**Summary**

**VMs**

In the past years, VMs (Virtual Machines) were the main mechanism to host an application. VMs encapsulate the code, configuration files, and dependencies necessary to execute the application.

In essence, a VM is composed of an **operating system** (OS) with a set of pre-installed libraries and packages. During execution, an **application** utilizes an OS filesystem, resources, and packages.

A set of VMs is managed through a **hypervisor**. A hypervisor provides the virtualization of the **infrastructure** which is composed of physical servers. As a result, a hypervisor is capable of creating, configuring, and managing multiple VMs on the available servers. For example, we are able to running applications A, B, and C on 3 separate VMs.

The utilization of VMs introduced standardization in infrastructure provisioning, in association with efficient use of available infrastructure. Instead of running an application per server, a hypervisor enables multiple VMs to run at the same time to host multiple applications. However, there is one downside to this mechanism: it is not efficient enough. For example, applications A, B, and C uses the same Operating System. Replicating an OS consumes a lot of resources, and the more applications we run the more space we allocate to the replication of the operating systems alone.

There was a clear need to optimize the usage of the available infrastructure. As a result, the virtualization of the Operating System was introduced. This prompted the appearance of **containers**, which represent the bare minimum an application requires for a successful execution e.g. code, config files, and dependencies. By default, there is a better usage of available infrastructure.

Multiple VMs on a hypervisor are replaced by multiple containers running on a single host operating system. The processes in the containers are completely isolated but are able to access the OS filesystem, resources, and packages. The creation and execution of containers is delegated to a container management tool, such as Docker.

The appearance of containers is unlocked by OS-level virtualization and as a result, multiple applications can run on the same OS. By nature, containers are lightweight, as these encapsulate only the application code and essential dependencies. Consequently, there is a better usage of available infrastructure and a more efficient pathway to release a product to consumers.