Most of the big companies and start-ups initially build their systems using monolithic architecture to set up business faster and get moving. After some time due to mature in projects and rapid growth the code gets complicated which leads to complex architecture to maintain it. At the same time, business may lose speed, flexibility and agility which makes it harder to react to the market needs.

There comes the microservice approach to answer problems of large and complex IT systems to develop their applications as a suite of small services which communicate each other via API gateways.

Benefits of micro services are they can be developed, scaled and deployed independently by specialized teams. Changes to a particular microservice can be done and deployed separately without disturbing the other parts of the system.

**Understand Containers and their Uses:**

**Container:**

A Container is an operating system virtualization form.

It contains only the necessary executables, binary code, libraries, and configuration files that are used to run anything from a small microservice or software process to a larger application.

It is a lightweight and portable since they do not contain operating system images.

**Uses:**

Containers are used to build, test, deploy, and redeploy applications on multiple environments from a developer’s local laptop to an on-premises data centre and even on the cloud.

**Benefits of containers include**

**Less overhead: because they use less system resources**

**Increased portability: applications in containers can be deployed in multiple platforms**

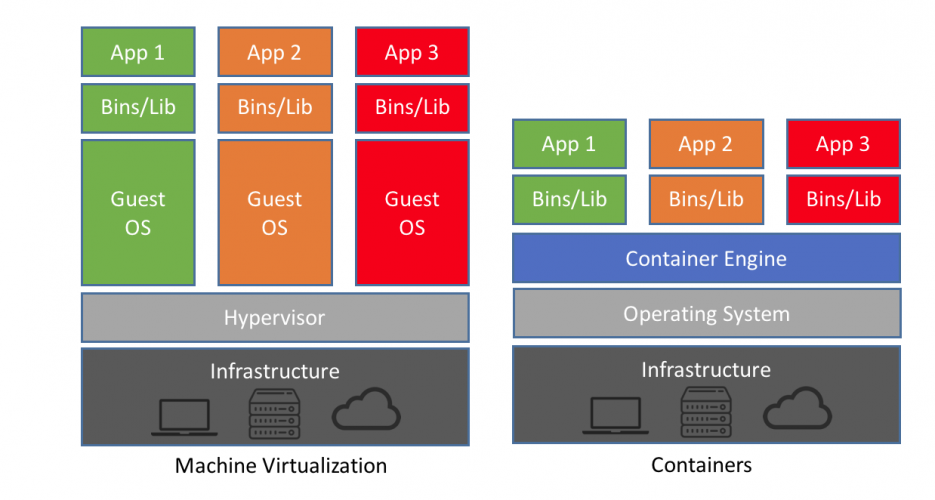
**consistent operation: applications run same regardless of where they are deployed**

**Greater efficiency: applications can be deployed, patched, scaled rapidly**

**Better application development: because containers support Agile and DevOps**

**Containerization Vs Virtualization:**

**Although there are some basic similarities, containers are very different from VMs.**



|  |  |
| --- | --- |
| **Virtualization** | **Containerization** |
| **Isolation OS** | **Isolation container** |
| **Every file is installed** | **Only required bin & lib files are installed** |
| **Take time to boot** | **Take less time to boot** |
| **Take time to restart VM** | **Take very less time to restart container** |
| **Space is wasted** | **Space is not wasted** |

**Docker:**

* Docker is a toolset that helps you package up and distribute your applications in a portable and repeatable way
* It is an open-source light-weight containerization technology
* It helps to run multiple OS on the same host
* It allows to automate deployment of applications in light weight & portable containers
* Developers use Docker to set up and run applications without the headache of having to install everything manually
* Operators use Docker to distribute applications across environments consistently and optimize server resources
* Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely, and with confidence to both Linux and Windows servers.

****

There are three components in the Docker Engine:

Server: It is the docker daemon called dockerd. It can create and manage docker images. Containers, networks, etc.

Rest API: It is used to instruct docker daemon what to do.

Command Line Interface (CLI): It is a client which is used to enter docker commands.

**Commonly used Docker terminology**

**Docker Universal Control Plane:**

It is the enterprise-grade cluster management solution from Docker. It manages your whole cluster from a single place.

**Docker Swarm:**

It is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host

**Docker Cloud:**

It is a hosted service that provides a Registry with build and testing facilities for Dockerized application images, tools to help you set up and manage your host infrastructure, and deployment features to help you automate deploying your images to your infrastructure.

**Docker Hub:**

It is a cloud-hosted service from Docker that provides registry capabilities for public and private content.

**Docker file:**

It is a text document that contains all the commands a user could call on the command line to assemble an image.

**Docker Compose:**

It is a tool for defining and running multi-container Docker applications.

**Docker Kitematic:**

It is an open-source project built to simplify and streamline using Docker on a Mac or Windows.

This tool automates the Docker installation and setup process and provides an intuitive graphical user interface (GUI) for running Docker containers.

**Docker CLI Commands:**

# docker images 🡪 list images

# docker pull Ubuntu 🡪 pull an image or a repository to the registry

# docker run –it –rm Ubuntu /bin/bash 🡪 run a command in a new container

# docker ps 🡪 list running containers

# docker ps –a 🡪 list stop or passed containers

# docker start <container id> 🡪 start one or more containers

# docker inspect <container id> 🡪 gives complete information

# docker run –it –name

When you develop a web, web API and sql services for an application, the options to deploy are on your local host or use Paas service on azure which internally launches a VM with required runtime and version. Have you ever tried the same app to run on another computer to find out that somethings missing and it just won’t work? Perhaps it might be a .dll file or a particular framework.

Also in an another scenario most of the big companies and start-ups initially build their systems using monolithic architecture to set up business faster and get moving. After some time due to mature in projects and rapid growth the code gets complicated which leads to complex architecture to maintain it. At the same time, business may lose speed, flexibility and agility which makes it harder to react to the market needs.

Here comes the concept of containers and microservice kind of approach to answer problems of large and complex IT systems to develop their applications as a suite of small services which communicate each other via API gateways.

Understanding Containers and their benefits:

Container:

A Container is an operating system virtualization form.

It contains only the necessary executables, binary code, libraries, and configuration files that are used to run anything from a small microservice or software process to a larger application.

It is a lightweight and portable since they do not contain operating system images.

Benefits:

Less overhead: because they use less system resources

Increased portability: applications in containers can be deployed in multiple platforms

consistent operation: applications run same regardless of where they are deployed

Greater efficiency: applications can be deployed, patched, scaled rapidly

Better application development: because containers support Agile and DevOps.

Who will do all this for you? Here comes the concept of Docker.

Docker:

Docker is an open-source light-weight containerization technology that helps you package up and distribute your applications in a portable and repeatable way.

The way to build the container and run the app is to use the Docker file.

Docker file:

It is a text document that contains all the commands a user could call on the command line to assemble an image.

What if there are multiple applications need to be run in containers?

Docker Compose:

It is a tool for defining and running multi-container Docker applications.

Then how to manage these multiple containers?

Kubernetes:

It is an open-source container management software which helps to manage your containers like auto scaling, load balancing, deployments, roll backs etc.