Build and store images by using Azure Container Registry

Azure Container Registry enables you to store Docker images in the cloud, in an Azure storage account.

In the example scenario, the team has decided to use Container Registry to host their Docker images. They can use Container Registry to create a Docker image registry in Azure, alongside their other Azure resources, and store their Docker images securely.

In this unit, you'll learn more about Container Registry and the advantages it provides for storing Docker images.

What is Container Registry?

Container Registry is an Azure service that you can use to create your own private Docker registries. Like Docker Hub, Container Registry is organized around repositories that contain one or more images. Container Registry also lets you automate tasks such as redeploying an app when an image is rebuilt.

Security is an important reason to choose Container Registry instead of Docker Hub:

- You have much more control over who can see and use your images.
- You can sign images to increase trust and reduce the chances of an image becoming accidentally (or intentionally) corrupted or otherwise infected.
- All images stored in a container registry are encrypted at rest.

Working with images in Container Registry is like working with Docker Hub, but offers a few unique benefits:

- Container Registry runs in Azure. The registry can be replicated to store images near where they're likely to be deployed.
- Container Registry is highly scalable, providing enhanced throughput for Docker pulls that can span many nodes concurrently. The Premium SKU of Container Registry includes 500 GiB of storage.

Using Container Registry

You create a registry by using either the Azure portal or the Azure CLI **acr create** command. In the following code example, the name of the new registry is *myregistry*:

```
az acr create --name myregistry --resource-group mygroup --sku standard --admin-enabled true
```

In addition to storing and hosting images, you can also use Container Registry to build images. Instead of building an image yourself and pushing it to Container Registry, use the CLI to upload the Docker file and other files that make up your image. Container Registry will then build the image for you. Use the **acr build** command to run a build:

```
az acr build --file Dockerfile --registry myregistry --image myimage .
```

Additional information about Azure Container Registry as well as a link to supported CLI commands to manage private registries are available in the learn more section of this module.

Exercise - Build and store an image by using Azure Container Registry

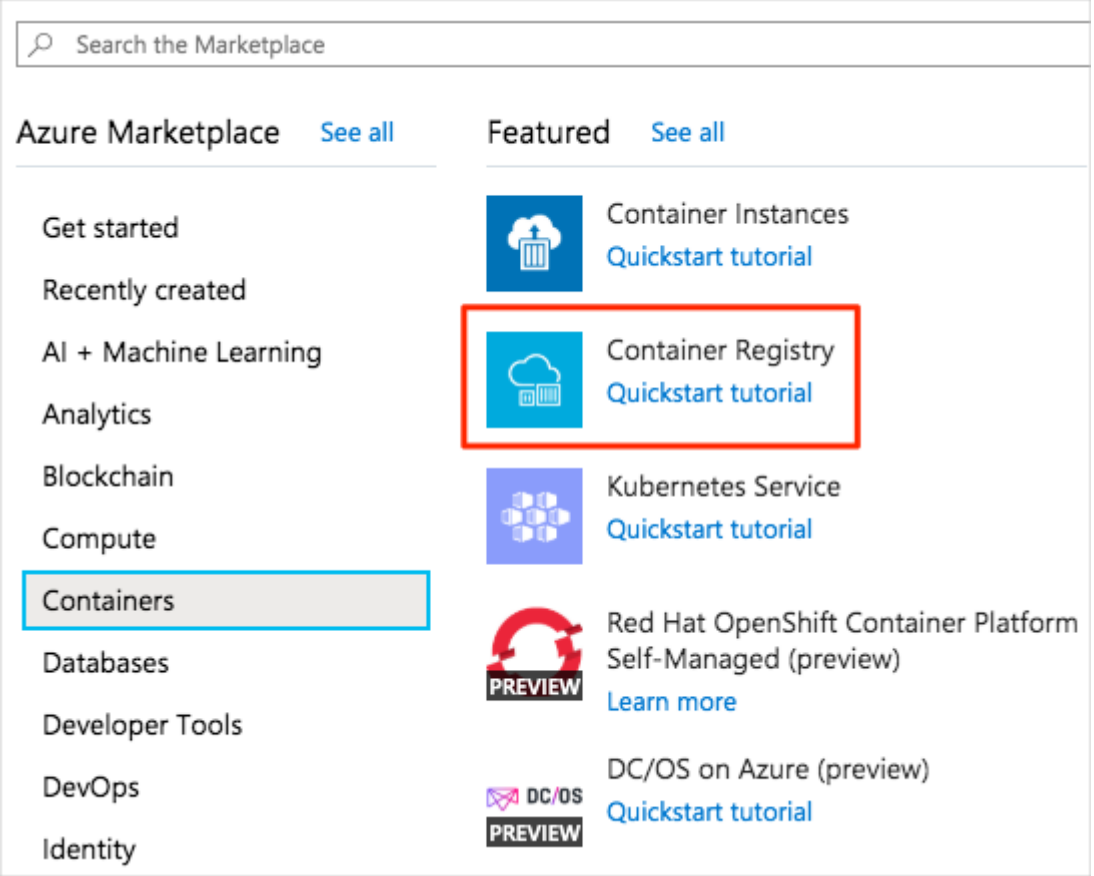
Azure Container Registry provides storage for Docker images in the cloud.

