Build and store container images with Azure Container Registry

Azure Container Registry is a managed Docker registry service based on the open-source Docker Registry 2.0. Container Registry is private, hosted in Azure, and allows you to build, store, and manage images for all types of container deployments. Learn how to build and store container images with Azure Container Registry, and deploy the image to Azure Container Instances.

In this module, you will:

- Deploy an Azure container registry
- Build a container image using Azure Container Registry Tasks
- Deploy the container to an Azure container instance
- Replicate the container image to multiple Azure regions

Introduction to Azure Container Registry

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

Container images can be pushed and pulled with Container Registry using the Docker CLI or the Azure CLI. Azure portal integration allows you to visually inspect the container images in your 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.

In addition to storing container images, Azure Container Registry Tasks can 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 using DevOps processes and tooling.

Important
You need your own Azure subscription to run the exercises in this lab, and you may incur charges. If you don't already have an Azure subscription, create a **free account** before you begin.

Exercise - Deploy Azure Container Registry

In this unit, you will create an Azure Container Registry using the Azure CLI.

Create an Azure container registry

1. Sign into the Azure portal with your Azure subscription.
2. Open the Azure Cloud Shell from the Azure portal using the Cloud Shell icon.

3. Create a new resource group with the name **learn-deploy-acr-rg** so that it will be easier to clean up these resources when you are finished with the module. If you choose a different resource group name, remember it for the rest of the exercises in this module. You also need to choose a region in which you want to create the resource group, for example in US East, *eastus*.

   Before running the command, replace the `<choose-a-location>` with the region of your choice.

```
az group create --name learn-deploy-acr-rg --location <choose-a-location>
```

Next, we'll create an Azure container registry with the `az acr create` command. The container registry name must be unique within Azure and contain between 5 and 50 alphanumeric characters.

In this example, a premium registry SKU is deployed. The premium SKU is required for geo-replication which we will use in one of the future exercises in this lab.

To begin, we'll define an environment variable in the Cloud Shell called **ACR_NAME** to hold the name we want to give our new container registry.

1. Run the following command to define a variable called ACR_NAME.

   Important
   Before running the command, replace `<registry-name>` with the unique name you want to give your new container registry. The registry name must be unique within Azure, and contain 5-50 **alphanumeric** characters. For more information on naming, see **Naming conventions for Azure resources** .

```
ACR_NAME=<registry-name>
```

2. Enter the following command into the cloud shell editor to create our new container registry.

```
az acr create --resource-group learn-deploy-acr-rg --name $ACR_NAME --sku
Premium
```

   The following snippet is an example response from the `az acr create` command. In this example, the registry name was *myACR*. Note that the loginServer value below is the registry name in lowercase, by default.

```
{
  "adminUserEnabled": false,
  "creationDate": "2018-08-15T19:19:07.042178+00:00",
  "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.ContainerRegist
ry/registries/myACR0007",
  "location": "eastus",
  "loginServer": "myacr.azurecr.io",
  "name": "myACR",
  "provisioningState": "Succeeded",
  "resourceGroup": "myResourceGroup",
  "sku": {
    "name": "Premium",
    "tier": "Premium"
  },
  "status": null,
  "storageAccount": null,
  "tags": {},
  "type": "Microsoft.ContainerRegistry/registries"
}
```

Important
Commands in the rest of this module will use the **ACR_NAME** variable value.

In this unit, you created an Azure Container Registry using the Azure CLI. We'll use that new container registry in the next unit when we build container images.

Exercise - Build container images with Azure Container Registry Tasks

Suppose your company makes use of container images to manage compute workloads. You use the local Docker tooling to build your container images.

As an alternative, you can now use Azure Container Registry Tasks to build these containers. Container Registry Tasks also allows for DevOps process integration with automated build on source code commit.

Let's automate the creation of a container image using Azure Container Registry Tasks.

Create a container image with Azure Container Registry Tasks

A standard Dockerfile provides build instructions. Azure Container Registry Tasks allows you to reuse any Dockerfile currently in your environment, including multi-staged builds.

