

# Configure Azure Container Instances

---

## 1. Introduction

<https://learn.microsoft.com/en-us/training/modules/configure-azure-container-instances/1-introduction>

## Introduction

---

Completed

Containers and virtual machines are both forms of virtualization, but there are some key differences between them.

To provide context, let's consider a scenario: You're an Azure Administrator responsible for deploying and managing applications in a cloud environment. Your organization is looking for a solution that offers fast startup times, easy management, and the ability to run applications in isolated containers. You want to understand the benefits of using Azure Container Instances and how it compares to virtual machines.

In this module, you learn when to use Azure Container Instances instead of virtual machines. You also get an overview of features and use cases.

The goal of this module is to introduce you to AI-ready Azure Container Instances.

## Learning objectives

In this module, you learn how to:

- Identify when to use containers versus virtual machines.
- Identify the features and usage cases of Azure Container Instances.
- Implement Azure container groups.

## Skills measured

The content in the module helps you prepare for [Exam AZ-104: Microsoft Azure Administrator](#).

## Prerequisites

- Working knowledge of containerization concepts and terminology.
- Familiarity with cloud computing and experience with the Azure portal.

## 2. Compare containers to virtual machines

<https://learn.microsoft.com/en-us/training/modules/configure-azure-container-instances/2-compare-containers-to-virtual-machines>

# Compare containers to virtual machines

Completed

Hardware virtualization makes it possible to run multiple isolated instances of operating systems concurrently on the same physical hardware. Containers represent the next stage in the virtualization of computing resources.

Container-based virtualization allows you to virtualize the operating system. This approach lets you run multiple applications within the same instance of an operating system, while maintaining isolation between the applications. The containers within a virtual machine provide functionality similar to that of virtual machines within a physical server.

## Things to know about containers versus virtual machines

To better understand container-based virtualization, let's [compare containers and virtual machines](#).

| Compare          | Containers                                                                                                                                                                   | Virtual machines                                                                                                                                                                                                                                             |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Isolation</b> | A container typically provides lightweight isolation from the host and other containers, but a container doesn't provide as strong a security boundary as a virtual machine. | A virtual machine provides complete isolation from the host operating system and other virtual machines. This separation is useful when a strong security boundary is critical, such as hosting apps from competing companies on the same server or cluster. |

| Compare                   | Containers                                                                                                                                                                            | Virtual machines                                                                                                                                                                                    |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Operating system</b>   | Containers run the user mode portion of an operating system and can be tailored to contain just the needed services for your app. This approach helps you use fewer system resources. | Virtual machines run a complete operating system including the kernel, which requires more system resources (CPU, memory, and storage).                                                             |
| <b>Deployment</b>         | You can deploy individual containers by using Docker via the command line. You can deploy multiple containers by using an orchestrator such as Azure Kubernetes Service.              | You can deploy individual virtual machines by using Windows Admin Center or Hyper-V Manager. You can deploy multiple virtual machines by using PowerShell or System Center Virtual Machine Manager. |
| <b>Persistent storage</b> | Containers use Azure Disks for local storage for a single node, or Azure Files (SMB shares) for storage shared by multiple nodes or servers.                                          | Virtual machines use a virtual hard disk (VHD) for local storage for a single machine, or an SMB file share for storage shared by multiple servers.                                                 |
| <b>Fault tolerance</b>    | If a cluster node fails, the orchestrator on another cluster node rapidly recreates any containers running on the node.                                                               | Virtual machines can fail over to another server in a cluster, where the virtual machine's operating system restarts on the new server.                                                             |

## Things to consider when using containers

Containers offer several advantages over physical and virtual machines. Review the following benefits and consider how you can implement containers for the internal apps for your company.

- **Consider flexibility and speed.** Gain increased flexibility and speed when developing and sharing your containerized application code.
- **Consider testing.** Choose containers for your configuration to allow for simplified testing of your apps.
- **Consider app deployment.** Implement containers to gain streamlined and accelerated deployment of your apps.
- **Consider workload density.** Support higher workload density and improve your resource utilization by working with containers.

## 3. Review Azure Container Instances

# Review Azure Container Instances

---

Completed

Containers are becoming the preferred way to package, deploy, and manage cloud applications. There are many options for teams to build and deploy cloud native and containerized applications on Azure. In this unit, we review Azure Container Instances (ACI).

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.

