Exercise - Create a private, highly available container registry

100 XP

15 minutes

The Fruit Smoothies software development and operations teams made the decision to containerize all newly developed applications. Containerized applications provide the teams with mutual benefits. For example,

- The ease of managing hosting environments
- The guarantee of continuity in software delivery
- The efficient use of server hardware
- The portability of applications between environments.

The teams made the decision to store all containers in a central and secure location and the decision made is to use Azure Container Registry.

In this exercise, you will:

- ✔ *Create a container registry* by using the Azure CLI
- ✔ *Build container images* by using Azure Container Registry Tasks
- ✔ *Verify container images* in Azure Container Registry
- ✔ *Configure an AKS cluster* to authenticate to an Azure Container Registry

Create a container registry

Azure Container Registry is a managed Docker registry service based on the open-source Docker Registry 2.0. Container Registry is private and hosted in Azure. You use it to build, store, and manage images for all types of container deployments.

Container images can be pushed and pulled with Container Registry by using the Docker CLI or the Azure CLI. You can use Azure portal integration to visually inspect the container images in the container registry. In distributed environments, the Container Registry geo-replication feature can be used to distribute container images to multiple Azure datacenters for localized distribution.

Azure Container Registry Tasks can also build container images in Azure. Tasks use a standard Dockerfile to create and store a container image in Azure Container Registry without the need for local Docker tooling. With Azure Container Registry Tasks, you can build on-demand or fully automate container image builds by using DevOps processes and tooling.

Let's deploy a container registry for the Fruit Smoothies environment.

1. The container registry name must be unique within Azure and contain between 5 and 50 *alphanumeric* characters. For learning purposes, run this command from Azure Cloud Shell to create a Bash variable that holds a unique name.

   bash
   ```
   ACR_NAME=acr$RANDOM
   ```

2. You use the `az acr create` command to create the registry in the same resource group and region as your Azure Kubernetes Service (AKS) cluster. For example, **akswkrshop** in **East US**.

   Run the command below to create the ACR instance.

   Azure CLI
   ```
   az acr create \
       --resource-group $RESOURCE_GROUP \
       --location $REGION_NAME \
       --name $ACR_NAME \
       --sku Standard
   ```

   You'll see a response similar to this JSON example when the command completes.

   JSON
   ```
   {
     "adminUserEnabled": false,
     "creationDate": "2019-12-28T01:33:23.906677+00:00",
     "id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/akswrkshop/providers/Microsoft.ContainerRegistry/registries/acr4229",
     "location": "eastus",
     "loginServer": "acr4229.azurecr.io",
     "name": "acr4229",
     "networkRuleSet": null,
     "policies": {
       "quarantinePolicy": {
         "status": "disabled"
       },
       "retentionPolicy": {
         "days": 7,
         "lastUpdatedTime": "2019-12-28T01:33:25.870458+00:00",
         "status": "disabled"
       },
       "trustPolicy": {
         "status": "disabled",
         "type": "Notary"
       }
     },
     "provisioningState": "Succeeded",
     "resourceGroup": "akswrkshop",
     "sku": {
       "name": "Standard",
       "tier": "Standard"
     },
     "status": null,
     "storageAccount": null,
     "tags": {},
     "type": "Microsoft.ContainerRegistry/registries"
   }
   ```

Build the container images by using Azure Container Registry Tasks

The Fruit Smoothies rating app makes use of two container images, one for the front-end website and one for the RESTful API web service. Your development teams use the local Docker tooling to build the container images for the website and API web service. A third container is used to deploy the document database provided by the database publisher and will not be stored the database container in ACR.

You can use Azure Container Registry to build these containers using a standard Dockerfile to provide build instructions. With Azure Container Registry, you can reuse any Dockerfile currently in your environment, which includes multi-staged builds.

Build the ratings-api image

The ratings API is a Node.js application that's built using Express, a Node.js web framework. The source code is on GitHub and already includes a Dockerfile, which builds images based on the Node.js *Alpine* container image.

Here, you'll clone the repository and then build the Docker image using the included Dockerfile. Use the built-in ACR functionality to build and push the container image into your registry by running the `az acr build` command.

1. Clone the repository to your Cloud Shell.

   bash
   ```
   git clone https://github.com/MicrosoftDocs/mslearn-aks-workshop-ratings-api.git
   ```

2. Change into the newly cloned directory.

   bash
   ```
   cd mslearn-aks-workshop-ratings-api
   ```

3. Run `az acr build`. This command builds a container image by using the Dockerfile. Then it pushes the resulting image to the container registry.

   Azure CLI
   ```
   az acr build \
       --registry $ACR_NAME \
       --image ratings-api:v1 .
   ```

   > **Note**
   >
   > Don't forget the period `.` at the end of the preceding command. It represents the source directory that contains the **Dockerfile**. In this case, it's the current directory. Because you didn't specify the name of a file with the `--file` parameter, the command looks for a file called **Dockerfile** in the current directory.

