What is a container?

Standardized, portable packaging for your applications.

Containers explained

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.

Get started with Docker

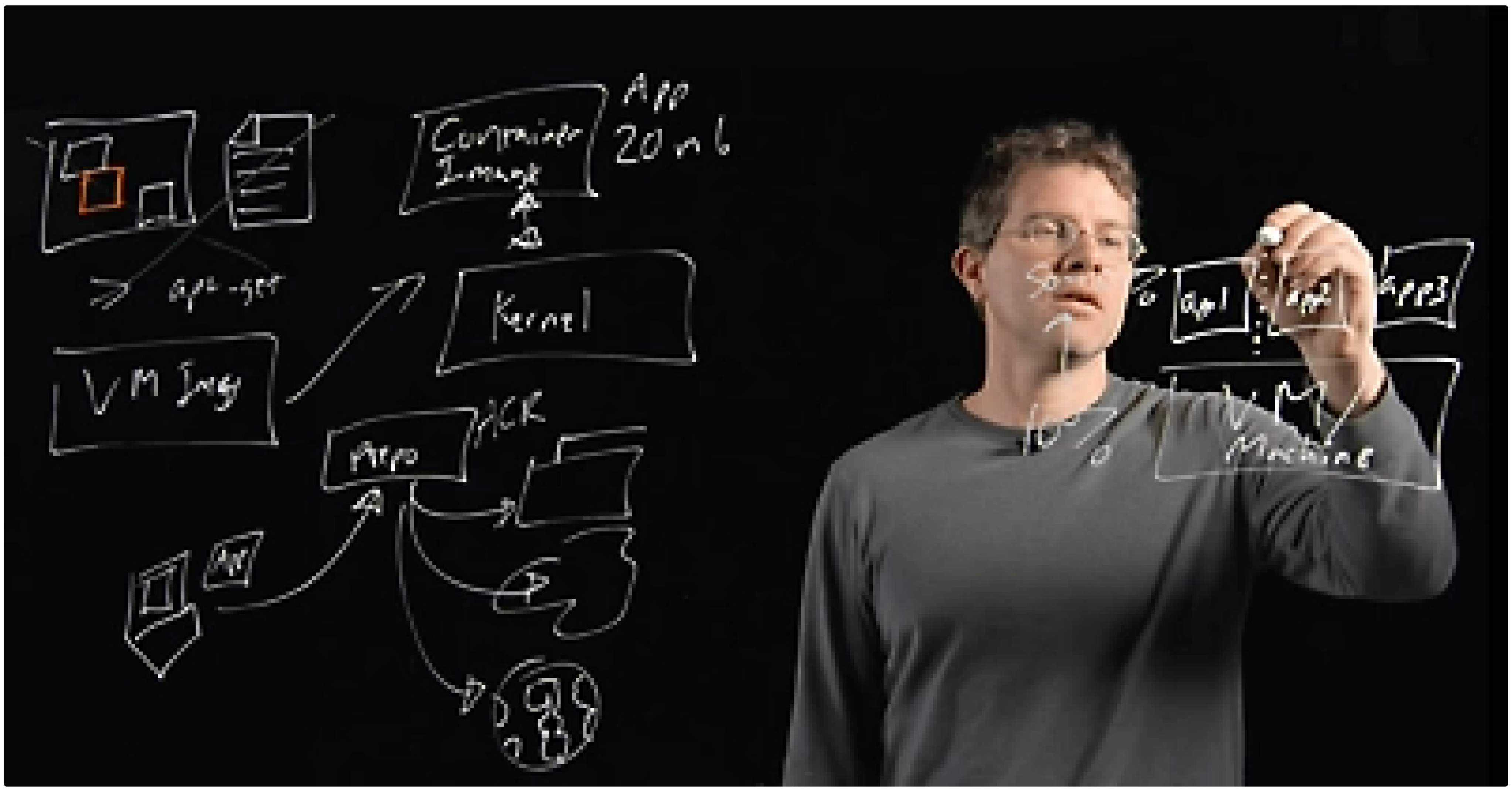
Why you should care about containers

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.

This way, containers enable developers and IT professionals to deploy applications across environments with little or no modification.

Learn more about the genesis and beauty of containers.



Container vs. virtual machine

When people think about virtualization, virtual machines (VMs) often come to mind. In fact, virtualization can take many forms, and containers are one of those. So what's the difference between VMs and containers?