We'll use a new Dockerfile for our example.

The first step is to create a new file named `Dockerfile`. You can use any text editor to edit the file. We'll use Cloud Shell Editor for this example.

1. Enter the following command into the Cloud Shell window to open the editor.

```
code
```

2. Copy the following contents into the editor.

```
FROM    node:9-alpine
ADD     https://raw.githubusercontent.com/Azure-Samples/acr-build-helloworld-node/master/package.json /
ADD     https://raw.githubusercontent.com/Azure-Samples/acr-build-helloworld-node/master/server.js /
RUN     npm install
EXPOSE  80
CMD     ["node", "server.js"]
```

3. Use the key combination `Ctrl+S` (`Cmd+S` for Mac) to save your changes. Name the file `Dockerfile` when prompted. Close the Editor window using the ... in the upper right corner of the window. Once the Editor window is closed use the ls command to list the files in current directory, and you should see the Dockerfile listed.

   This configuration adds a Node.js application to the `node:9-alpine` image. After that, it configures the container to serve the application on port 80 via the *EXPOSE* instruction.

4. Run the following Azure CLI command to build the container image from the Dockerfile. *$ACR_NAME* is the variable you defined in the preceding unit to hold your container registry name.

```
az acr build --registry $ACR_NAME --image helloacrtasks:v1 .
```

   Note
   Don't forget the period `.` at the end of the preceding command. It represents the source directory containing the docker file, which in our case is the current directory. Since we didn't specify the name of a file with the --file parameter, the command looks for a file called **Dockerfile** in our current directory.

Verify the image

1. Run the following command in the Cloud Shell to verify that the image has been created and stored in the registry.

```
az acr repository list --name $ACR_NAME --output table
```

The output from this command should look similar to the following:

```
Result
-------------
helloacrtasks
```

The `helloacrtasks` image is now ready to be used.

Exercise - Deploy images from Azure Container Registry

Container images can be pulled from Azure Container Registry using many container management platforms, such as Azure Container Instances, Azure Kubernetes Service, and Docker for Windows or Mac. Here, we will deploy our image to an Azure Container Instance.

About registry authentication

Azure Container Registry does not support unauthenticated access; all operations on a registry require a login. Registries support two types of identities:

- **Azure Active Directory identities**, including both user and service principals. Access to a registry with an Azure Active Directory identity is role-based, and identities can be assigned one of three roles: **reader** (pull access only), **contributor** (push and pull access), or **owner** (pull, push, and assign roles to other users).

- The **admin account** included with each registry. The admin account is disabled by default.

The admin account provides a quick option to try a new registry. You enable the account and use its username and password in workflows and apps that need access. Once you have confirmed that the registry works as expected, you should disable the admin account and use Azure Active Directory identities exclusively to ensure the security of your registry.

Important
Only use the registry admin account for early testing and exploration, and do not share the username and password. Disable the admin account and use only role-based access with Azure Active Directory identities to maximize the security of your registry.

Enable the registry admin account

In this exercise, we will enable the registry admin account and use it to deploy your image to an Azure Container Instance from the command line.

1. Run the following command in the Cloud Shell to enable the admin account on your registry.

```
az acr update -n $ACR_NAME --admin-enabled true
```

2. Now run the following command in the Cloud Shell to retrieve the username and password for the admin account you enabled in the preceding step.

```
az acr credential show --name $ACR_NAME
```

   Take note of the `username` and `password` values that are returned from this command. You will need them in this exercise.

