LIVE

Introduction to Container Fundamentals

Nived Velayudhan

docker

OPENSHIFT

cri-o

podman

buildah

kubernetes

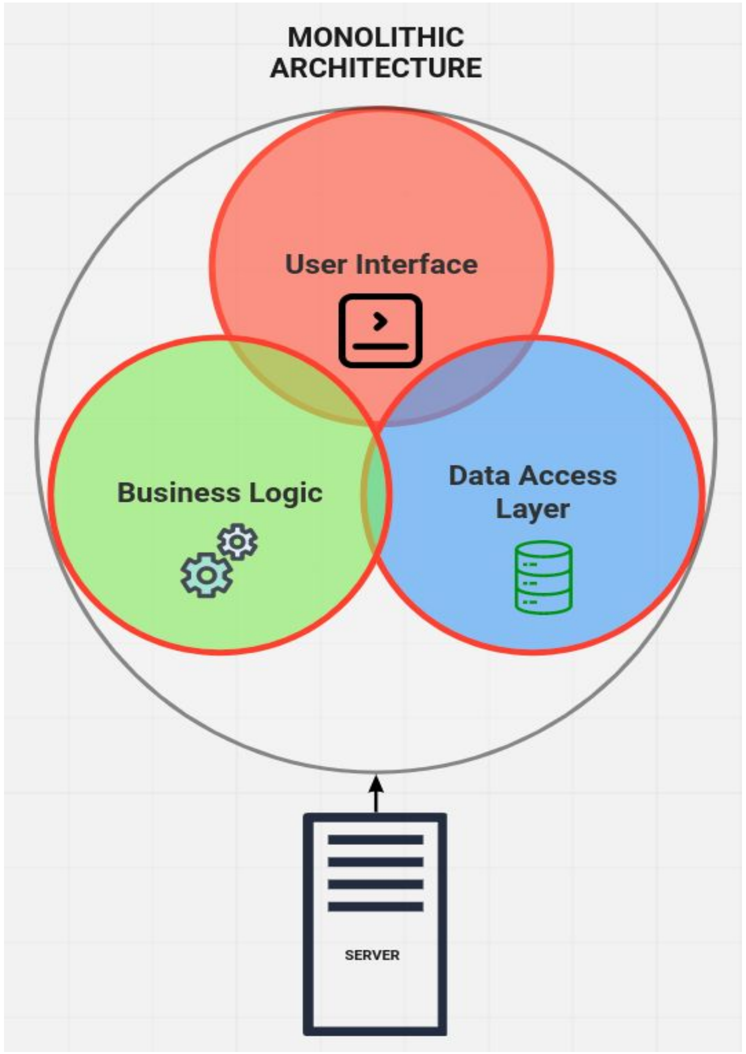
10 Aug 2021, 8.30 PM SGT/MYT, 6.00 PM IST

youtube.com/techbeatly

User Interface:

This is the presentation layer with which the user interacts. (CSS styles, HTML, JavaScript code)

The main function of the presentation layer is to display information and interpret user input commands, transforming them into appropriate operations in the context of business logic.



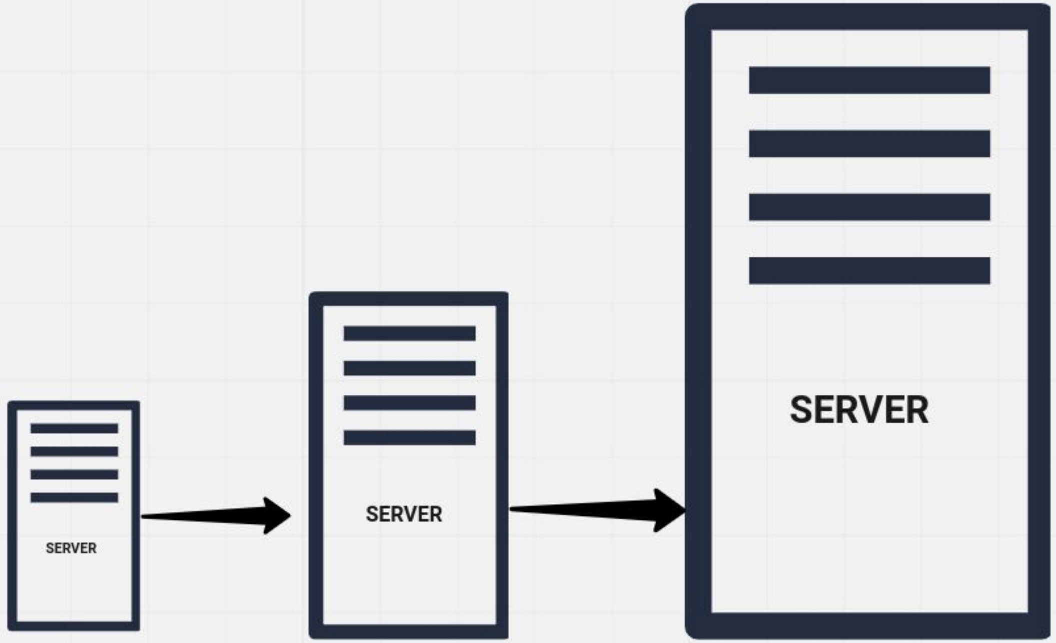
Business logic layer:

