Azure Containers

examlabpractice.com

What is a container?

Just as shipping industries use physical containers to isolate different cargos—for example, to transport in ships and trains—software development technologies increasingly use an approach called containerization.

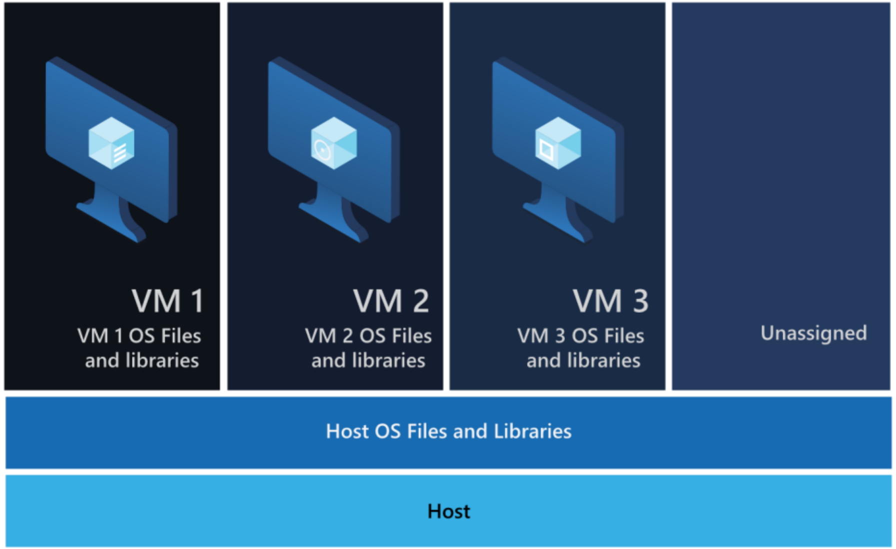
A standard package of software—known as a container—bundles an application’s code together with the related configuration files and libraries, and with the dependencies required for the app to run. This allows developers and IT pros to deploy applications seamlessly across environments.

Why are containers so helpful?

The problem of an application failing to run correctly when moved from one environment to another is as old as software development itself. Such problems typically arise due to differences in configuration underlying library requirements and other dependencies.

- Containers address this problem by providing a lightweight, immutable infrastructure for application packaging and deployment.
- An application or service, its dependencies, and its configuration are packaged together as a container image.
- The containerized application can be tested as a unit and deployed as a container image instance to the host operating system.

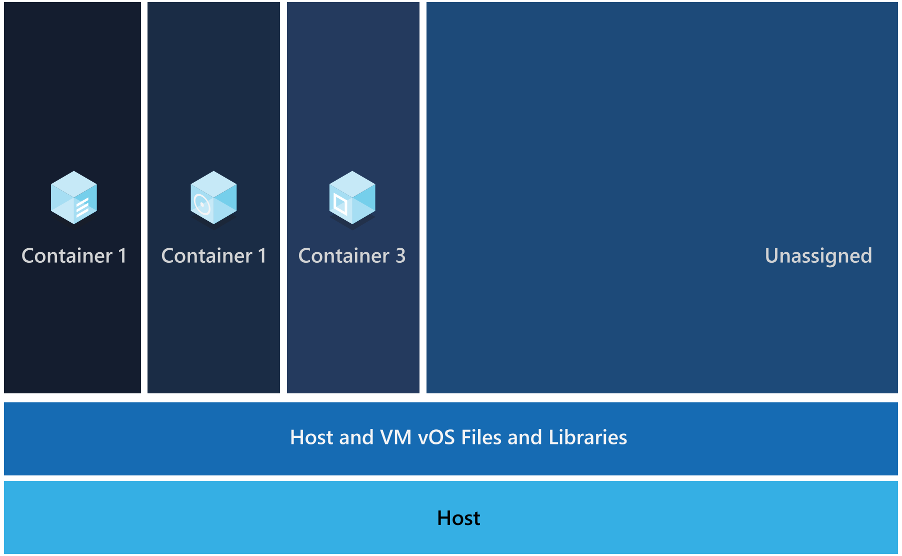
Containers vs. Virtual Machines Part 1


VM 1
VM 1 OS Files and libraries
VM 2
VM 2 OS Files and libraries
VM 3
VM 3 OS Files and libraries
Unassigned
Host OS Files and Libraries
Host

At a high level, VMs virtualize the underlying hardware so that multiple operating system (OS) instances can run on the hardware. Each VM runs an OS and has access to virtualized resources representing the underlying hardware.

