**Azure Container Instances**

Azure Container Instances offers the fastest and simplest way to run a container in Azure, without having to provision 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)**.

**Features**

1. **Fast startup times**

Containers offer significant startup benefits over virtual machines (VMs). Azure Container Instances can start containers in Azure in seconds, without the need to provision and manage VMs.

1. **Public IP connectivity and DNS name**

Azure Container Instances enables exposing your containers directly to the internet with an IP address and a fully qualified domain name (FQDN). When you create a container instance, you can specify a custom DNS name label so your application is reachable at *customlabel.azureregion.azurecontainer.io*.

1. **Hypervisor-level security**

Historically, containers have offered application dependency isolation and resource governance but have not been considered sufficiently hardened for hostile multi-tenant usage. Azure Container Instances guarantees your application is as isolated in a container as it would be in a VM.

1. **Custom sizes – Pay per second**

Containers are typically optimized to run just a single application, but the exact needs of those applications can differ greatly. Azure Container Instances provides optimum utilization by allowing exact specifications of CPU cores and memory. You pay based on what you need and get billed by the second, so you can fine-tune your spending based on actual need.

1. **Persistent storage**

To retrieve and persist state with Azure Container Instances, we offer direct mounting of Azure Files shares.

1. **Support for Linux and Windows containers**

Azure Container Instances can schedule both Windows and Linux containers with the same API. Simply specify the OS type when you create your container groups. Some features are currently restricted to Linux containers.

1. **Co-scheduled groups (Multi-container groups)**

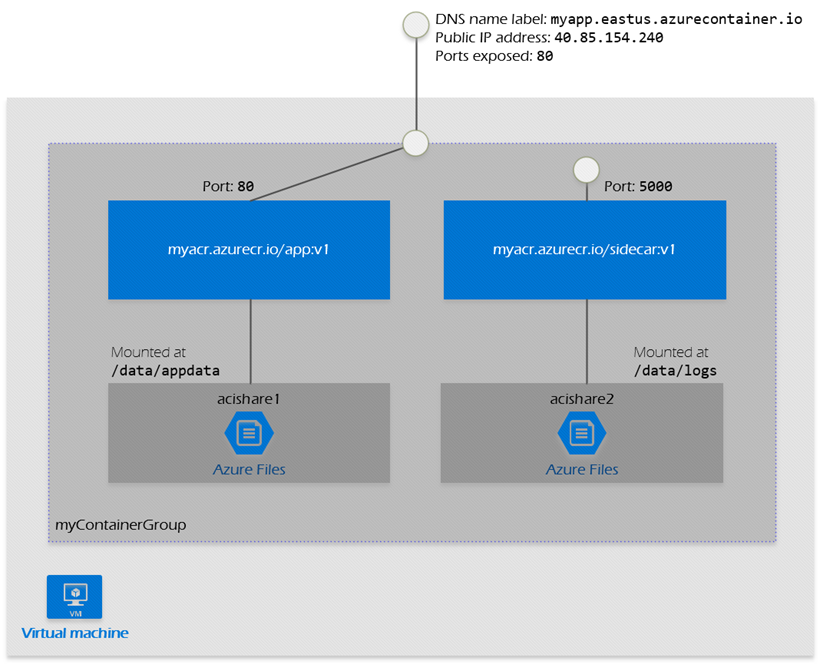
Azure Container Instances supports scheduling of multi-container groups that share a host machine, local network, storage, and lifecycle. This enables you to combine your main application container with other supporting role containers, such as logging sidecars.

**Container groups in Azure Container Instances**

The top-level resource in Azure Container Instances is the container group. This article describes what container groups are and the types of scenarios they enable.

**How a container group works**

A container group is a collection of containers that get scheduled on the same host machine. The containers in a container group share a lifecycle, local network, and storage volumes. It's similar in concept to a pod in Kubernetes and DC/OS. **Multi-container groups are currently restricted to Linux containers**.



**Container Group characteristics**

* **Deployment**

Container groups have a minimum resource allocation of 1 vCPU and 1 GB memory. Individual containers within a container group can be provisioned with less than 1 vCPU and 1 GB memory. Within a container group, the distribution of resources can be customized to multiple containers within the limits established at the container group-level. For example, two containers each with 0.5 vCPU residing in a container group that's allocated 1 vCPU.

* **Networking**

Container groups share an IP address and a port namespace on that IP address. To enable external clients to reach a container within the group, you must expose the port on the IP address and from the container. Because containers within the group share a port namespace, port mapping is not supported. Containers within a group can reach each other via localhost on the ports that they have exposed, even if those ports are not exposed externally on the group's IP address.

* **Storage**

You can specify external volumes to mount within a container group. You can map those volumes into specific paths within the individual containers in a group.