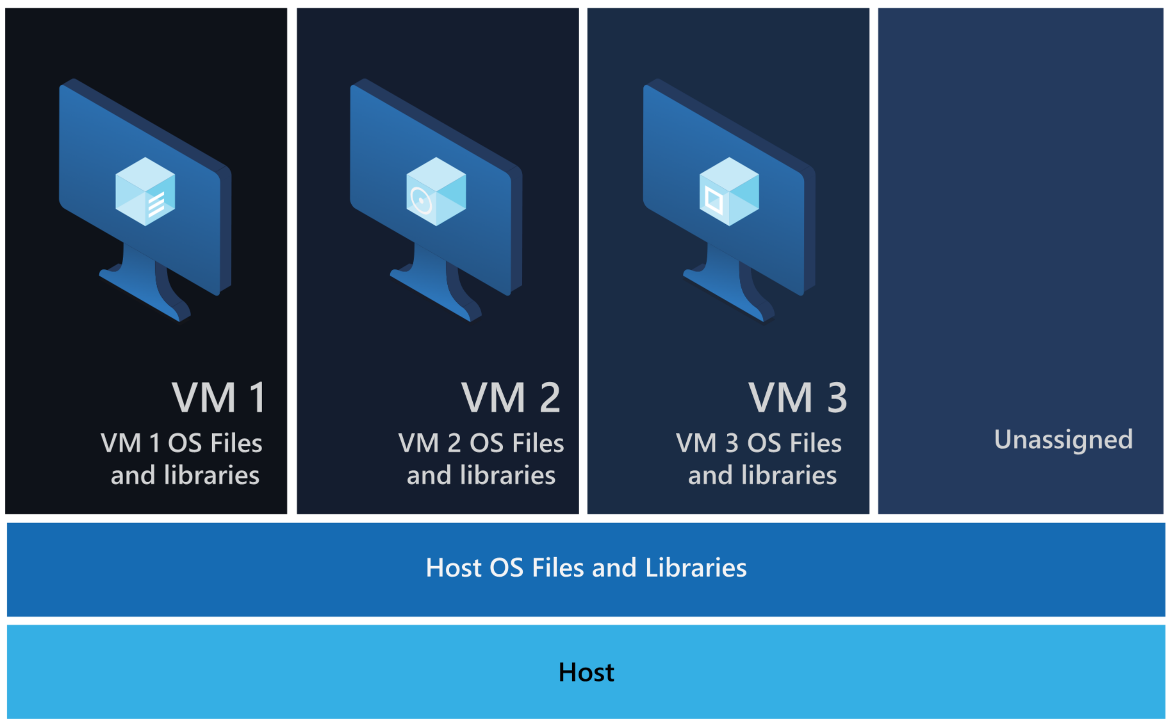
At a high level, VMs virtualize the underlying hardware so that multiple operating system (OS) instances can run on the hardware. Each VM runs an OS and has access to virtualized resources representing the underlying hardware.

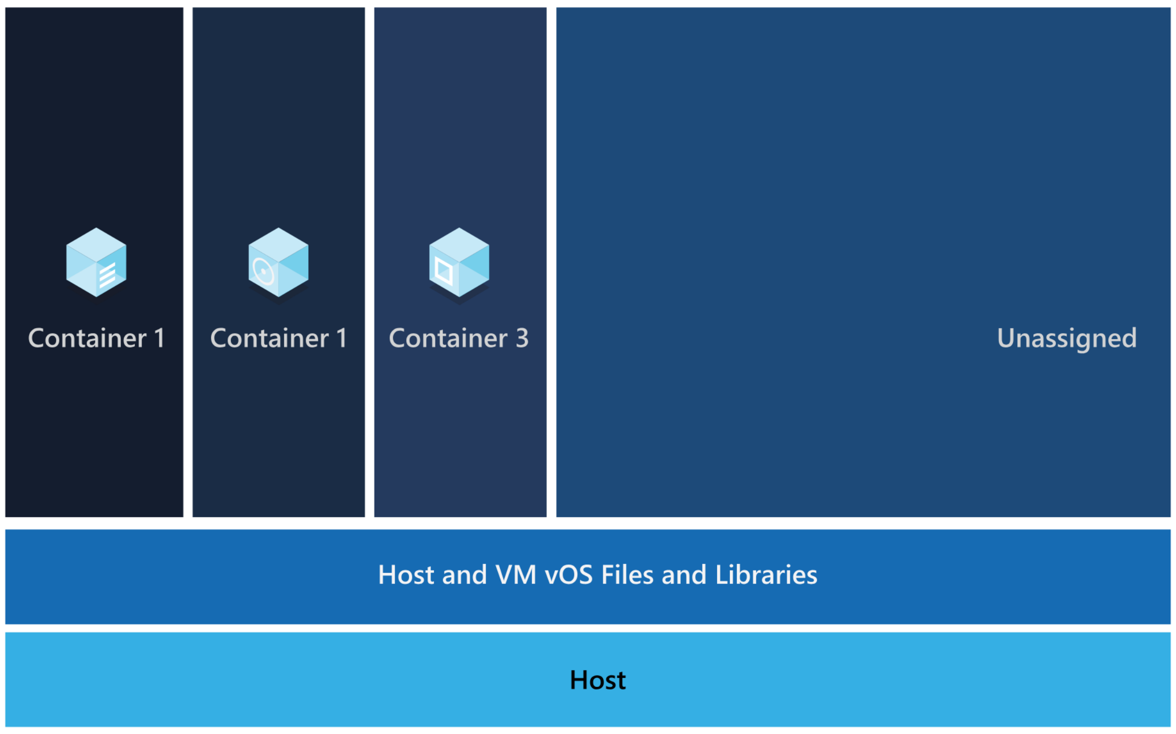
VMs have many benefits. These include the ability to run different operating systems on the same server, more efficient and cost-effective utilization of physical resources, and faster server provisioning. On the flip side, each VM contains an OS image, libraries, applications, and more, and therefore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