This comprises the set of components responsible for processing data received from the presentation layer. (Java EE , ASP.NET etc)

Data Access Layer:

Is responsible for monitoring transactions and maintaining a consistent data state. (SQL, PostgreSQL etc)

VERTICAL SCALING



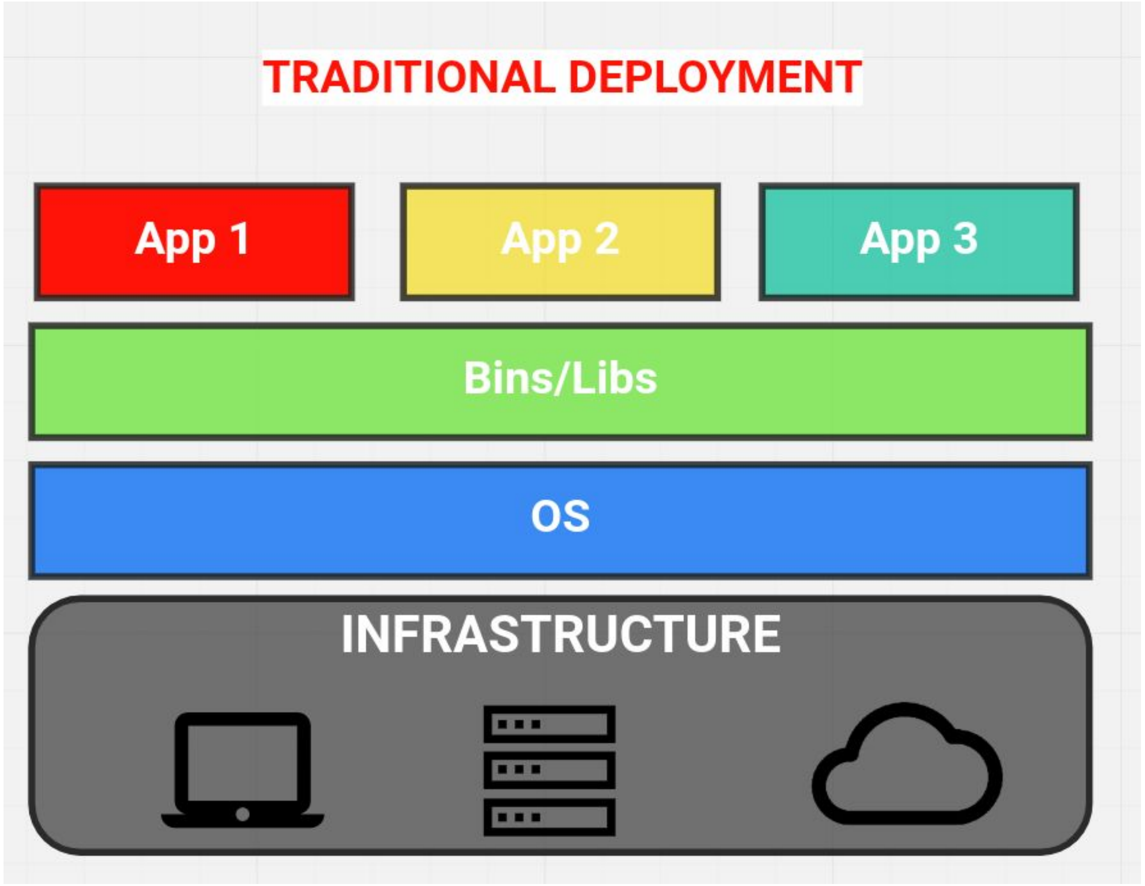