VMs have many benefits. These include the ability to run different operating systems on the same server, more efficient and cost-effective utilization of physical resources, and faster server provisioning. On the flip side, each VM contains an OS image, libraries, applications, and more, and therefore can become quite large.

Containers vs. Virtual Machines Part 2



Container 1

Container 1

Container 3

Unassigned

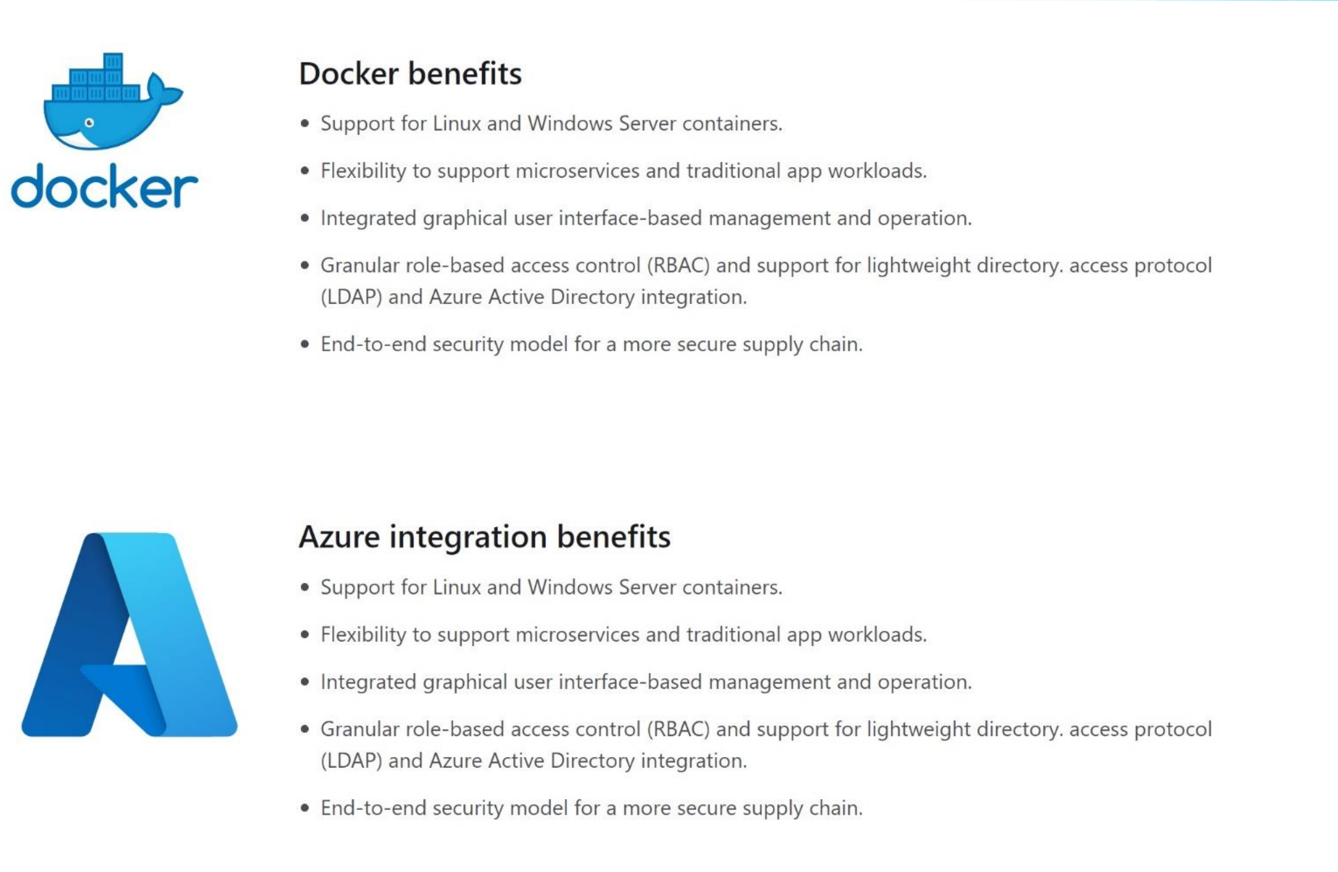
Host and VM vOS Files and Libraries

Host

A container virtualizes the underlying OS and causes the containerized app to perceive that it has the OS—including CPU, memory, file storage, and network connections—all to itself. Because the differences in underlying OS and infrastructure are abstracted, as long as the base image is consistent, the container can be deployed and run anywhere.