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. For developers, this is incredibly attractive.

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.

Though containers are portable, they’re constrained to the operating system they’re defined for. For example, a container for Linux can’t run on Windows, and vice versa.


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



Why containers

Agility

When developers build and package their applications into containers and provide them to IT to run on a standardized platform, this reduces the overall effort to deploy applications and can streamline the whole dev and test cycle. This also increases collaboration and efficiency between dev and operations teams to ship apps faster.

Portability

Containers provide a standardized format for packaging and holding all the components necessary to run the desired application. This solves the typical problem of “It works on my machine” and allows for portability between OS platforms and between clouds. Any time a container is deployed anywhere, it executes in a consistent environment that remains unchanged from one deployment to another. You now have a consistent format, from dev box to production.

Rapid scalability

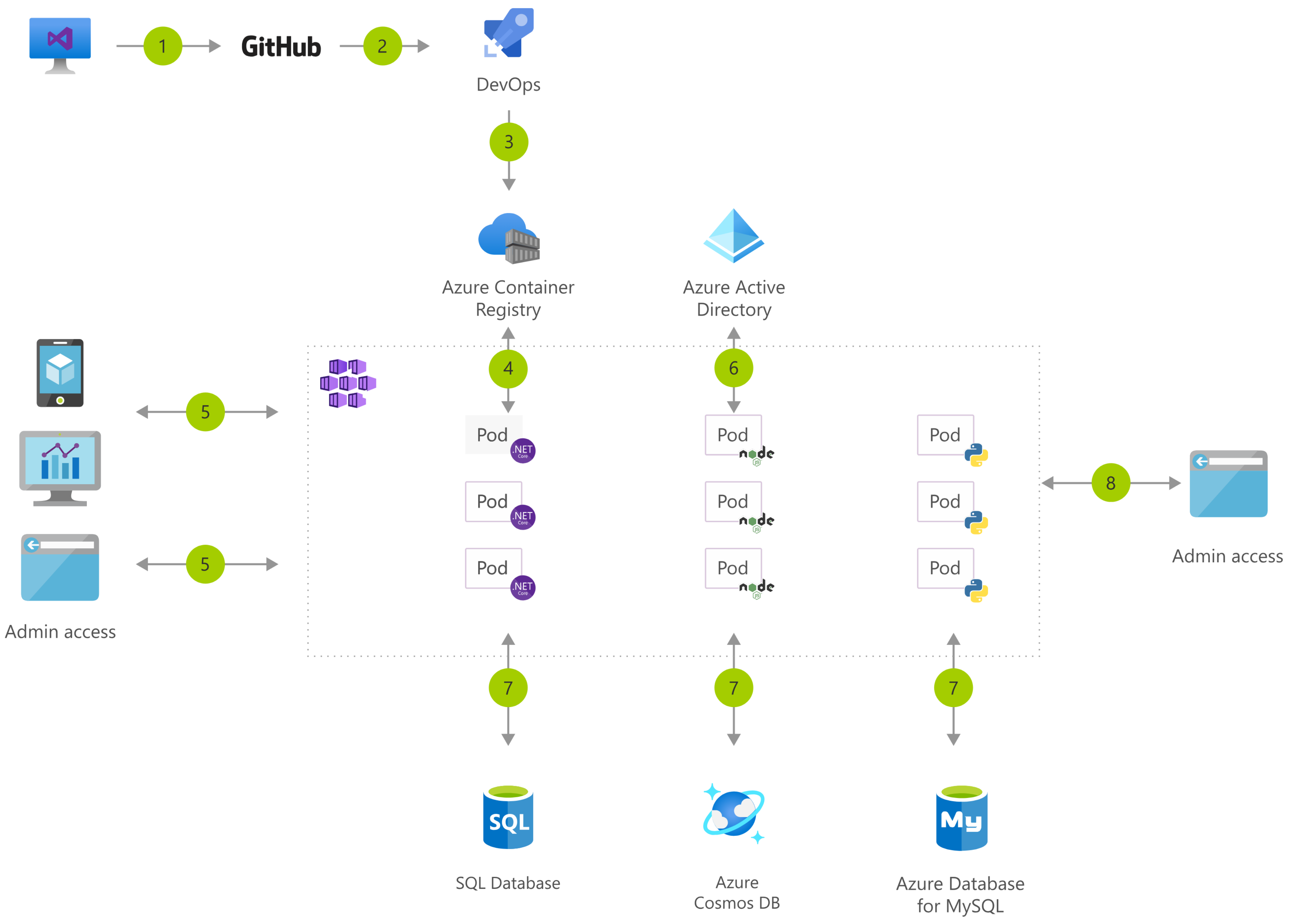
Since containers do not have the overhead typical of VMs, including separate OS instances, many more containers can be supported on the same infrastructure. The lightweight nature of containers means they can be started and stopped quickly, unlocking rapid scale-up and scale-down scenarios.