MORE CPU, MORE RAM, MORE DISK

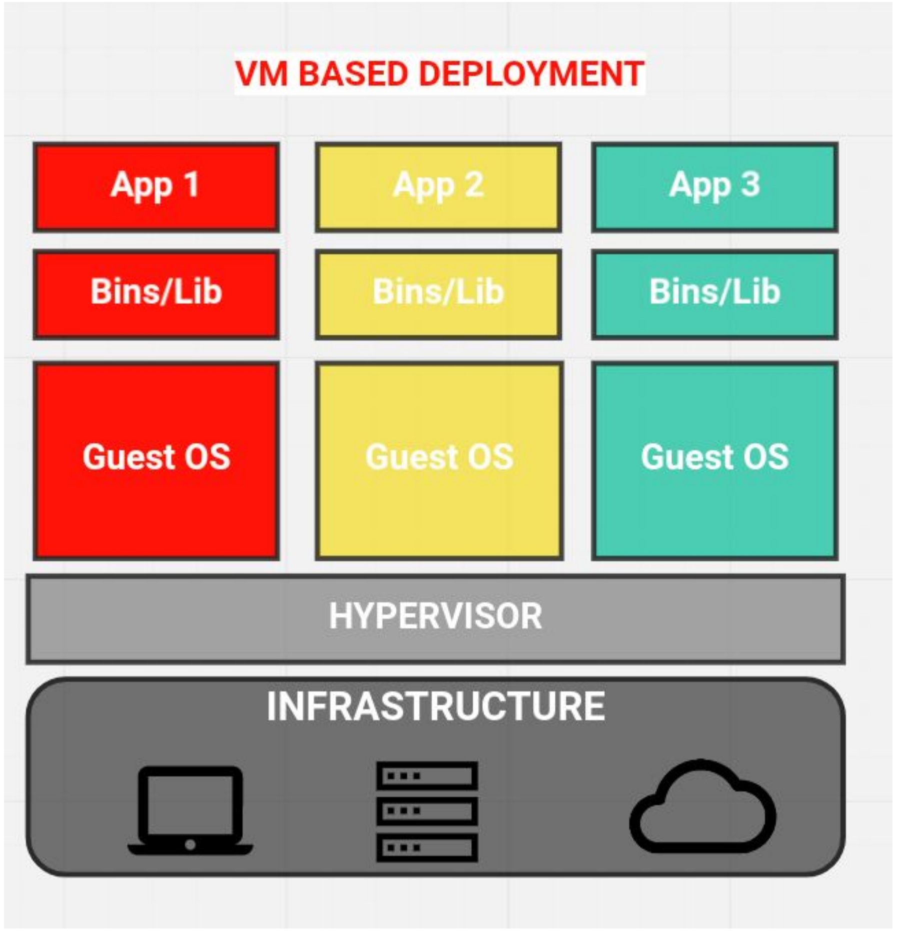
Advantages of Monolithic Architecture

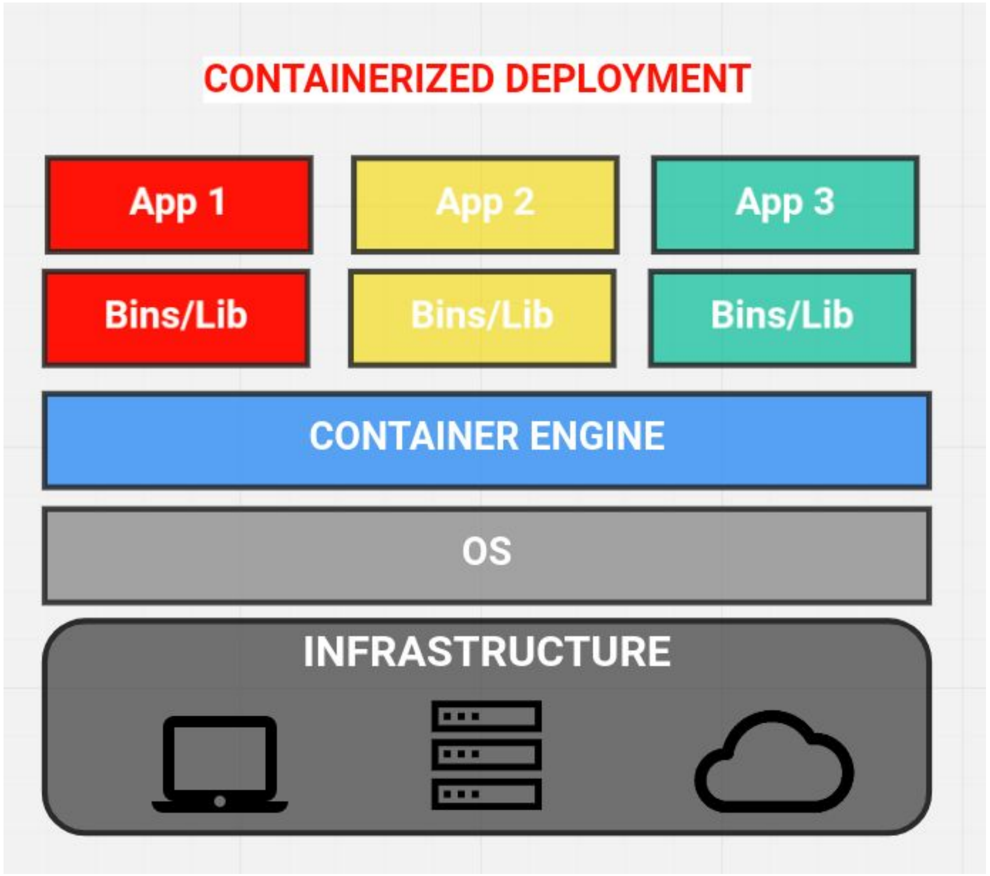
- **Less cross-cutting concerns:** One application means easier logging, handling, caching, and performance monitoring.
