Azure Containers

Table of Contents

1. Introduction

Docker was initially developed for Linux and has expanded to support Windows. Individual Docker images are either Windows-based or Linux-based, but can’t be both at the same time.

For example, Microsoft offers Windows and Linux Docker images containing an ASP.NET Core environment that can be used as the basis for containerized ASP.NET Core applications.

Linux computers with Docker installed can only run Linux containers. Windows computers with Docker installed can run both kinds of containers. Windows accomplishes this by using a virtual machine to run a Linux system and uses the virtual Linux system to run Linux containers.

2. Glossary

Registry

A registry is a web service to which Docker can connect to upload and download container images. The most well-known registry is Docker Hub, which is a public registry.

Repository

A registry is organized as a series of repositories. Each repository contains multiple Docker images that share a common name and generally the same purpose and functionality. These images normally have different versions identified with a tag.

When you download and run an image, you must specify the registry, repository, and version tag for the image. Tags are text labels, and you can use your version numbering system (v1.0, v1.1, v1.2, v2.0, and so on).

Base image

The process of identifying a suitable base image usually starts with a search on Docker Hub for a ready-made image that **already contains an application framework** and all the utilities and tools of a Linux distribution like Ubuntu or Alpine.

Dockerfile

A Dockerfile is a plain text file containing all the commands needed to build an image. Dockerfiles are written in a minimal scripting language designed for **building and configuring images**, and documents the operations required to build an image starting with a base image. By convention, applications meant to be packaged as Docker images typically have a Dockerfile located in the **root of their source code**.

The `docker build` command creates a new image by running a Dockerfile. The `-f` flag indicates the name of the Dockerfile to use. The `-t` flag specifies the name of the image to be created. The final parameter, `.`, provides the build context for the source files for the *COPY* command.

The PORTS field indicates port 80 in the image was mapped to port 8080 on your computer.

Azure Container Registry (ACR)

Azure Container Registry is a registry hosting service provided by Azure. Each Azure Container Registry resource you create is a separate registry with a unique URL. These registries are private: they require authentication to push or pull images.

Azure Container Instance (ACI)

Azure Container Instances is a great solution for any scenario that can operate in isolated containers, including simple applications, task automation, and build jobs.

Azure Container Instances also supports executing a command in a running container by providing an interactive shell to help with application development and troubleshooting. Access takes places over HTTPS, using TLS to secure client connections.

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

Azure Kubernetes Service (AKS)

Azure Kubernetes Service simplifies deploying a managed Kubernetes cluster in Azure by offloading much of the complexity and operational overhead to Azure. As a hosted Kubernetes service, Azure handles critical tasks for you, like health monitoring and maintenance.

Resource Group

An Azure resource group is a logical container into which Azure resources are deployed and managed.

3. Create an ACR — build images

1. `az group create --name [group_name] --location [location]` create a resource group if necessary
2. `az acr create --name [registry_name] --resource-group [group_name] --sku standard --admin -enabled true` — create an ACR of standard scalability and storage

3. `az acr credential show --name [registry_name]` — get registry password and username to login to the specified registry
4. `docker login [registry_name].azurecr.io` — login to the private ACR (enter registry username and password)
5. `docker tag [local_image_name:tag] [registry_name].azurecr.io/[local_image_name:tag]` — create an alias for the image that specifies the repository and tag to be created in the Docker registry
6. `docker images` — check image with annotated tags
7. `docker push [registry_name].azurecr.io/[local_image_name:tag]` — upload the specified image to the registry in the ACR
8. `az acr repository list --name [registry_name]` — query the repositories in the registry to confirm the image upload (you can additionally specify the username (`--username [username]` and password `--password [password]`)
9. `az acr repository show --repository [repository_name] --name [registry_name]` — list the images and their properties as JSON in the specified registry
10. cleanup resources if necessary with `az group delete --name [group_name]`

Keep in mind that you'll have to create a **resource group before** you create the registry

You'll see at least two tags for each image in a repository. One tag will be value you specified in the acr build command. The other will be latest. Every time you rebuild an image, ACR automatically creates the latest tag as an alias for the most recent version of the image.

The name of the repo in the registry is equivalent to the name of the local image.