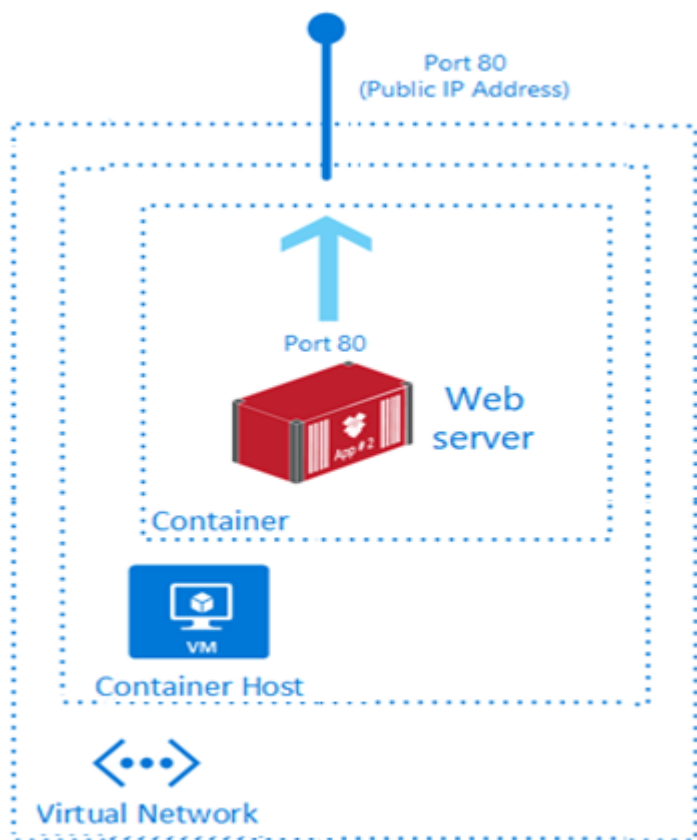
## Understand container images

All containers are created from container images. A container image is a lightweight, standalone, executable package of software that encapsulates everything needed to run an application. It includes the following components:

- **Code:** The application's source code.
- **Runtime:** The environment required to execute the application.
- **System tools:** Utilities necessary for the application to function.
- **System libraries:** Shared libraries used by the application.
- **Settings:** Configuration parameters specific to the application.

When you create a container image, it becomes a portable unit that can run consistently across different computing environments. These images are the building blocks for containers, which are instances of these images running at runtime.

The following illustration shows a web server container built with Azure Container Instances. The container is running on a virtual machine in a virtual network.



## Things to know about Azure Container Instances

Let's review some of the [benefits of using Azure Container Instances](#). As you review these points, think about how you can implement Container Instances for your internal applications.

- **Fast startup times.** Containers can start in seconds without the need to deploy and manage virtual machines.
- **Public IP connectivity and DNS names.** Containers can be directly exposed to the internet with an IP address and FQDN (fully qualified domain name).
- **Custom sizes.** You specify CPU cores (from 0.1 to 4 vCPU) and memory (from 0.1 to 16 GB) for each container at deployment time. Resource allocation is fixed for the lifetime of the container group.
- **Persistent storage.** Containers support direct mounting of Azure Files file shares.
- **Linux and Windows containers.** Container Instances can schedule both Windows and Linux containers. Specify the operating system type when you create your container groups.
- **Coscheduled groups.** Container Instances supports scheduling of multi-container groups that share host machine resources.
- **Virtual network deployment.** Linux container groups can be deployed into an Azure virtual network for private communication with other Azure resources. Virtual network deployed

containers receive no public IP address and communicate only within the virtual network or peered networks.

## 4. Implement container groups

<https://learn.microsoft.com/en-us/training/modules/configure-azure-container-instances/4-implement-container-groups>

# Implement container groups

---

Completed

The top-level resource in Azure Container Instances is the **container group**. A **container group** is a collection of containers that get scheduled on the same host machine. The containers share a lifecycle, resources, local network, and storage volumes.