Use cases

Cloud-native applications | Lift and shift | Batch | Machine learning

Cloud-native applications

Cloud-native applications rely on containers for a common operational model across environments, including public, private, and hybrid. The low overhead and high density of containers allow many of them to be hosted inside the same virtual machine and makes them ideal for delivering cloud-native applications.



Beyond containers

To maximize the benefits of containers, consider complete solutions including container-optimized tools and services that help you achieve agility, security, and scale.

Orchestration

Running containers at scale requires orchestration and management of distributed, containerized applications via an orchestration platform such as Kubernetes.

[Explore Kubernetes with Azure >](#)

Security

With containers requires a layered approach, from container image to cluster isolation. Configuration of these guardrails is best set with your CI/CD pipelines.

[Check out security essentials for containers and Kubernetes >](#)

Serverless containers

You can further increase agility with containers on demand. Use serverless container technologies to easily run containers without managing servers and burst from your Kubernetes clusters when traffic comes in spikes.

[Try out serverless containers](#) >

DevOps

Containers allows developers to easily share software and dependencies across IT and production environments. When combined with DevOps practices, you can effectively ship code faster and shorten software development cycles.

Resources

Learn more about containers and related topics

What is cloud-native? > What is Kubernetes? > Introduction to Docker and containers >

Run containers and Kubernetes with Azure > Get started with containers >

Experience containers first-hand

Learn how to administer containers in Azure > Run Docker containers with Azure Container Instances >

Prepare your containers for Kubernetes >

Ready when you are—try containers and Kubernetes free with Azure **Start free**

Get the Azure mobile app

Explore Azure

Explore Azure

What is Azure?

Get started

Global infrastructure

Datacenter regions

Trust your cloud

Customer enablement

Customer stories

Products and pricing

Products and pricing

Products

Pricing

Free Azure services

Flexible purchase options

Cloud economics

Optimize your costs

Solutions and support

Solutions and support

Solutions

Resources for accelerating growth

Solution architectures

Support

Azure demo and live Q&A

Partners

Partners

Azure Marketplace

Find a partner

Join ISV Success

Resources

Resources

Training and certifications

Documentation

Blog

Developer resources

Students

Events and webinars

Analyst reports, white papers, and e-books

Videos

Cloud computing

Cloud computing

What is cloud computing?

What is cloud migration?

What is a hybrid cloud?

What is AI?

What is IaaS?

What is SaaS?

What is PaaS?

What is DevOps?

Change language

English (US)

Your Privacy Choices