What are Containers

Description

Have you ever faced a scenario when your developed application works on your machine or a particular environment, but does not run on your colleagues machine or any higher/production environment? Well personally I have experienced this ðŸ™, If its a web app for examples, either images might not get loaded properly or some feature does not work properly and is a bit glitchy due to dependency issue. Then there is a debate between the Developers and Testers or between the Developers itself that, “But! but! it works on my machine/environment” ðŸ™• . The issue can be anything, either the dependency version might be different across machines/environment or there is a discrepancy in certain configuration or OS libraries across environments which is unknown.

To address the above problems Docker comes to our rescue. In out next post we will learn about Docker in detail. But before that lets learn about Containers and the existing VM technology which addresses the so called environment issue.

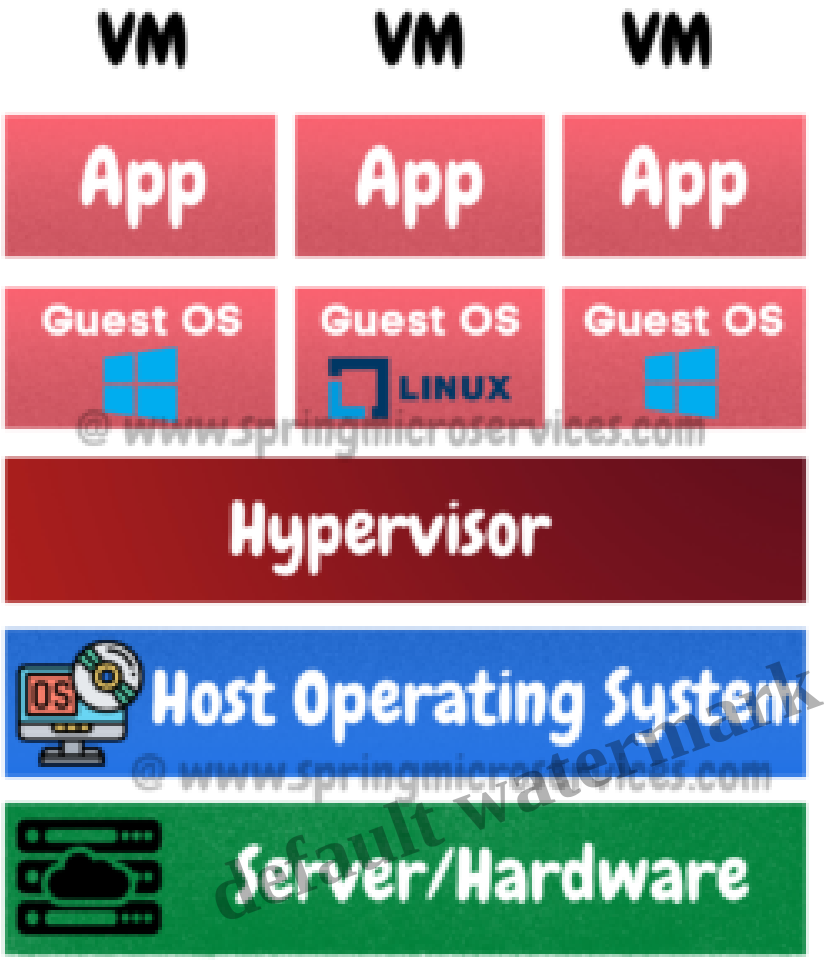
Containers are nothing but a lightweight, stand alone executable package of a piece of software that includes everything need to run it: **Code, runtime, system tools, system libraries, settings**. To reiterate container is a package which has the complete set of software information which is packaged into an image.

Advantages of a container is that it is not specific to any operating system. It can be used across OS. It can be ported on to a Linux machine or a windows machine or even in cloud machine.

Another advantage of a container is that we can isolate the software from its surrounding’s. To be precise the same image can be executed in the development, staging and the other environments and there is less coupling with the hardware. When we create an image, the runtime, the system libraries, the environment settings/configurations, the code are all packaged into one image and it will be deployed and tested across all the environments and this will ensure that lesser coupling is there with the hardware.

When something is adopted by the industry, then obviously there must be some existing problem which the new trend is trying to address. Before moving to the Container architecture lets talk about how the things were before i.e. in the traditional virtual machine architecture.

Traditional Virtual Machine Architecture

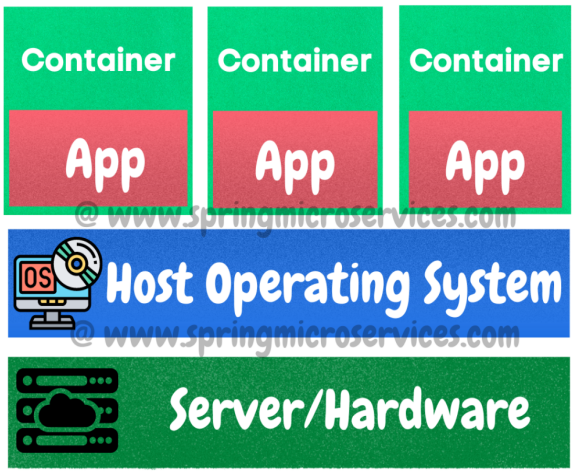


If you look at the above diagram, we have an operating system running on the Server/Hardware. On top of that we have something called as Hypervisor. Hypervisor will allow us to create virtual machines and each virtual machine will have its own operating system i.e. Guest OS, and then your app on the VM.

Drawback of virtualization.

- Lot of unnecessary space gets wasted by creating multiple OS in Hypervisor. Already there is an OS running on the server and then on top of that we have multiple OS running per instance.
- Licensing of each OS per VM is costlier.
- Additional configuration of VMs/Hypervisors.

Container Architecture



As opposed to the VM architecture which had a different OS per instance, here in case of Container technology it will share the same machine OS. The concept of guest OS is not applicable here like virtualization. Hypervisor layer is also eliminated in the container architecture.

Container is an environment that is isolated from the rest of your operating system. Have a look at the below diagram closely, We have a blue laptop which represents a Server and it. has the Host operating system in it, and then we have an orange laptop inside which represents a container. In this container I can install anything like MongoDB, Postgres, JDK and it does not affect the laptop which has the Host operating system.

Now that you are aware of the containerization concept, you can go over what is docker and how does it fit in the container architecture.

Category

1. Cloud
2. Docker

Date Created
December 15, 2021
Author
kk-ravi144gmail-com