## Things to know about container groups

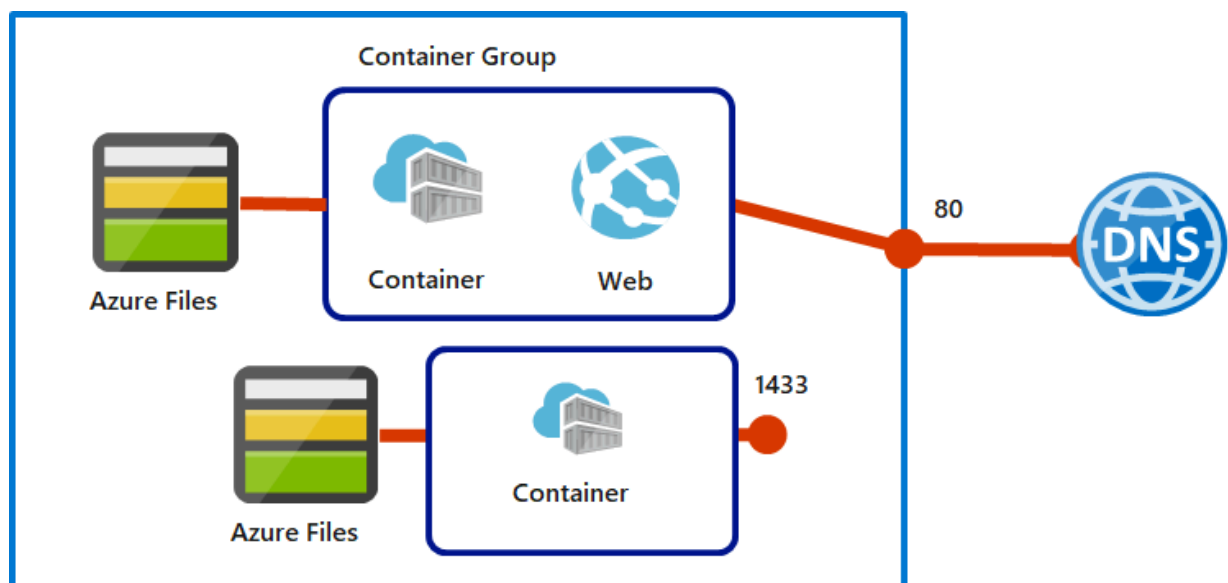
Let's review some of details about container groups for Azure Container Instances.

- A container group is similar to a pod in Kubernetes. A pod typically has a 1:1 mapping with a container, but a pod can contain multiple containers. The containers in a multi-container pod can share related resources.
- Azure Container Instances allocates resources to a multi-container group by adding together the resource requests of all containers in the group. Resources can include items such as CPUs, memory, and GPUs.
- There are three common ways to deploy a multi-container group.
  - **Azure Resource Manager template.** JSON-based infrastructure as code, ideal when deploying alongside other Azure resources.
  - **Bicep.** Microsoft's recommended infrastructure as code language, more concise than Azure Resource Manager templates. Bicep includes full IntelliSense support.

- **YAML files.** Container-focused format, ideal for deployments that include only container instances.
- Container groups can share an external-facing IP address, one or more ports on the IP address, and a DNS label with an FQDN.
  - **External client access.** You must expose the port on the IP address and from the container to enable external clients to reach a container in your group.
  - **Port mapping.** Port mapping isn't supported because containers in a group share a port namespace.
  - **Deleted groups.** When a container group is deleted, its IP address and FQDN are released.

### Configuration example

Consider the following example of a multi-container group with two containers.



The multi-container group has the following characteristics and configuration:

- The container group is scheduled on a single host machine, and is assigned a DNS name label.
- The container group exposes a single public IP address with one exposed port.
- One container in the group listens on port 80. The other container listens on port 1433.
- The group includes two Azure Files file shares as volume mounts. Each container in the group mounts one of the file shares locally.

### Things to consider when using container groups

Multi-container groups are useful when you want to divide a single functional task into a few container images. Different teams can deliver the images, and the images can have separate resource requirements.

Consider the following scenarios for working with multi-container groups. Think about what options can support your internal apps for the online retailer.

- **Consider web app updates.** Support updates to your web apps by implementing a multi-container group. One container in the group serves the web app and another container pulls the latest content from source control.
- **Consider log data collection.** Use a multi-container group to capture logging and metrics data about your app. Your application container outputs logs and metrics. A logging container collects the output data and writes the data to long-term storage.
- **Consider app monitoring.** Enable monitoring for your app with a multi-container group. A monitoring container periodically makes a request to your application container to ensure your app is running and responding correctly. The monitoring container raises an alert if it identifies possible issues with your app.
- **Consider front-end and back-end support.** Create a multi-container group to hold your front-end container and back-end container. The front-end container can serve a web app. The back-end container can run a service to retrieve data.