In the example scenario, the team needs to create a registry to store the images for their web apps.

In this unit, you'll use the Azure portal to create a new registry in Azure Container Registry. You'll build a Docker image from the source code for a web app and upload it to a repository in your registry. Finally, you'll examine the contents of the registry and the repository.

Create a registry in Azure Container Registry

1. Sign in to the Azure portal with your Azure subscription.
2. Select **Create a resource**, then select **Containers**, and then select **Container Registry**.



3. Specify the values in the following table for each of the properties:

TABLE 1

Property	Value
Registry name	Enter a unique name and make a note of it for later.
Subscription	Select your default Azure subscription in which you are allowed to create and manage resources.
Resource Group	Create a new resource group with the name **learn-deploy-container-acr-rg** so that it will be easier to clean up these resources when you're finished with the module. If you choose a different resource group name, remember it for the rest of the exercises in this module.
Location	Select a location that is close to you.
SKU	**Standard**

4. Select **Create**. Wait until the container registry has been created before you continue.

Build a Docker image and upload it to Azure Container Registry

1. In the Azure Cloud Shell in the portal, run the following command to download the source code for the sample web app. This web app is simple. It presents a single page that contains static text and a carousel control that rotates through a series of images.

```
git clone https://github.com/MicrosoftDocs/mslearn-deploy-run-container-app-service.git
```

2. Move to the source folder:

```
cd mslearn-deploy-run-container-app-service/dotnet
```

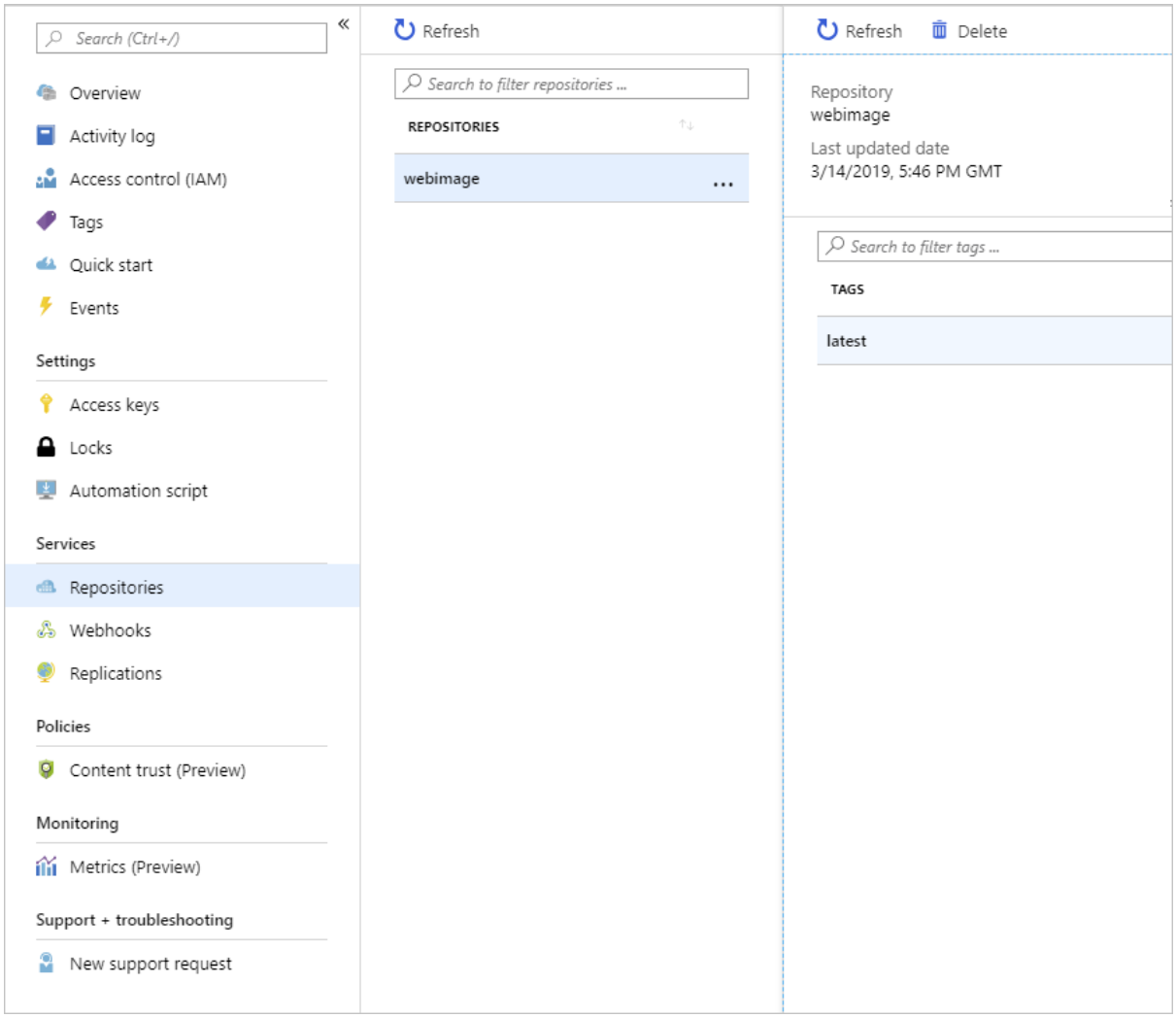
3. Run the following command. This command sends the folder's contents to Azure Container Registry, which uses the instructions in the Docker file to build the image and store it. Replace **<container_registry_name>** with the name of the registry you created earlier. **Take care not to leave out the .** character at the end of the command.

```
az acr build --registry <container_registry_name> --image webimage .
```