Deploy a container with Azure CLI

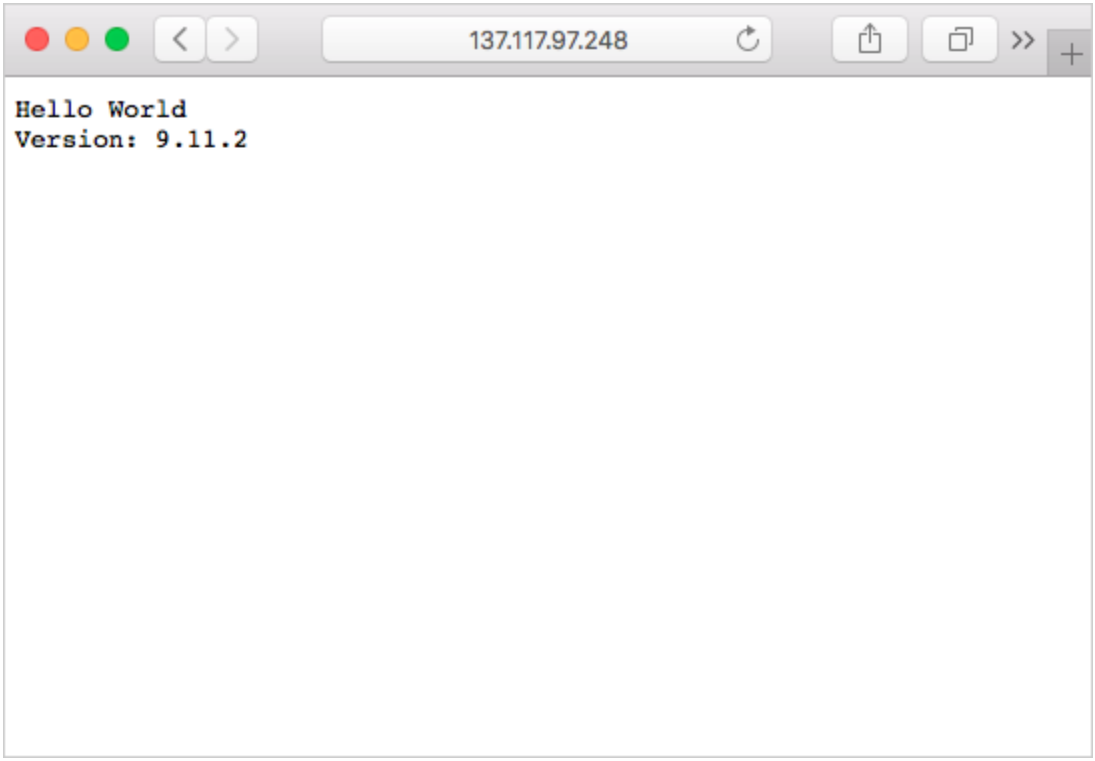
1. Execute the following `az container create` command to deploy a container instance. Replace `<username>` and `<password>` in the following command with your registry's admin username and password.

```
az container create \
    --resource-group learn-deploy-acr-rg \
    --name acr-tasks \
    --image $ACR_NAME.azurecr.io/helloacrtasks:v1 \
    --registry-login-server $ACR_NAME.azurecr.io \
    --ip-address Public \
    --location eastus \
    --registry-username <username> \
    --registry-password <password>
```

2. Get the IP address of the Azure container instance using the following command.

```
az container show --resource-group  learn-deploy-acr-rg --name acr-tasks --
query ipAddress.ip --output table
```

3. Open a browser and navigate to the IP address of the container. If everything has been configured correctly, you should see the following results:



Exercise - Replicate a container image to different Azure regions

Suppose your company has compute workloads deployed to several regions to make sure you have a local presence to serve your distributed customer base.

Your aim is to place a container registry in each region where images are run. This strategy will allow for network-close operations, enabling fast, reliable image layer transfers.

Geo-replication enables an Azure container registry to function as a single registry, serving several regions with multi-master regional registries.

A geo-replicated registry provides the following benefits:

- Single registry/image/tag names can be used across multiple regions
- Network-close registry access from regional deployments
- No additional egress fees, as images are pulled from a local, replicated registry in the same region as your container host
- Single management of a registry across multiple regions

Create a replicated region for an Azure Container Registry

In this exercise, you'll use the `az acr replication create` Azure CLI command to replicate your registry from one region to another.