   After a few minutes, you'll see a response similar to this example.

   output
   ```
   2019/12/28 02:04:11 Successfully pushed image: acr4229.azurecr.io/ratings-api:v1
   2019/12/28 02:04:13 Step ID: build marked as successful (elapsed time in seconds: 240.205952)
   2019/12/28 02:04:13 Populating digests for step ID: build...
   2019/12/28 02:04:13 Successfully populated digests for step ID: build
   2019/12/28 02:04:13 Step ID: push marked as successful (elapsed time in seconds: 33.293102)
   2019/12/28 02:04:13 The following dependencies were found:
   2019/12/28 02:04:13
   - image:
       registry: acr4229.azurecr.io
       repository: ratings-api
       tag: v1
       digest: sha256:b35cc14b16e3a4f51b86d8ed61f74dcfabb00f463e015ed33ec1fef748c55abda
     runtime-dependency:
       registry: registry.hub.docker.com
       repository: library/node
       tag: 13.5-alpine
       digest: sha256:a5af7f4267a810a019c7c3732b3c463a892a61937d84ee952c34af2fb4860858d
     git: {}

   Run ID: ca2 was successful after 4m41s
   ```

   Make a note of the pushed image registry and name, for example, `acr4229.azurecr.io/ratings-api:v1`. You'll need this information when you configure the Kubernetes deployment.

Build the ratings-web image

The ratings front-end is a Node.js application that was built by using the Vue JavaScript framework and WebPack to bundle the code. The source code is on GitHub and already includes a Dockerfile, which builds images based on the Node.js Alpine image.

The steps you follow are the same as before. Clone the repository and then build the docker image using the included Dockerfile using the `az acr build` command.

1. First, change back to the home directory.

   bash
   ```
   cd ~
   ```

2. Clone the *ratings-web* repo.

   bash
   ```
   git clone https://github.com/MicrosoftDocs/mslearn-aks-workshop-ratings-web.git
   ```

3. Change into the newly cloned directory.

   bash
   ```
   cd mslearn-aks-workshop-ratings-web
   ```

4. Run `az acr build`. This command builds a container image by using the Dockerfile. Then it pushes the resulting image to the container registry.

   Azure CLI
   ```
   az acr build \
       --registry $ACR_NAME \
       --image ratings-web:v1 .
   ```

   In a few minutes, you'll see a response similar to this example.

   output
   ```
   2019/12/28 02:09:51 Successfully pushed image: acr4229.azurecr.io/ratings-web:v1
   2019/12/28 02:09:53 Step ID: build marked as successful (elapsed time in seconds: 26.612936)
   2019/12/28 02:09:53 Populating digests for step ID: build...
   2019/12/28 02:09:53 Successfully populated digests for step ID: build
   2019/12/28 02:09:53 Step ID: push marked as successful (elapsed time in seconds: 35.571607)
   2019/12/28 02:09:53 The following dependencies were found:
   2019/12/28 02:09:53
   - image:
       registry: acr4229.azurecr.io
       repository: ratings-web
       tag: v1
       digest: sha256:ae4bab55e74d057e48b05b45761eef8d1c71874d9cfeeef6e8c3c1178f01f0f2
     runtime-dependency:
       registry: registry.hub.docker.com
       repository: library/node
       tag: 13.5-alpine
       digest: sha256:a5af7f4267a810a019c7c3732b3c463a892a61937d84ee952c34af2fb4860858d
     git: {}

   Run ID: ca3 was successful after 1m9s
   ```

   Make a note of the pushed image registry and name, for example, `acr4229.azurecr.io/ratings-web:v1`. Use this information when you configure the Kubernetes deployment.

Verify the images

1. Run the following command in Cloud Shell to verify that the images were created and stored in the registry.

   Azure CLI
   ```
   az acr repository list \
       --name $ACR_NAME \
       --output table
   ```

   The output from this command looks similar to this example.

   output
   ```
   Result
   ------------
   ratings-api
   ratings-web
   ```

The images are now ready for use.

Configure the AKS cluster to authenticate to the container registry

We need to set up authentication between your container registry and Kubernetes cluster to allow communication between the services.

Let's integrate the container registry with the existing AKS cluster by supplying valid values for **AKS_CLUSTER_NAME** and **ACR_NAME**. You can automatically configure the required service principal authentication between the two resources by running the `az aks update` command.

Run the following command.

Azure CLI
```
az aks update \
    --name $AKS_CLUSTER_NAME \
    --resource-group $RESOURCE_GROUP \
    --attach-acr $ACR_NAME
```

Summary

In this exercise, you created a *container registry* for the Fruit Smoothies application. You then built and added container images for the *ratings-api* and *ratings-web* to the container registry. You then verified the container images, and configured your AKS cluster to authenticate to the container registry.

Next, you'll take the first step to deploy your ratings app. The first component you'll deploy is MongoDB as your document store database, and you'll see how to use the HELM package manager for Kubernetes.

Next unit: Exercise - Deploy MongoDB

Continue