Docker Assignment

1. Explain the difference between Monolithic and Microservice architecture.

Provide at least 5 key differences.

Give one real-world example of each.

Monolithic :-
Monolithic architecture , in which thr entire program is constructed as a single unit. This means that any changes or updates to the application require modifying and redeploying the entire monolith.

Microservices :-

A microservice results in an application designed as a set of small, independent services. The services are loosely couple with one another and communicate over the network. Each service is responsible for a single functionality or feature of the application and can be developed deployed and scaled independently.

	Monolithic	**Microservices**
Structure	A single, unified codebase containing all application functionality.	A collection of small, independent, loosely coupled services, each with its own database and codebase.
Dependencies	Components are tightly coupled and interdependent.	Components are independent and communicate through well-defined interfaces like APIs or message brokers.
Development	Easier to develop and manage for smaller, simpler projects.	Requires more specialized skills to manage the coordination between services, but enables agility and independent development.
Scaling	The entire application must be scaled, even if only one component needs more resources	Each service can be scaled independently, leading to more efficient resource usage.
Maintenance	Difficult to maintain and update as it grows, with changes potentially impacting large parts of the codebase.	Easier to manage and update because services are small and isolated.

What is the difference between Containers and Virtual Machines?

Explain in terms of OS, resources, performance, startup time, and scalability.

A virtual machine (VM) is a software-based, virtual version of a physical computer that runs an operating system and applications, allowing multiple operating systems and isolated environments to run on a single physical machine.

Containers are packages of software that contain all of the necessary elements to run in any environment. Containers make it easy to share CPU, memory, storage, and network resources at the operating systems level and offer a logical packaging mechanism.

	Virtual Machines	**Containers**
Os	Each VM runs a full guest OS on top of the host machine, creating a complete, isolated virtual computer.	Share the host OS's kernel, providing isolation at the user-space level. This allows for running different applications and libraries without needing a separate OS for each container.
Resources	Resource-intensive because each VM includes its own operating system, virtual hardware, and applications, consuming more CPU, memory, and storage.	Lightweight and resource-efficient, as they only package the application and its dependencies, avoiding the overhead of a full guest OS.
Performance	Suffer from performance overhead due to running a separate OS and emulating hardware, though they can provide superior isolation and stability for certain workloads.	Offer high performance due to their efficient resource utilization and minimal overhead, as they directly access the host OS kernel.
Startup time	Take significantly longer to start, potentially minutes, due to the need to boot and initialize their own full guest operating system and virtual hardware.	Start almost instantly, often within seconds, because they only need to spin up the application processes rather than booting an entire OS.
Scalability	Less efficient for high-scale deployments due to their larger resource footprint per instance, though they offer powerful virtualization for resource-intensive applications like machine learning and software-defined networks.	They are well-suited for microservices architecture, allowing individual components of an application to be scaled independently.

3. What is Docker Architecture?

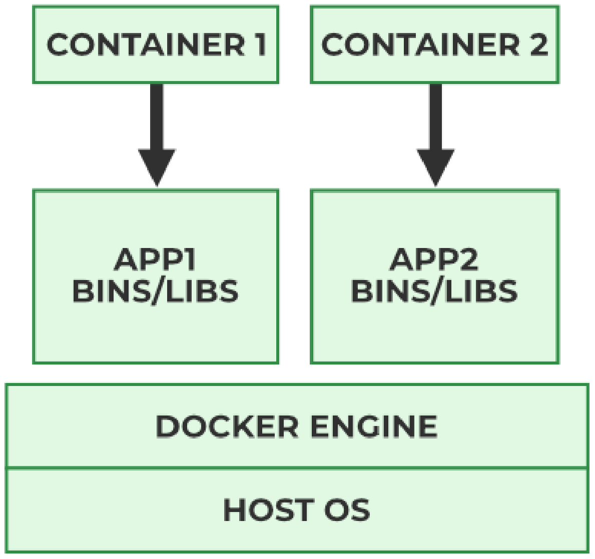
*** Explain the roles of:**

*** Docker Engine**

*** Docker Daemon**

*** Docker Client**
*** Docker Registry**
*** Images & Containers**

Docker's architecture operates on a client-server model and comprises several key components that facilitate the building, running, and management of containers.



Docker Architecture

Docker Engine
The Docker Engine is the core component that enables Docker to run containers on a system. It follows a client-server architecture and is responsible for building, running, and managing Docker containers.

Docker Daemon
Docker Daemon acts as the central component responsible for managing and orchestrating all Docker objects, including images, containers, networks, and volumes.

Docker Client
The Docker client is the command line interface for interacting with the underlying Docker Ecosystem. With the docker client, users can execute their commands to mange their docker containers, images, networks, volumes and other docker objects.

Docker Registry

Docker Registry is a centralized storage and distributed system for collecting and managing Docker images. It acts as a server-side application that stores, manages, and distributes container images across environments

Write the command to check Docker version and Docker info?

For versions

```
docker  -v
docker –version
```

for info

```
docker info
```