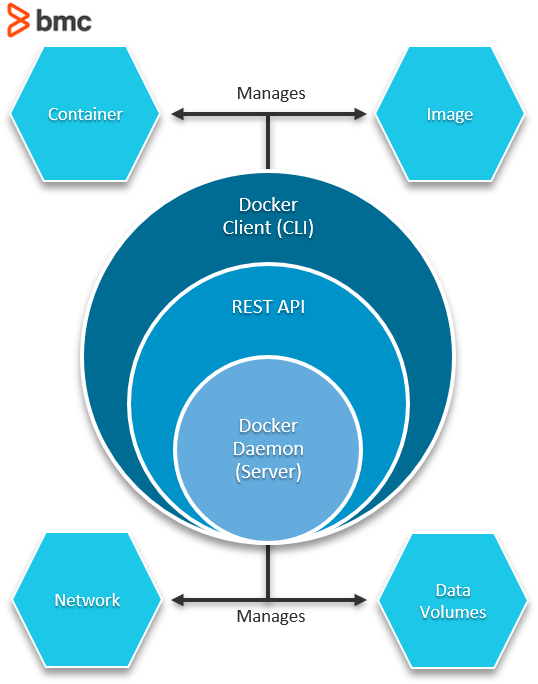
CONTAINER MANAGEMENT SYSTEM (DOCKER)

Introduction

Docker is one of the most popular tools for application containerization. Docker enables efficiency and reduces operational overheads so that any developer, in any dev environment, can build stable and reliable applications.

Docker is a Linux-based, open-source containerization platform that developers use to build, run, and package applications for deployment using containers. Unlike virtual machines, Docker containers offer:

* OS-level abstraction with optimum resource utilization
* Interoperability
* Efficient build and test
* Faster application execution



Components of a Docker architecture

Docker comprises the following different components within its core architecture:

* Images
* Containers
* Registries
* Docker Engine

Images

Images are like blueprints containing instructions for creating a Docker container. Images define:

* Application dependencies
* The processes that should run when the application launches

You can get images from Docker Hub or create your own images by including specific instructions within a file called Docker file.

Containers

Containers are live instances of images on which an application or its independent modules are run.

In an object-oriented programming analogy, an image is a class, and the container is an instance of that class. This allows operational efficiency by allowing to you to multiple containers from a single image.

Registries

A Docker registry is like a repository of images.

The default registry is the Docker Hub**,**a public registry that stores public and official images for different languages and platforms. By default, a request for an image from Docker is searched within the Docker Hub registry.

You can also own a private registry and configure it to be the default source of images for your custom requirements.

Docker Engine

The Docker Engine is of the core components of a Docker architecture on which the application runs. You could also consider the Docker Engine as the application that’s installed on the system that manages containers, images, and builds.

A Docker Engine uses a client-server architecture and consists of the following sub-components:

* **The** **Docker Daemon** is basically the server that runs on the host machine. It is responsible for building and managing Docker images.
* **The** **Docker Client** is a command-line interface (CLI) for sending instructions to the Docker Daemon using special Docker commands. Though a client can run on the host machine, it relies on Docker Engine’s REST API to connect remotely with the daemon.
* **A** **REST**[**API**](https://www.bmc.com/blogs/microservice-vs-api/) supports interactions between the client and the daemon.

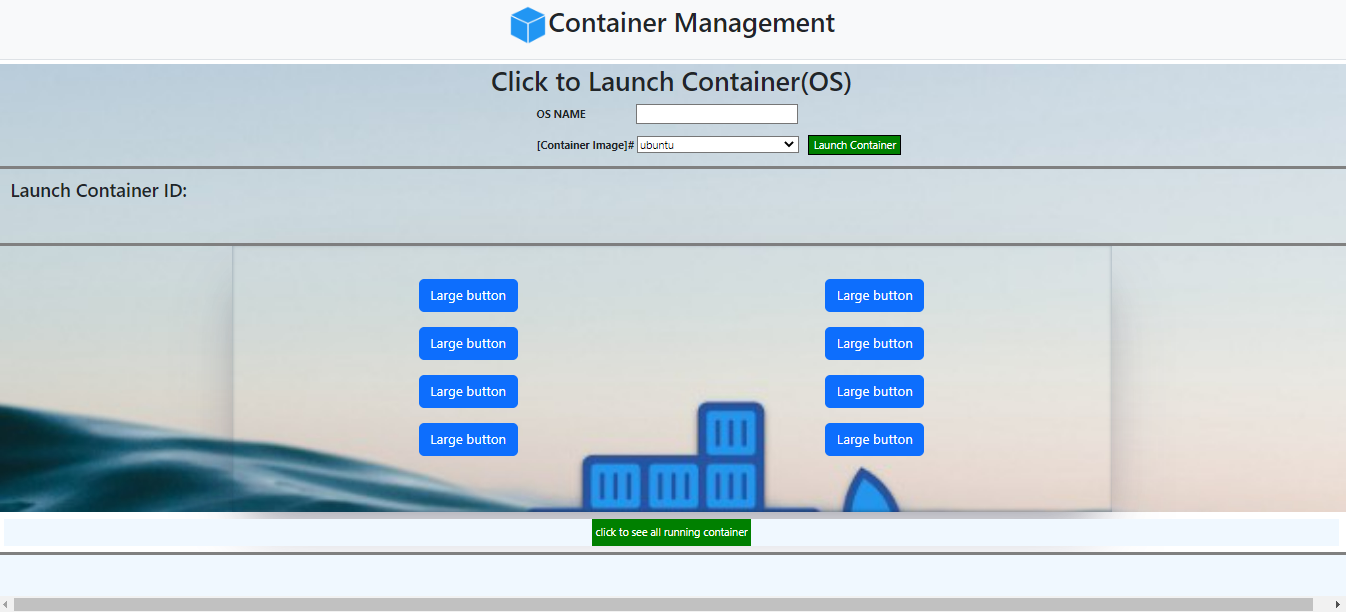
Existing System

First of all, we are having an existing system in the form of Linux terminal or bash to execute commands of docker, and the people who don’t know docker, plus who don’t know docker commands we are enhancing the system further to eliminate the time to remember plus a little or no knowledge is required to run all this from front end.



Proposed System

In the proposed system we are eliminating the user to remember the commands and execute from shell instead they can just use the front end and can run a command just by clicking the button, where it saves time as well as they can execute the command swiftly.



A screenshot of a computer

Description automatically generated

Future Work

We can more enhance this application by integrating command Linux terminal as well as we can use Kubernetes concept to implement the same as well and the system can be enhanced by more readiness as well-structured Front end as well as including Bash terminal from front end where user can experience the dashboard and integration of this commands to enhance the kubectl.

Contributors / Authors of Container Management

1. Anshul Pant - Team Lead (Express/UI/ Embedding)
2. Syed Shoeb Ahmed – (Ui/Express/Frontend/Embedding)
3. Anmol Saini -- (Versioning Control / Implementing Other Inputs)
4. Aditya ranote- (Versioning Control/Backend Issues)
5. Rahul Sehgal – (Versioning Control /Testing /Inputs to Html)

GITHUB REPOSITORY

https://github.com/container-management/mern\_2022\_project