4. Manage ACR

- `az acr credential show --name [registry_name]` — get registry password and username to login to the specified registry
- `az acr credential show --name [registry_name] --query passwords[0].value` — get registry password to login to the specified registry
- `az acr show --name [registry_name] --query loginServer` get the URL of the login-server
- `docker login [registry_name].azurecr.io --username=[username] --password=[password]` — login to the login server with the specified username and password
- `az acr repository show-tags --name [registry_name] --repository [repository_name] --username [username] --password [password] --output text` — show tags of the specified repo as text
- `az group delete --name [group_name]` — remove the resource group, the container registry, and the container images stored

The **login server URL** for a registry in Azure Container Registry has the form [registry_name].azurecr.io.

5. Create an ACI — run containers

1. `az container create --resource-group [group_name] --name [instance_name] --image [registry_name].azurecr.io/[image_name:latest] --dns-name-label [dns_name] --registry-username [username] --registry-password [password]` — create an ACI, which loads the image from the ACR, and run it in Azure
2. `az container show --resource-group [group_name] --name [instance_name] --query ipAddress.fqdn` — query the IP address of the instance to find the fully qualified domain name of the instance

The instance will be **allocated a public IP address**. You access the instance with this IP address. You can **optionally specify a DNS name** if you prefer to reference the instance through a more user-friendly label.

The default port is 80 and the port protocol is TCP.

6. Manage containers

- `az container create --resource-group [group_name] --name [instance_name] --image [full_image_name] --dns-name-label [dns_name] --cpu [1] --memory [1] --ip-address public --image-registry-login-server [login_server] --image-registry-username [username] --image-registry-password [password]` — deploy a container inside the specified resource group with the specified name, domain name, cpu, memory, ip address
- `az container start --resource-group [group_name] --name [instance_name]` — start the specified container
- `az container restart --resource-group [group_name] --name [instance_name]` — restart the specified container
- `az container stop --resource-group [group_name] --name [instance_name]` — stop the specified container
- `az container delete --resource-group [group_name] --name [instance_name]` — delete the specified container
- `az container list` — list containers
- `az container list --resource-group [group_name]` — list all containers in a resource group
- `az container list --resource-group [group_name] --query value[].[name,provisioningState` — list specified information of all containers in a resource group
- `az container show --resource-group [group_name] --name [instance_name] --query "{FQDN:ipAddress.fqdn,ProvisioningState:provisioningState}" --out table` — check container status
- `az container show --resource-group [group_name] --name [instance_name]` — show details of the

specified container (JSON)

- `az container show --resource-group [group_name] --name [instance_name] --query value[].[name,provisioningState` — list specified information of the specified container in a resource group
- `az container logs --resource-group [group_name] --name [instance_name]` — show the logs of the specified container (JSON)`
- `az container attach --resource-group [group_name] --name [instance_name]` – attach the local standard out and standard error streams to that of the container

> The container details is shown as JSON, specifying e.g. the operating system, the allocated memory, the number of CPUs (vcpu),

7. Integrate Azure containers with Kubernetes

Deploy an ACI with Kubernetes as orchestrator

1. `az acs create --resource-group [group_name] --name [cluster_name] --dns-prefix [prefix] --generate-ssh-key --orchestrator-type kubernetes` — create a new container service
2. `az acs kubernetes install-cli` — install the Kubernetes CLI tool to manage the Kubernetes cluster **kubectl**
3. `az acs kubernetes get-credentials --resource-group [group_name] --name [cluster_name]` — get the credentials to configure kubectl to connect to your Kubernetes cluster
4. `kubectl get nodes` — verify the connection to your cluster (status must be ready)

---

1. `az aks create --resource-group [group_name] --name [cluster_name] --node-count 1 --enable-addons monitoring --generate-ssh-keys` — create an AKS cluster
2. `az aks install-cli`
3. `az aks get-credentials --resource-group [group_name] --name [cluster_name]`

8. Manage Azure Account

- `az login` — sign in to the Azure CLI
- `az logout` — Log out to remove access to Azure subscriptions
- `az account list -o table` — list Azure accounts table-formatted
- `az version` — find the version and dependent libraries that are installed
- `az upgrade` — upgrade to the latest version

9. Resources

- Quickstart: Create a private container registry using the Azure CLI
- Build a containerized web application with Docker
- Quickstart: Deploy an Azure Kubernetes Service cluster using the Azure CLI