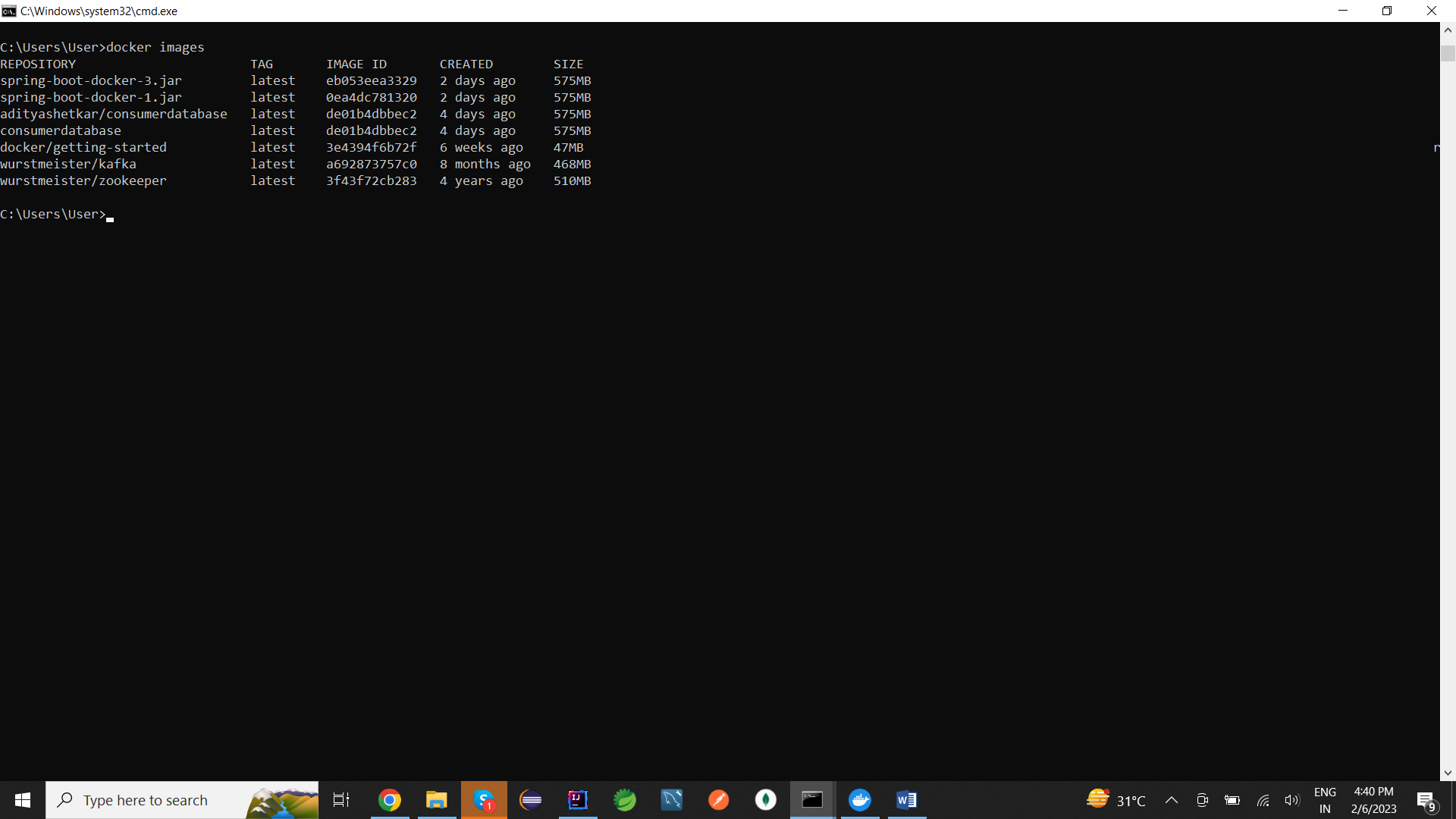
* docker login : To login to your Docker account



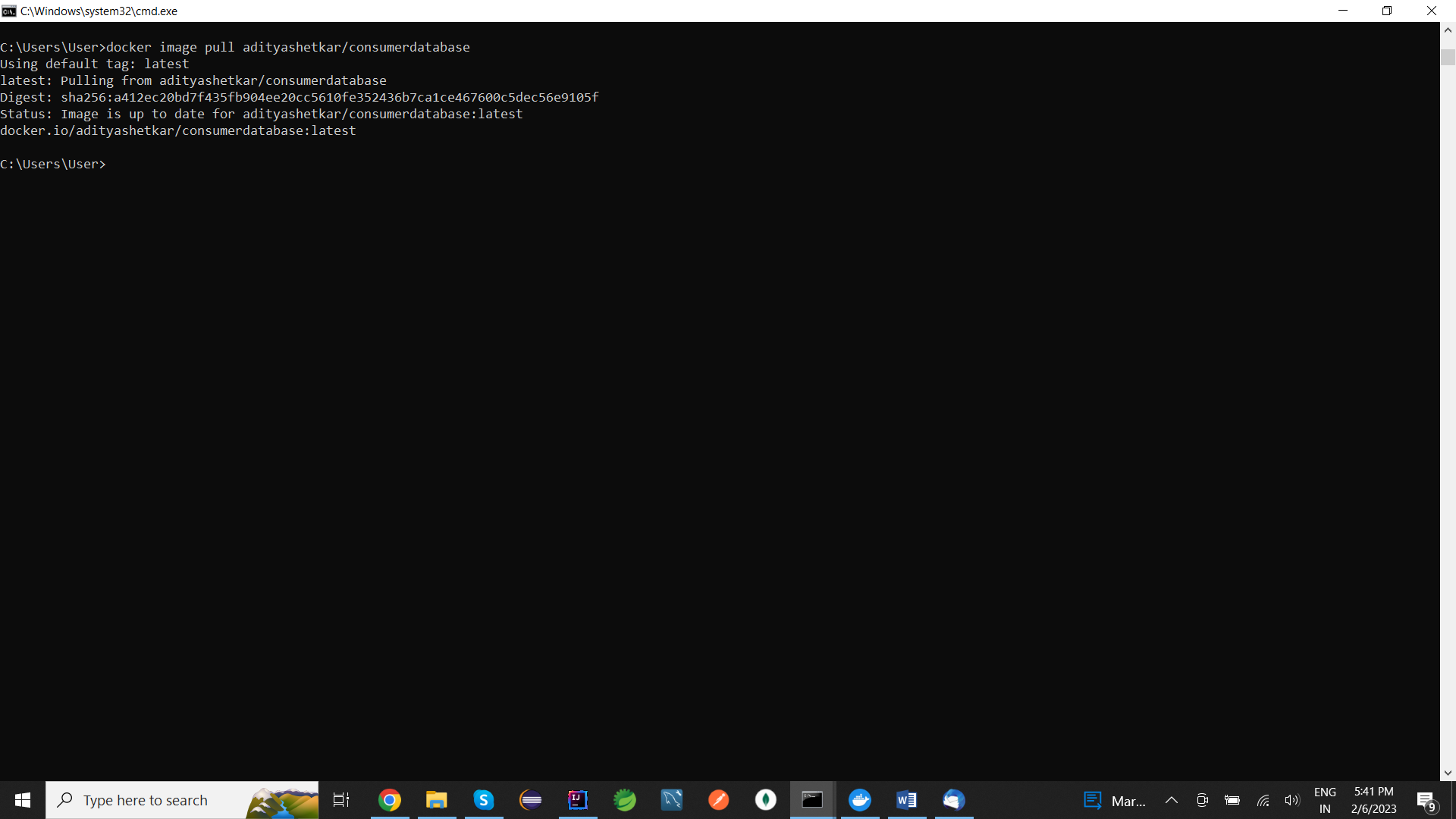