The Docker file contains the step-by-step instructions for building a Docker image from the source code for the web app. Azure Container Registry runs these steps to build the image, and as each step completes a message is generated. The build process should finish after a couple of minutes without any errors or warnings.

Examine the container registry

1. In the Azure portal , navigate to the Overview page for your container registry.
2. Under **Services**, select **Repositories**. You'll see a repository named `webimage`.
3. Select the `webimage` repository. It contains an image with the `latest` tag. This is the Docker image for the sample web app.



The Docker image that contains your web app is now available in your registry for deployment to App Service.

Deploy a web app by using an image from an Azure Container Registry repository

You can deploy a web app to Azure App Service directly from Azure Container Registry.

In the example scenario, the team wants to host the web app in App Service. They need to configure App Service to retrieve the image for the web app from the repository in Container Registry.

In this unit, you'll learn how you can configure App Service to deploy a web app from a repository in Container Registry.

Deploy a web app from a repository in Azure Container Registry

When you create a web app from a Docker image, you configure the following properties:

- The registry that contains the image. The registry can be Docker Hub, Azure Container Registry, or some other private registry.
- The image. This item is the name of the repository.
- The tag. This item indicates which version of the image to use from the repository. By convention, the most recent version is given the tag *latest* when it's built
- Startup File. This item is the name of an executable file or a command to be run when the image is loaded. It's equivalent to the command that you can supply to Docker when running an image from the command line by using `docker run`. If you're deploying a ready-to-run, containerized app that already has the `ENTRYPOINT` and/or `COMMAND` values configured, you don't need to fill this in.

After you've configured the web app, the Docker image is pulled and run as a "cold start" operation the first time a user attempts to visit the site. The app might take a few seconds to start initially, but thereafter it will be available immediately.

Exercise - Create and deploy a web app from a Docker image

Azure App Service provides the hosting environment for an Azure-based web app. You can configure App Service to retrieve the image for the web app from a repository in Azure Container Registry.

In the example scenario, the team has uploaded the image for the web app to Azure Container Registry and is now ready to deploy the web app.

In this unit, you'll create a new web app by using the Docker image stored in Azure Container Registry. You'll use App Service with a predefined App Service plan to host the web app.

Enable Docker access to the Azure Container Registry (ACR)

You'll use Docker to login to the registry and pull the web image that you want to deploy. Docker needs a username and password to perform this action. The ACR allows