## 5. Review Azure Container Apps

<https://learn.microsoft.com/en-us/training/modules/configure-azure-container-instances/5-review-docker-platform>

# Review Azure Container Apps

---

Completed

There are many options for teams to build and deploy cloud native and containerized applications on Azure. Let's understand which scenarios and use cases are best suited for Azure Container Apps and how it compares to other container options on Azure. Listen to a developer's view of Azure Container Instances.





## Things to know about Azure Container Apps

[Azure Container Apps](#) is a serverless platform that allows you to maintain less infrastructure and save costs while running containerized applications. Instead of worrying about server configuration, container orchestration, and deployment details, Container Apps provides all the up-to-date server resources required to keep your applications stable and secure.

Common uses of Azure Container Apps include:

- Deploying API endpoints
- Hosting background processing jobs
- Handling event-driven processing
- Running microservices

Applications built on Azure Container Apps can dynamically scale based on the following characteristics:

- HTTP traffic
- Event-driven processing
- CPU or memory load
- Any KEDA-supported scaler

## Things to consider when using Azure Container Apps

Azure Container Apps enables you to build serverless microservices and jobs based on containers. Distinctive features of Container Apps include:

- Optimized for running general purpose containers, especially for applications that span many microservices deployed in containers.
- Powered by Kubernetes and open-source technologies like Dapr, KEDA, and envoy.
- Supports Kubernetes-style apps and microservices with features like service discovery and traffic splitting.
- Enables event-driven application architectures by supporting scale based on traffic and pulling from event sources like queues, including scale to zero.
- Supports running on demand, scheduled, and event-driven jobs.

Azure Container Apps doesn't provide direct access to the underlying Kubernetes APIs. If you would like to build Kubernetes-style applications and don't require direct access to all the native Kubernetes APIs and cluster management, Container Apps provides a fully managed experience based on best-practices. For these reasons, many teams prefer to start building container microservices with Azure Container Apps.

## Compare container management solutions

Azure offers several container platforms for different scenarios. Azure Container Instances (ACI) is best for isolated, short-lived tasks. Azure Container Apps (ACA) serves serverless microservices. Azure Kubernetes Service (AKS) provides full Kubernetes control for complex orchestration needs.

| Feature     | Azure Container Apps (ACA)                                                                                                                                                  | Azure Kubernetes Service (AKS)                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Overview    | ACA is a serverless container platform that simplifies the deployment and management of microservices-based applications by abstracting away the underlying infrastructure. | AKS simplifies deploying a managed Kubernetes cluster in Azure by offloading the operational overhead to Azure. It's suitable for complex applications that require orchestration. |
| Deployment  | ACA provides a PaaS experience with quick deployment and management capabilities.                                                                                           | AKS offers more control and customization options for Kubernetes environments, making it suitable for complex applications and microservices.                                      |
| Management  | ACA builds upon AKS and offers a simplified PaaS experience for running containers.                                                                                         | AKS provides a more granular control over the Kubernetes environment, suitable for teams with Kubernetes expertise.                                                                |
| Scalability | ACA supports both HTTP-based autoscaling and event-driven scaling, making it ideal for applications that need to respond quickly to changes in demand.                      | AKS offers horizontal pod autoscaling and cluster autoscaling, providing robust scalability options for containerized applications.                                                |
| Use Cases   | ACA is designed for microservices and serverless applications that benefit from rapid scaling and simplified management.                                                    | AKS is best for complex, long-running applications. These applications require full Kubernetes features and tight integration with other Azure services.                           |

## 6. Exercise: Implement Container Instances

<https://learn.microsoft.com/en-us/training/modules/configure-azure-container-instances/6-simulation-containers>