- **Easier debugging and testing:** Since a monolithic app is a single indivisible unit, you can run end-to-end testing much faster.
- **Simple to deploy:** Need to handle only 1 file or directory
- **Simple to develop:** Standard way of building applications

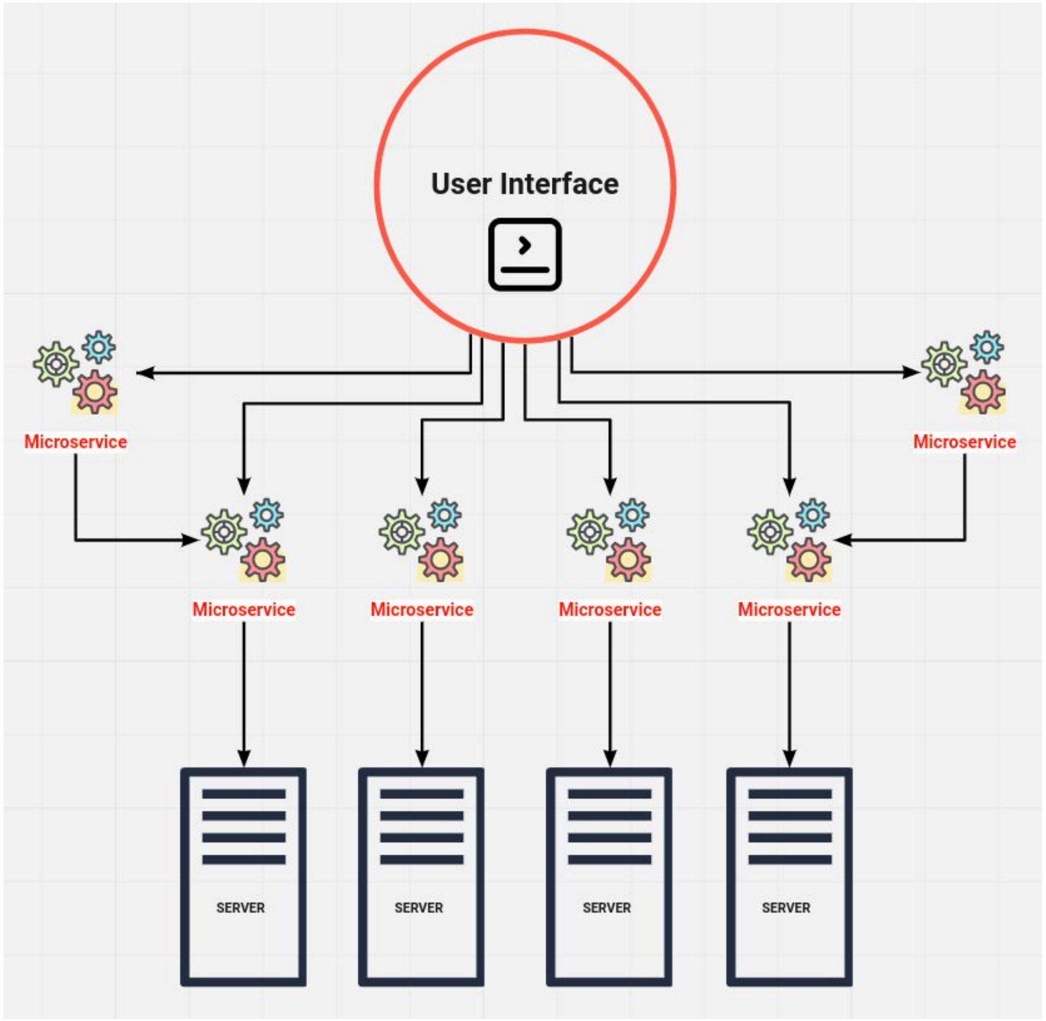
Disadvantages of Monolithic Architecture

