Docker and Challenges While Building and Deploying Microservices

# Challenges: Deploying, Scaling and Portability

## Portability

With possible tens of microservices that need to be moved from environment to environment, from dev to test to QA to Production Replica to Production, we need to make it less costly in terms of configurations, etc.

## Scalability

In monolithic application you have one code base and you can make replicas out of the main monolithic server and scale out the application.

In microservices though we need to scale up and down based on the traffic on each microservice with minimum cost and effort.

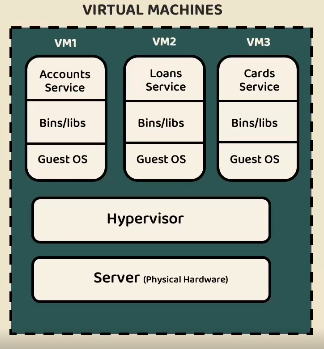
## The Solution:

In the monolithic apps we make a jar or war file out the project and deploy it on the server. In microservice ecosystem to solve the challenges above we **Containerize** our microservices.

* Containers are self-contained, isolated environment that carry all the necessary dependencies to run the application.
* By containerizing our projects, they become portable and we can run them on any cloud environments with a unified management, regardless of the technology and languages that are used inside these containers.
* **Docker** is an open-source platform that allows us to **package** and run our applications in a loosely isolated environment called a **container**.

# Containers vs Virtual Machines

* Before cloud computing you would have had to buy a hardware, install an OS, make the server public through an IP address and deploy your application on it.
* With cloud computing though you don’t have a physical data center. Instead, the cloud provider give you access to a virtual machine that runs on their hardware. They do this by using a hypervisor: a software that you can use to run multiple virtual machines on a single physical machine. The hypervisor allocates the underlying physical computing resources such as CPU and memory to individual virtual machines as required.



* **Without Containerization**, you would have to either buy one of these virtual machines for each of your microservices to serve their different requirements. And just imagine you had like 100 microservices.

Or you could run all of them inside one VM and let’s say you had to restart your VM, all the services will be down if you run them on a single VM. The thing about different dependencies of each microservice, I’m yet to know more about their requirements.

In terms of scalability, if all of a sudden the traffic on one microservice goes up, by the time you set up a new VM for that microservice your traffic might reduce again and you would have to scale down again.

* With Containerization you can use the kernel of the host operating system in the underlying infrastructure that can be a phisycal or virtual machine