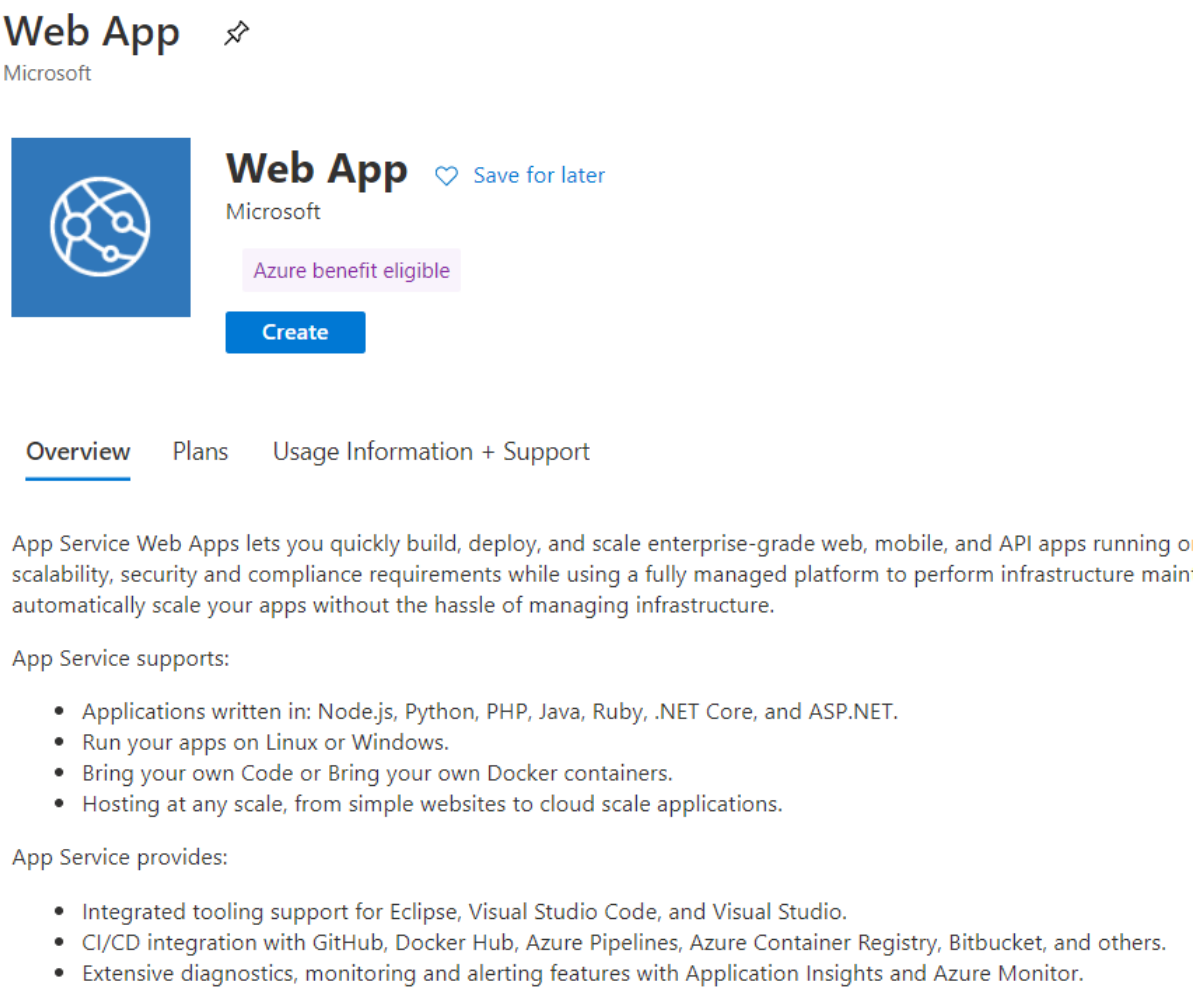
you to enable the registry name as the username and admin access key as the password to allow Docker to login to your container registry.

1. Sign in to the Azure portal , navigate to all resources.
2. Select the container registry you created earlier to navigate to the Overview page for the container registry.
3. Under **Settings**, select **Access keys**.
4. Set the **Admin user** option to **Enable**. This change saves automatically.

You're now ready to create your web app.

Create a web app

1. Select **Create a resource** > **Web** > **Web App**.



2. Specify these settings for each of the properties:

TABLE 1

Property	Value
Subscription	Select your default Azure subscription in which you are allowed to create and manage resources.
Resource Group	Reuse the existing resource group **learn-deploy-container-acr-rg**.
Name	Enter a unique name and make a note of it for later.
Publish	**Docker Container**
OS	**Linux**
App Service plan	Use the default.