- It becomes too large in size with time and hence, difficult to manage.
- We need to redeploy the whole application even for a small change.
- As the size of the application increases, its start-up and deployment time also increases.
- For any new developer joining the project, it is very difficult to understand the logic of large Monolithic application even if his responsibility is related to a single functionality.
- It is not very reliable as a single bug in any module can bring down the whole monolithic application.


TRADITIONAL DEPLOYMENT
App 1
App 2
App 3
Bins/Libs
OS
INFRASTRUCTURE


VM BASED DEPLOYMENT
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
HYPERVISOR
INFRASTRUCTURE


CONTAINERIZED DEPLOYMENT
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
CONTAINER ENGINE
OS
INFRASTRUCTURE


User Interface
Microservice
Microservice
Microservice
Microservice
Microservice
Microservice
SERVER
SERVER
SERVER
SERVER

Advantages of Microservice Architecture

- **Independent components:**
  - All the services can be deployed and updated independently, which gives more flexibility.
  - A bug in one microservice has an impact only on a particular service and does not influence the entire application.
  - It is much easier to add new features to a microservice application than a monolithic one.

- **Easier understanding:** Split up into smaller and simpler components, a microservice application is easier to understand and manage.

- **Better scalability:** Each component can be scaled independently. Therefore, the entire process is more cost and time-effective.

Disadvantages of Microservice Architecture

- **Complexity**: Building Microservices require profound expertise both from domain modelling and also DevOps team.

- **System distribution:** A microservices architecture is a complex system of multiple modules and databases so all the connections have to be handled carefully.

- **Cross-cutting concerns:** When creating a microservices application, you will have to deal with a number of cross-cutting concerns. They include externalized configuration, logging, metrics, health checks, and others.