1. Run the following command to replicate your registry to another region. In this example, we are replicating to the `japaneast` region. *$ACR_NAME* is the variable you defined earlier in the module to hold your container registry name.

```
az acr replication create --registry $ACR_NAME --location japaneast
```

Here's an example of what the output from this command looks like:

```
{
  "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myresourcegroup/providers/Microsoft.ContainerRegist
ry/registries/myACR0007/replications/japaneast",
  "location": "japaneast",
  "name": "japaneast",
  "provisioningState": "Succeeded",
  "resourceGroup": "myresourcegroup",
  "status": {
    "displayStatus": "Syncing",
    "message": null,
    "timestamp": "2018-08-15T20:22:09.275792+00:00"
  },
  "tags": {},
  "type": "Microsoft.ContainerRegistry/registries/replications"
}
```

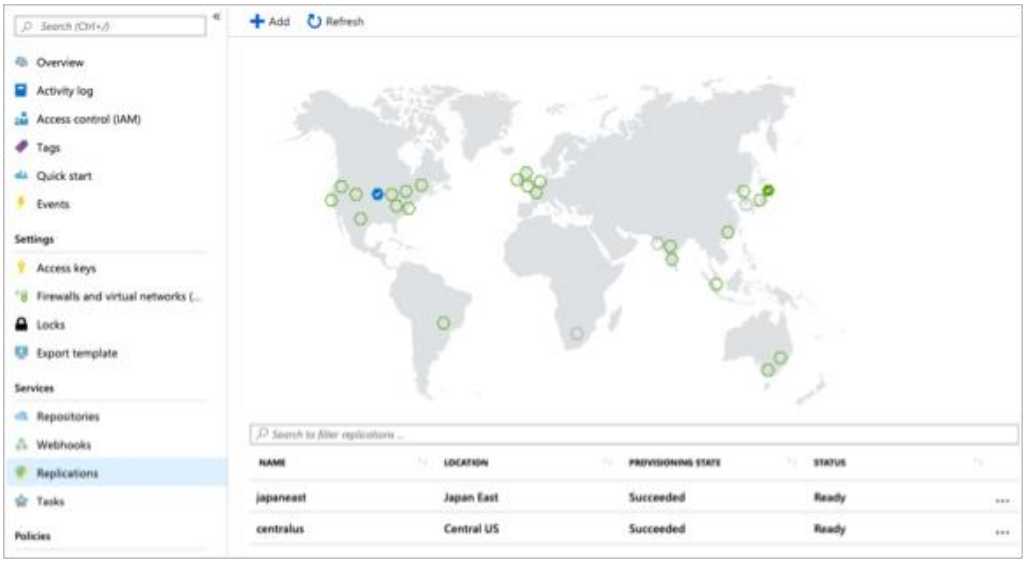
2. As a final step, retrieve all container image replicas created by running the following command.

```
az acr replication list --registry $ACR_NAME --output table
```

The output should look similar to the following:

```
NAME       LOCATION    PROVISIONING STATE    STATUS
---------  ----------  --------------------  --------
japaneast  japaneast   Succeeded             Ready
eastus     eastus      Succeeded             Ready
```

Keep in mind that you are not limited to the Azure CLI to list your image replicas. From within the Azure portal, selecting `Replications` for an Azure Container Registry displays a map that details current replications. Container images can be replicated to additional regions by selecting the regions on the map.



Summary

In this module you learned about the Azure Container Registry. You deployed your own registry, added a custom container, created a container image and deployed the image to an Azure Container Instance. Finally, you saw how easy it is to replicate the registry across Azure regions.

Clean up resources

In this module you created resources using your Azure subscription. You want to clean up these resources so that you will not continue to be charged for them.

1. In Azure, select **Resource groups** on the left.
2. Find the **learn-deploy-container-acr-rg** resource group, or whatever resource group name you used, and select it.
3. In the **Overview** tab of the resource group, select **Delete resource group**.
4. This opens a new dialog box. Type the name of the resource group again and select **Delete**. This will delete all of the resources we created in this module.

Learn More

Azure Container Registry (ACR) documentation

Docker on Azure