3. Click **Next: Docker >**.
4. In the **Docker** tab, specify these settings for each of the properties:

TABLE 2

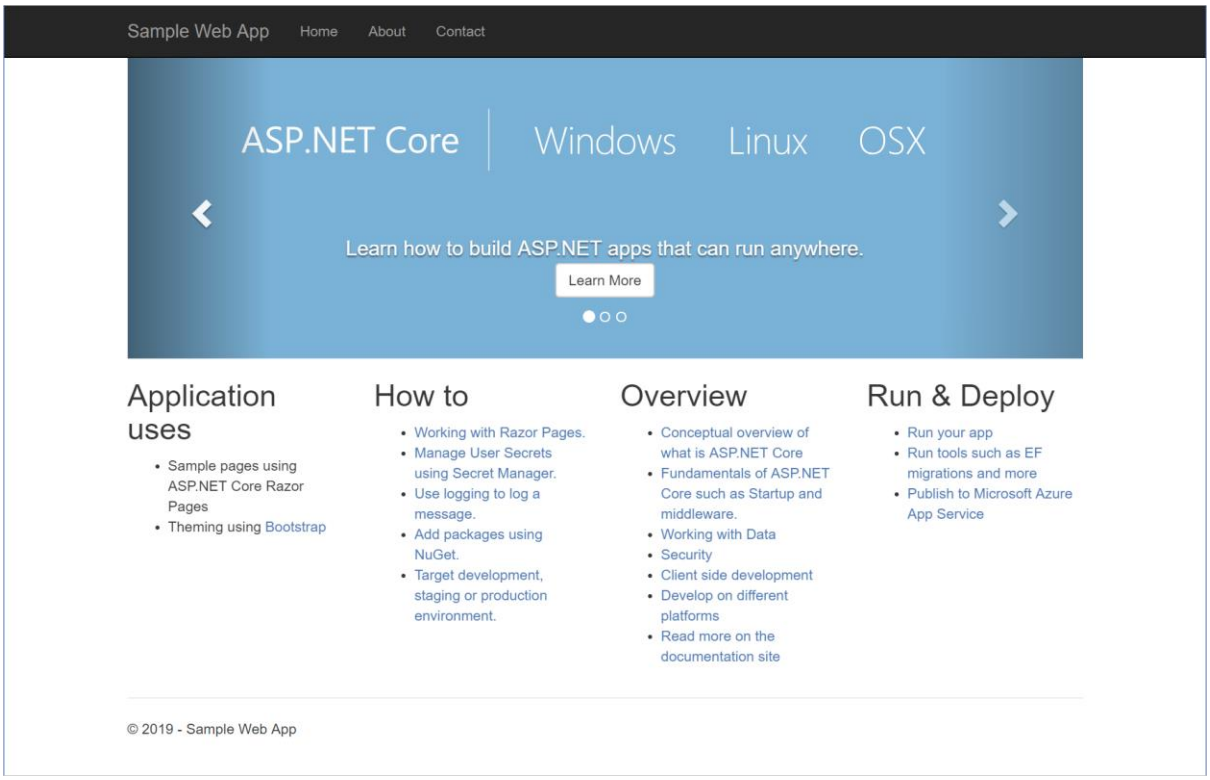
Property	Value
Options	**Single Container**
Image Source	**Azure Container Registry**
Registry	Select your registry.
Image	`webimage`
Tag	`latest`
Startup Command	Leave this empty.

5. Select **Review and create**, and then click **Create**. Wait until the web app has been deployed before you continue.

Test the web app

1. Use the **All resources** view in the Azure portal to go to the **Overview** page of the web app you just created.
2. Select the **Browse** button to open the site in a new browser tab.

3. After the cold-start delay while your app's Docker image loads and starts, you'll see a page like this:



App Service is now hosting the app from your Docker image.

Update the image and automatically redeploy the web app

Continuous deployment is a key feature for many fast-moving organizations. They need to deploy the latest version of their software quickly, but with the minimum of fuss.

In the example scenario, the development team adds new features and enhancements to the web app regularly. For this reason, the team has decided to adopt a continuous-deployment approach.

In this unit, you'll configure the continuous deployment of a web app that uses an image in Azure Container Registry.

What is a webhook?

Azure App Service supports continuous deployment using *webhooks*. A webhook is a service offered by Azure Container Registry. Services and applications can subscribe to

the webhook to receive notifications about updates to images in the registry. A web app that uses App Service can subscribe to an Azure Container Registry webhook to receive notifications about updates to the image that contains the web app. When the image is updated, and App Service receives a notification, your app automatically restarts the site and pulls the latest version of the image.

What is the Container Registry tasks feature?

You use the *tasks* feature of Container Registry to rebuild your image whenever its source code changes automatically. You configure a Container Registry task to monitor the GitHub repository that contains your code and trigger a build each time it changes. If the build finishes successfully, Container Registry can store the image in the repository. If your web app is set up for continuous integration in App Service, it receives a notification via the webhook and updates the app.

Let's use these two features to enable continuous integration from the App Service.

Enable continuous integration from App Service

The **Container settings** page of an App Service resource in the Azure portal automates the setup of continuous integration. If you turn on **Continuous Deployment**, App Service configures a webhook in your container registry to notify an App Service endpoint. Notifications from the registry that reach this endpoint cause your app to restart and pull the latest version of the container image.

Extend continuous integration to source control by using a Container Registry task

Container Registry tasks must be created from the command line. Unlike the `az acr build` command that we used earlier to build our image, the `az acr task create` command creates and registers a long-lived task.