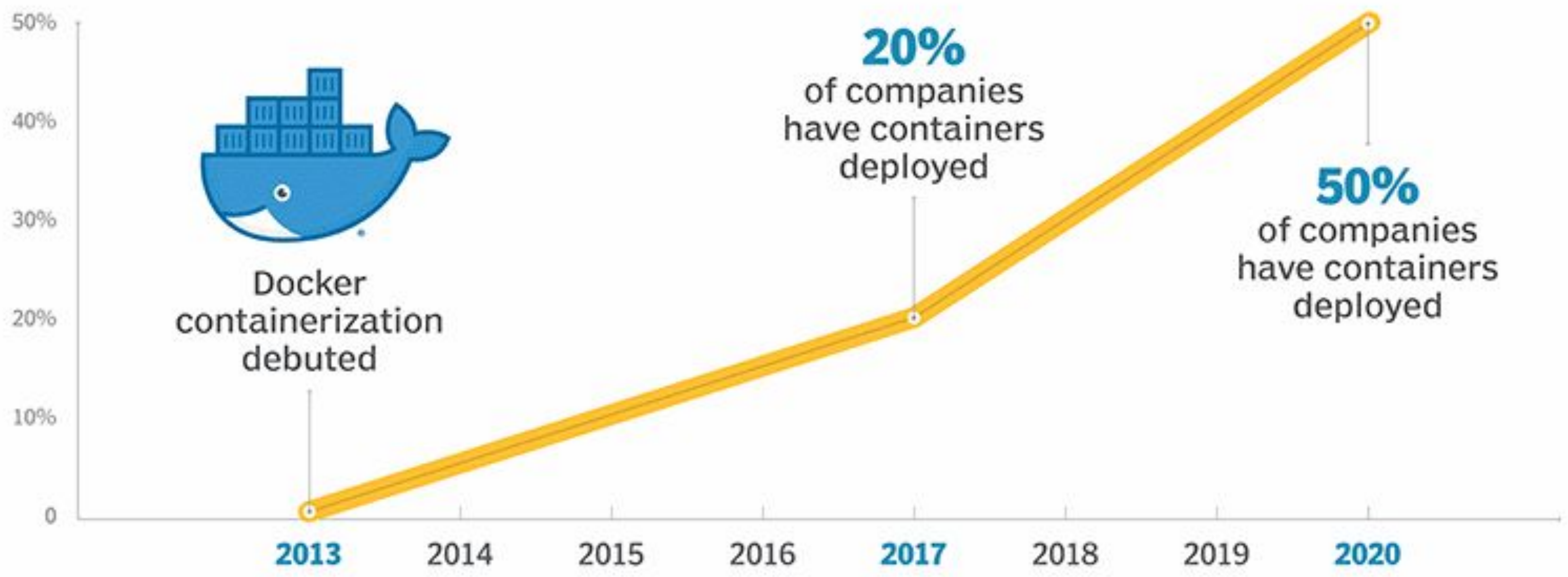
- **Testing:** A multitude of independently deployable components makes testing a microservices-based solution much harder.

LXC

- Release - August, 2008
- Virtualization solution on the operating system-level that enables the creation and operation of many isolated Linux virtual environments (VE)
- LXD provides a privileged daemon which exposes a REST API over local unix sockets and over network (if enabled).
- Integrates with OpenStack

- Release - March, 2013
- Server/Client Architecture
- Docker Daemon responsible for building images and executing containers.
- The daemon requires root privileges.
- Open Standards Compliant (OCI)
- Very Easy to use

Containerization timeline



SOURCE: GARTNER

podman

buildah

Note: When Kubernetes moved to CRI-O based on the OCI runtime specification, there was no need to run a Docker daemon and, therefore, no need to install Docker on any host in the Kubernetes cluster.

- Podman and Buildah came out due to issues where using Docker with its releases of things like swarm were breaking Kubernetes and to address how to **improve security of containers.**
- Is a ***daemon-less*** container engine for developing, managing, and running OCI Containers on your Linux System.
- Containers can either be run as root or in ***rootless*** mode.
- Open Standards Compliant (**OCI**)
- **Buildah** can control the layers of the container. Eg. Single layer versus 12 layers.

kubernetes

rkt

OPENSHIFT

Advantages of Containers

- **Containers are lightweight:** Containers require fewer system resources than traditional or hardware virtual machine environments because they don’t include operating system images.
- **Containers are portable:** A container wraps up an application with everything it needs to run, like configuration files and dependencies. This enables you to easily and reliably run applications on different environments such as your local desktop, physical servers, virtual servers, and public or private clouds
- **Better resource utilization:** Since containers do not require a separate operating system, they use up fewer resources.

Advantages of Containers

- **Faster Application Development:** Containers allow applications to be more rapidly deployed, patched, or scaled.
- **Faster Deployment:** Since containers wrap up an entire application and its dependencies, it is not dependant on the environment in which it was created thus making testing and debugging less complicated and less time-consuming
- **Supports modern development and architecture:** Due to its low size, consistency, and portability, it is an ideal fit for a modern development framework – DevOps, Serverless & Microservices.

Container Use Cases

- **IoT devices:** Since container images are small portable packages and are very easy to install, update, and require minimal processing power, they are ideal for IoT devices.
- **Containers as a Service (CaaS):** Container engines, orchestration, and underlying computer resources delivered as a service through cloud vendors.
- **Microservices:** Containers are used to decouple monolithic applications into reusable components by building container images.
- **DevOps support:** Container technology supports streamlined build, test, and deployment from the same container images.

QUESTIONS ??