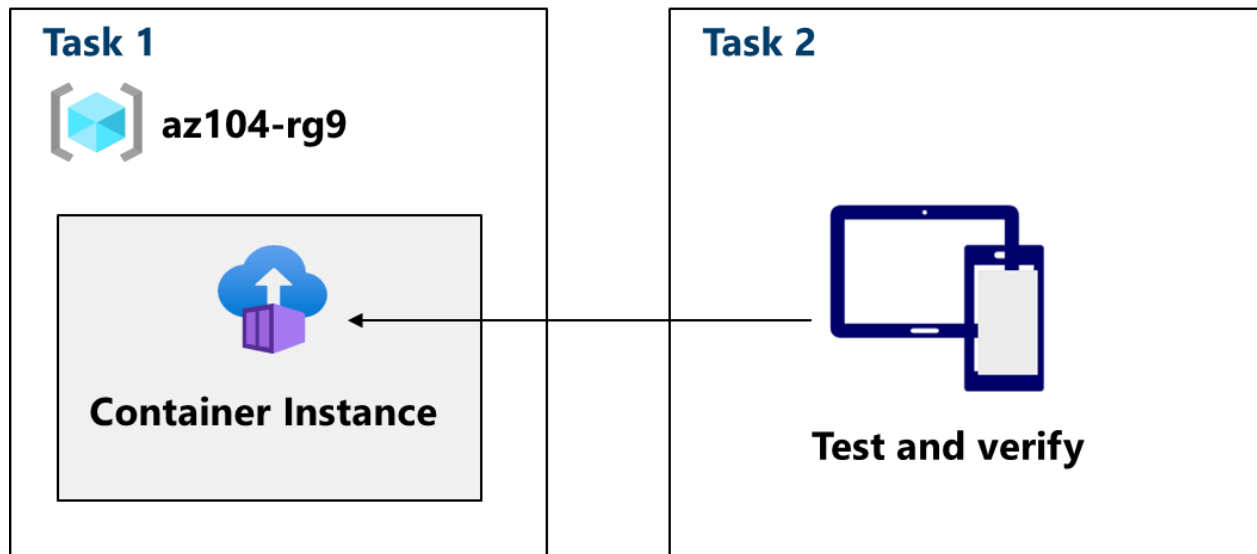
# Exercise: Implement Container Instances

Completed

## Lab scenario

Your organization has a web application that runs on a virtual machine in your on-premises data center. The organization wants to move all applications to the cloud but doesn't want to have a large number of servers to manage. You decide to evaluate Azure Container Instances and Docker.

## Architecture diagram



## Job skills

- Deploy an Azure Container Instance using a Docker image.
- Test and verify deployment of an Azure Container Instance.

### Note

Estimated time: 15 minutes. To complete this exercise, you need an [Azure subscription](#).

Launch the exercise, and follow the instructions. When finished, be sure to return to this page so you can continue learning.

[Launch Exercise](#)

## 7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/configure-azure-container-instances/7-knowledge-check>

# Module assessment

Completed

## 8. Summary and resources

## Summary and resources

---

### Completed

In this module, you learned how to identify when to use Azure Container Instances versus Azure virtual machines. You explored the features and usage cases of Azure Container Instances. You discovered how to implement Azure container groups.

The main takeaways from this module are:

- Containers provide lightweight isolation and use fewer system resources compared to virtual machines.
- Containers can be deployed individually using Docker or with an orchestrator like Azure Container Apps.
- Containers use Azure Disks or Azure Files for storage.
- A container group is a collection of containers that get scheduled on the same host machine.
- Containers can be rapidly recreated on another cluster node if a node fails.

### Learn more with Copilot

Copilot can assist you in configuring Azure infrastructure solutions. Copilot can compare, recommend, explain, and research products and services where you need more information. Open a Microsoft Edge browser and choose Copilot (top right) or navigate to [copilot.microsoft.com](https://copilot.microsoft.com). Take a few minutes to try these prompts and extend your learning with Copilot.

- Compare benefits and usage cases for containers and virtual machines.
- What are the best practices for configuring Azure Container Instances for task-based workloads? Explain the restart policies.
- How do I deploy a multi-container group in Azure Container Instances using Bicep? Show an example with environment variables.

### Learn more with documentation

- [Containers versus virtual machines](#). This article reviews the key similarities and differences between containers and virtual machines (VMs), and when you might want to use each.
- [Quickstart: Deploy a container instance in Azure using the Azure portal](#). In this quickstart, you use the Azure portal to deploy an isolated Docker container and make its application available with a fully qualified domain name (FQDN). After configuring a few settings and deploying the container, you can browse to the running application:

- [Container groups in Azure Container Instances](#). This article describes what container groups are and the types of scenarios they enable.

## Learn more with self-paced training

- [Run container images in Azure Container Instances](#). Learn how Azure Container Instances can help you quickly deploy containers, how to set environment variables, and specify container restart policies.
- [Implement Azure Container Apps](#). Learn how Azure Container Apps can help you deploy and manage microservices and containerized apps on a serverless platform that runs on top of Azure Kubernetes Service.
- [Introduction to Docker containers](#). Learn the benefits of using Docker containers as a containerization platform. Discuss the infrastructure provided by the Docker platform.