The following command shows how to create a task called *buildwebapp*. The task monitors the GitHub repository for the sample web app used by this module. Each time a change is committed, the task builds the `webimage` Docker image from the source code in GitHub and stores it to your registry in Container Registry. Before running this command, you need to create a GitHub personal access token with permissions to create a webhook in your repository. For private repositories, the token will also need full repository read permissions.

```
az acr task create \
--registry <container_registry_name> \
--name buildwebapp --image webimage \
```

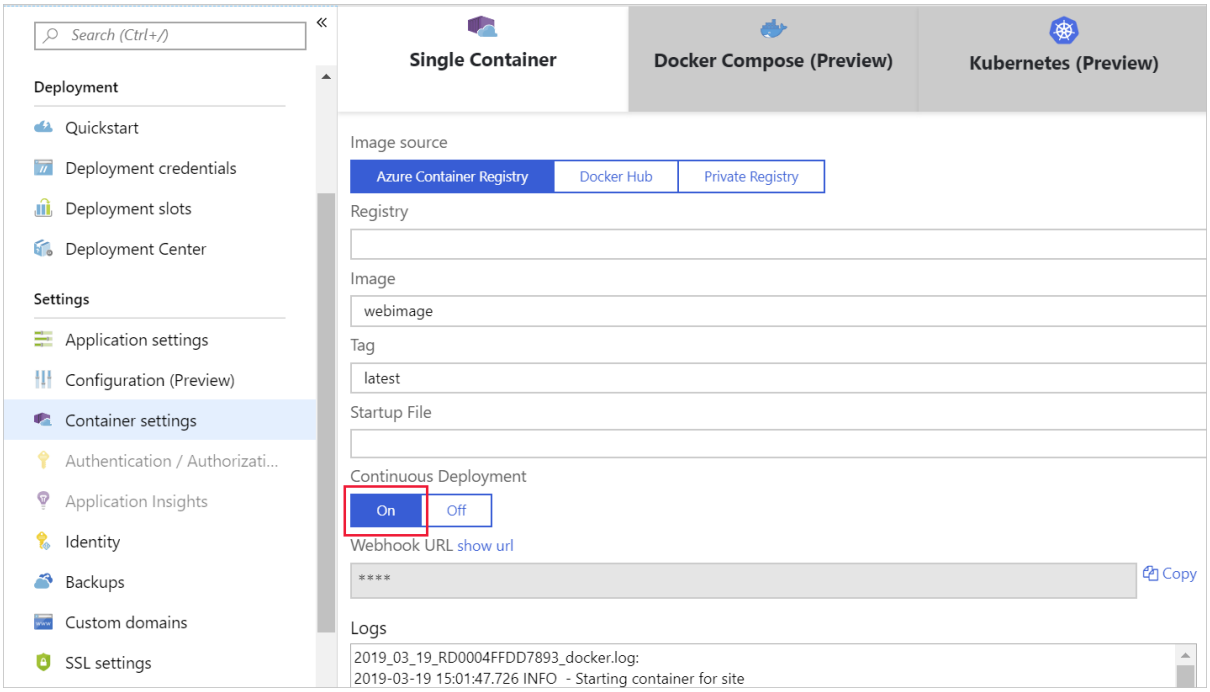
```
--context https://github.com/MicrosoftDocs/mslearn-deploy-run-container-app-service.git \
--branch master \
--file Dockerfile \

--git-access-token <access_token>
```

Exercise - Modify the image and redeploy the web app

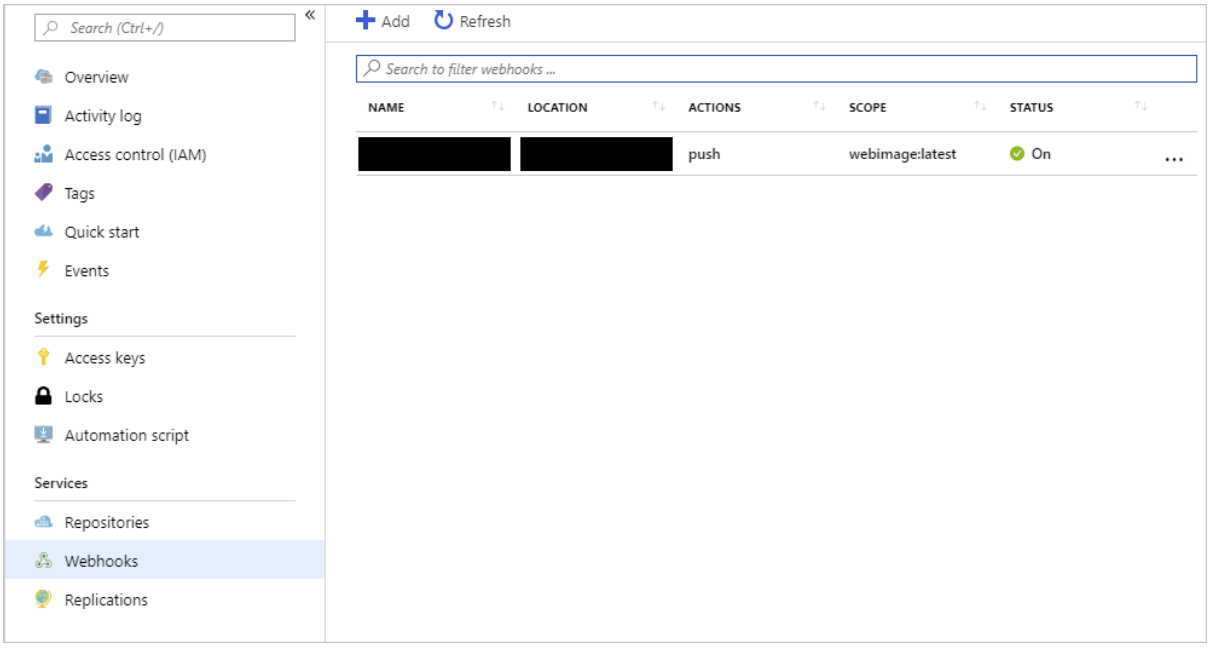
In this unit, you'll configure continuous deployment for the web app and create a webhook that links to the registry that contains the Docker image. Then, you'll make a change to the source code for the web app and rebuild the image. You'll visit the website that hosts the sample web app again and verify that the newest version is running.