* docker ps –a : To get list of all containers



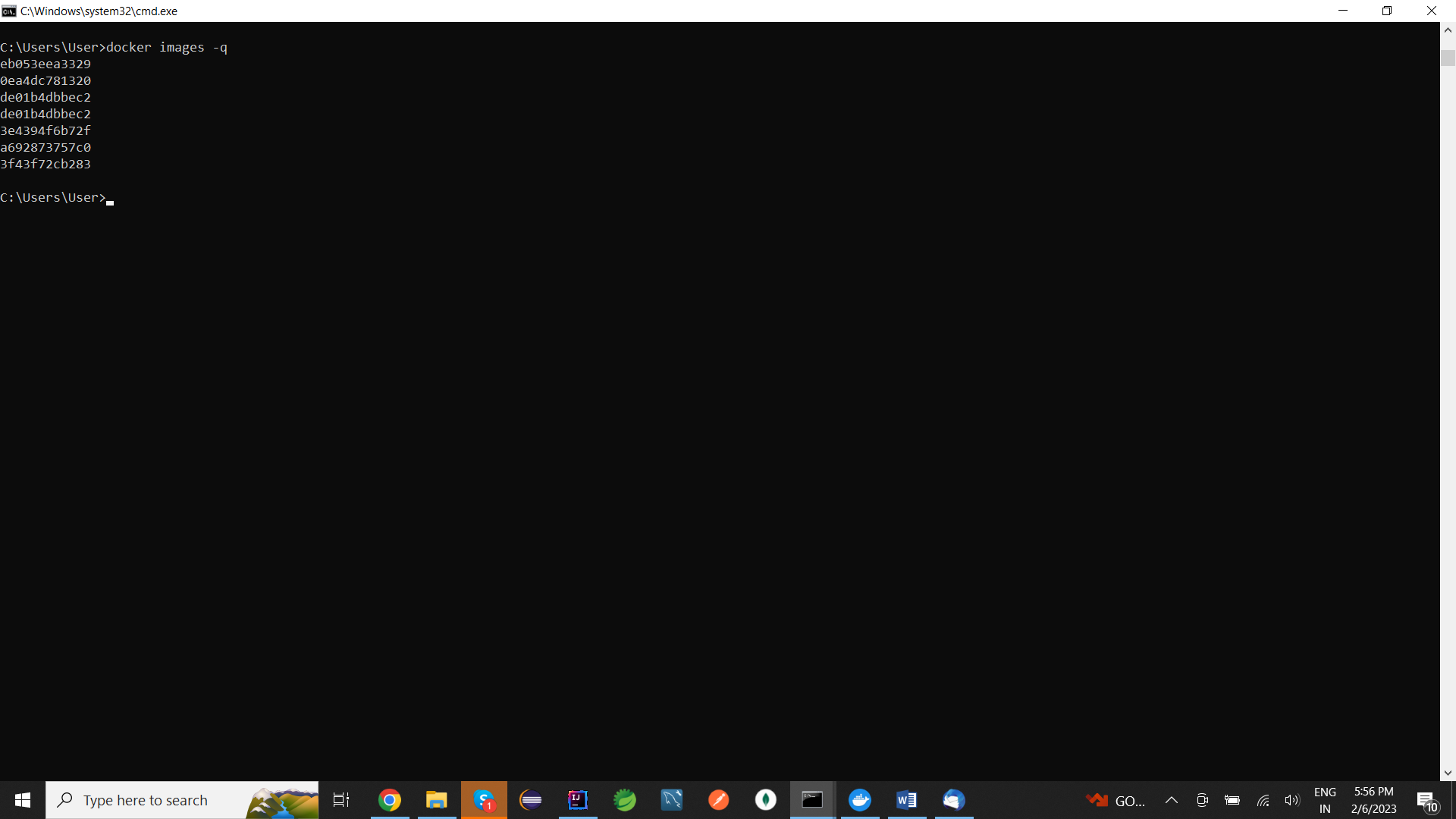
* docker images : To get all images



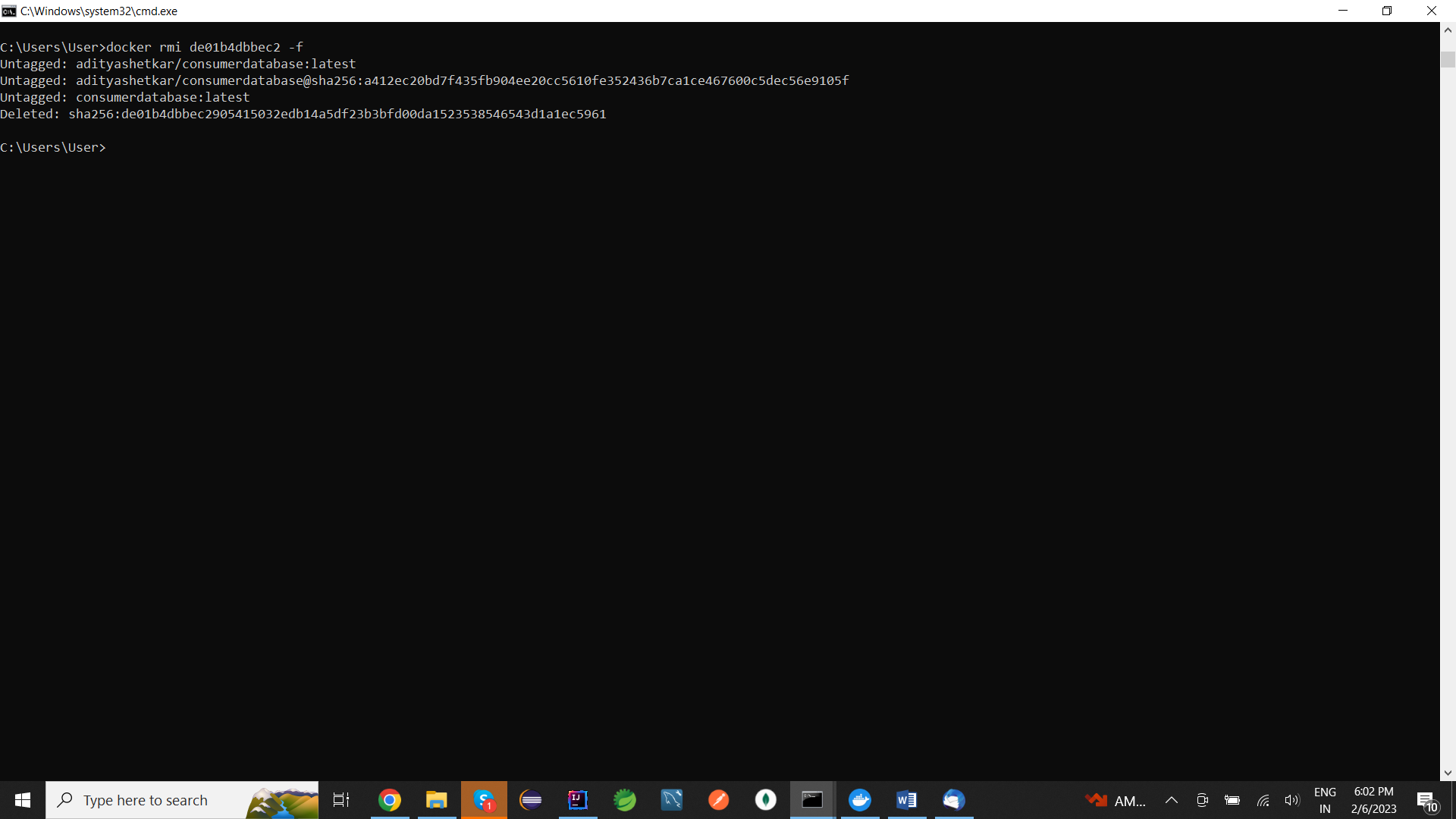
* docker image pull adityashetkar/consumerdatase : To pull the image from mentioned repository



* docker images –q : To get Id of all images



* docker rmi de01b4dbbec2 –f : To delete image with image Id



* java -jar spring-boot-docker.jar : To run the application by jar name