Since containers share the host OS, they don’t need to boot an OS or load libraries. This enables containers to be much more efficient and lightweight. Containerized applications can start in seconds, and many more instances of the application can fit onto the machine as compared to a VM scenario. The shared OS approach has the added benefit of reduced overhead when it comes to maintenance, such as patching and updates.

Docker support in Azure

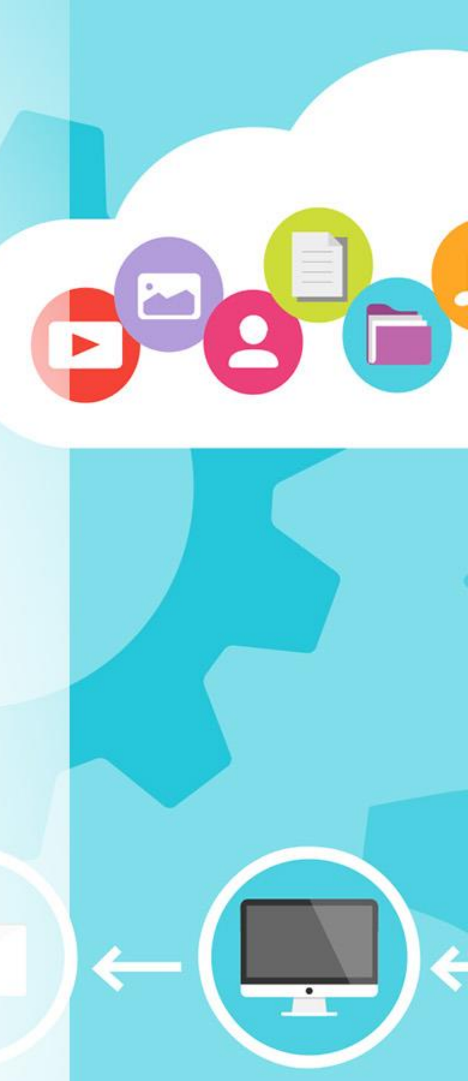


Docker benefits

- Support for Linux and Windows Server containers.
- Flexibility to support microservices and traditional app workloads.
- Integrated graphical user interface-based management and operation.
- Granular role-based access control (RBAC) and support for lightweight directory. access protocol (LDAP) and Azure Active Directory integration.
- End-to-end security model for a more secure supply chain.

Azure integration benefits

- Support for Linux and Windows Server containers.
- Flexibility to support microservices and traditional app workloads.
- Integrated graphical user interface-based management and operation.
- Granular role-based access control (RBAC) and support for lightweight directory. access protocol (LDAP) and Azure Active Directory integration.
- End-to-end security model for a more secure supply chain.



Azure Container Instances (ACI)

Containers are becoming the preferred way to package, deploy, and manage cloud applications. Azure Container Instances offers the fastest and simplest way to run a container in Azure, without having to manage any virtual machines and without having to adopt a higher-level service.

Azure Container Instances is a great solution for any scenario that can operate in isolated containers, including simple applications, task automation, and build jobs. For scenarios where you need full container orchestration, including service discovery across multiple containers, automatic scaling, and coordinated application upgrades, we recommend Azure Kubernetes Service (AKS).

Azure Kubernetes Service (AKS)

Azure Kubernetes Service (AKS) simplifies deploying a managed Kubernetes cluster in Azure by offloading the operational overhead to Azure. As a hosted Kubernetes service, Azure handles critical tasks, like health monitoring and maintenance. Since Kubernetes masters are managed by Azure, you only manage and maintain the agent nodes. Thus, AKS is free; you only pay for the agent nodes within your clusters, not for the masters.

When you deploy an AKS cluster, the Kubernetes master and all nodes are deployed and configured for you. Advanced networking, Azure Active Directory (Azure AD) integration, monitoring, and other features can be configured during the deployment process.