Configure continuous deployment and create a webhook

1. Return to the Azure portal and go to the **Container settings** page of your web app.
2. Set **Continuous Deployment** to **On**, and then select **Save**. This setting configures a webhook that Container Registry uses to alert the web app that the Docker image has changed.



If you went to the **Webhooks** page for your container registry, you'd see the newly configured webhook:



Update the web app and test the webhook

1. In the Azure Cloud Shell, go to the dotnet/SampleWeb/Pages folder. This folder contains the source code for the HTML pages that are displayed by the web app:

```
cd ~/mslearn-deploy-run-container-app-service/dotnet/SampleWeb/Pages
```

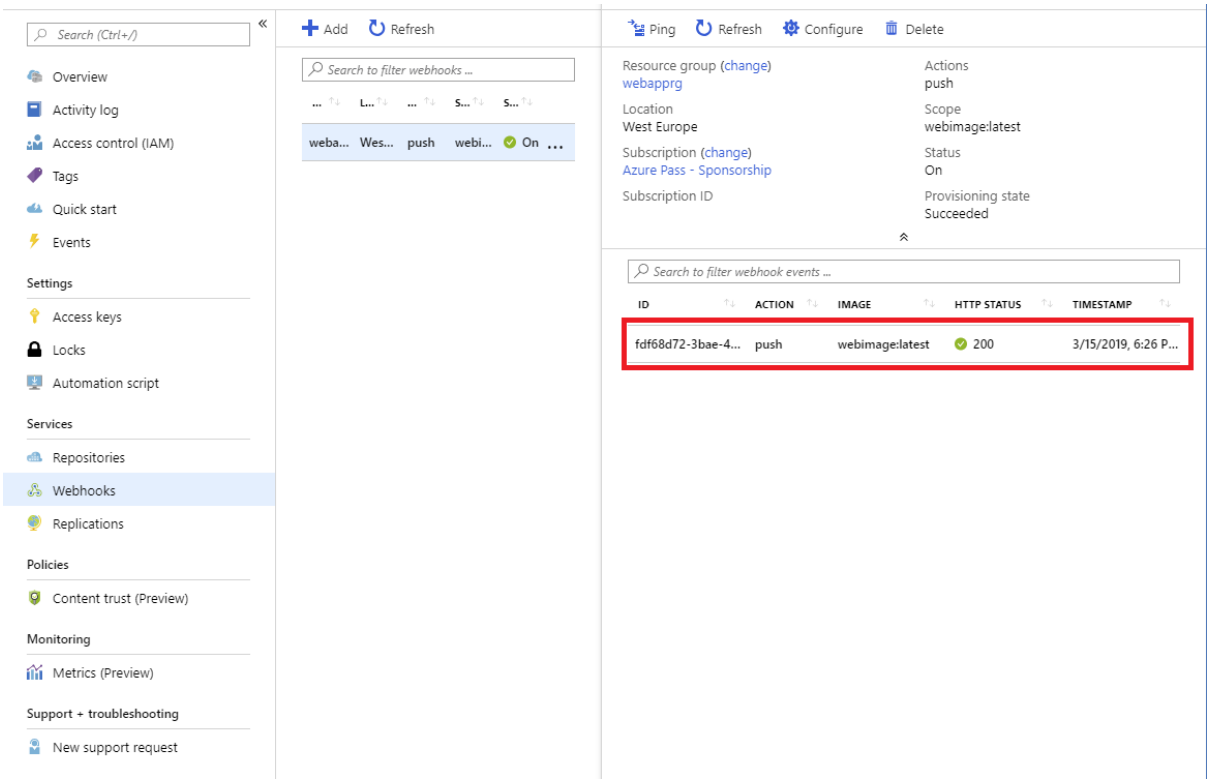
2. Run the following commands to replace the default page in the web app (Index.cshtml) with a new version that has an additional item in the carousel control. These commands simulate continued development on the app and add a new page to the carousel.

```
mv Index.cshtml Index.cshtml.old
mv Index.cshtml.new Index.cshtml
```

3. Run the next set of commands to rebuild the image for the web app and push it to Container Registry. Replace <container_registry_name> with the name of your registry. Don't forget the . at the end of the second command.

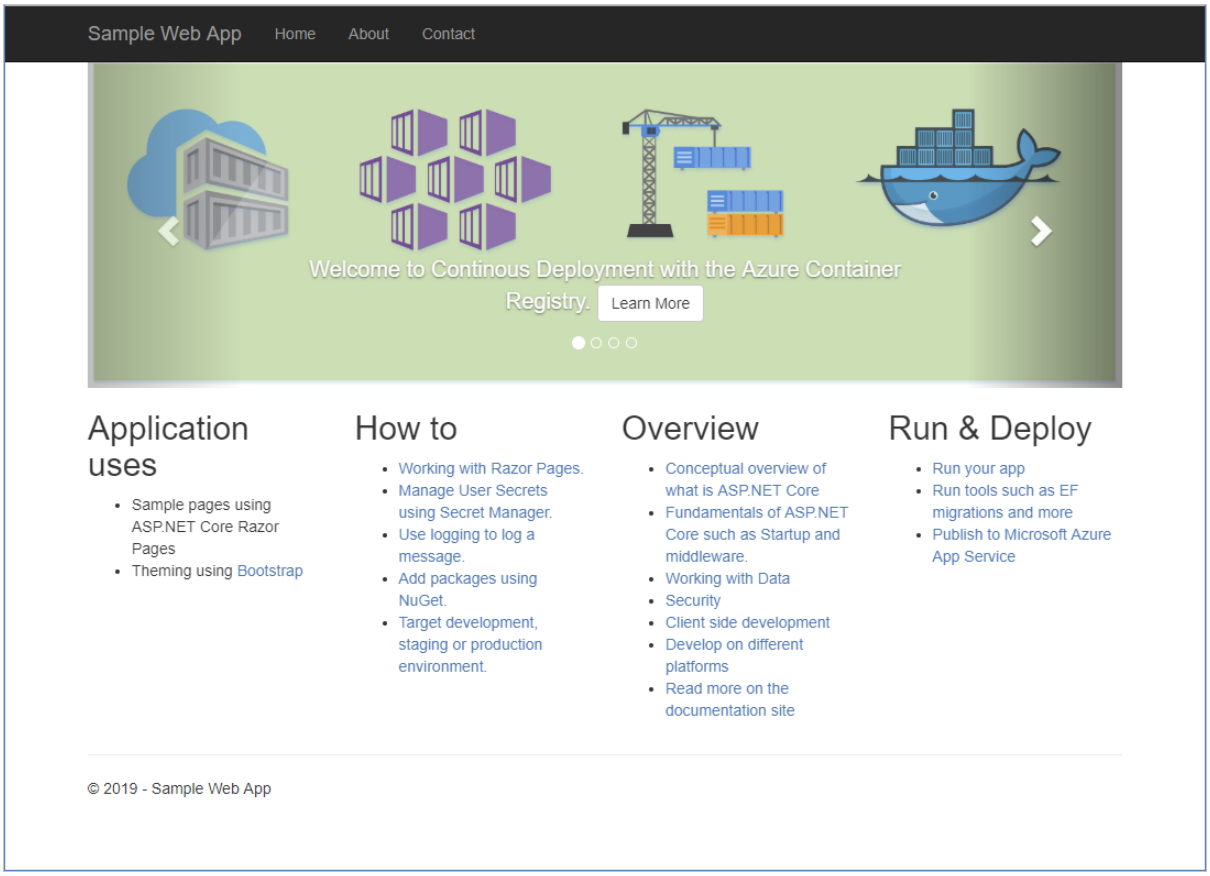
```
cd ~/mslearn-deploy-run-container-app-service/dotnet
az acr build --registry <container_registry_name> --image webimage .
```

4. Go to the **Webhooks** page of your container registry in the Azure portal and select the single webhook in the list.
5. At the bottom of the page, note that there's a record of the webhook that just fired in response to the build and push you ran.



Test the web app again

1. Go back to your web app in the browser. If you closed the tab for it earlier, you can go to the Overview page of the app in the Azure portal and select **Browse**. There will be a cold-start delay while the web app loads the new image from Container Registry.
2. Review the items in the carousel control. Note that the control now contains four pages. The new page looks like the following:



The web app has been updated and redeployed automatically based on the new image. The webhook service in your registry notified your web app that the container image had been modified, triggering an update.