Lab requirements

- This lab requires **Microsoft Edge** or an Azure DevOps supported browser.
- **Set up an Azure DevOps organization:** If you don't already have an Azure DevOps organization that you can use for this lab, create one by following the instructions available at Create an organization or project collection.
- Identify an existing Azure subscription or create a new one.
- Verify that you have a Microsoft account or an Azure AD account with the Contributor or the Owner role in the Azure subscription. For details, refer to List Azure role assignments using the Azure portal and View and assign administrator roles in Azure Active Directory.

Lab overview

In this lab, you will learn how to use an Azure DevOps CI/CD pipeline to build a custom Docker image, push it to Azure Container Registry, and deploy it as a container to Azure App Service.

Objectives

After you complete this lab, you will be able to:

- Build a custom Docker image by using an Microsoft hosted Linux agent.
- Push an image to Azure Container Registry.
- Deploy a Docker image as a container to Azure App Service by using Azure DevOps.

Estimated timing: 60 minutes

Instructions

Exercise 1: Configure the lab prerequisites

In this exercise, you will set up the prerequisites for the lab, which consist of a team project based on an Azure DevOps Demo Generator template and Azure resources, including an Azure App Service web app, an Azure Container Registry instance, and an Azure SQL database.

Task 1: Configure the team project

In this task, you will use Azure DevOps Demo Generator to generate a new project based on the Docker template.

Note: The Docker template-based project builds and deploys a containerized ASP.NET Core app to Azure App Service

1. On your lab computer, start a web browser and navigate to Azure DevOps Demo Generator. This utility site will automate the process of creating a new Azure DevOps project within your account that is prepopulated with content (work items, repos, etc.) required for the lab.
2. **Note**: For more information on the site, see https://docs.microsoft.com/en-us/azure/devops/demo-gen.

3. Click **Sign in** and sign in using the Microsoft account associated with your Azure DevOps subscription.
4. If required, on the **Azure DevOps Demo Generator** page, click **Accept** to accept the permission requests for accessing your Azure DevOps subscription.
5. On the **Create New Project** page, in the **New Project Name** textbox, type **Deploying Docker containers to Azure App Service web apps**, in the **Select organization** dropdown list, select your Azure DevOps organization, and then click **Choose template**.
6. In the list of templates, in the toolbar, click **DevOps Labs**, select the **DevOps Labs** header, click the **Docker** template, and then click **Select Template**.
7. Back on the **Create New Project** page, if prompted to install missing extensions, select the checkbox below the **Docker Integration** label and click **Create Project**.
8. **Note**: Wait for the process to complete. This should take about 2 minutes. In case the process fails, navigate to your DevOps organization, delete the project, and try again.
9. On the **Create New Project** page, click **Navigate to project**.

Task 2: Create Azure resources

In this task, you will use Azure Cloud Shell to create Azure resources required in this lab:

- Azure Container Registry
- Azure Web App for Containers
- Azure SQL Database

10. From the lab computer, start a web browser, navigate to the **Azure Portal**, and sign in with the user account that has the Owner role in the Azure subscription you will be using in this lab and has the role of the Global Administrator in the Azure AD tenant associated with this subscription.
11. In the web browser displaying the Azure portal, in the toolbar, click the **Cloud Shell** icon located directly to the right of the search text box.
12. If prompted to select either **Bash** or **PowerShell**, select **Bash**.
13. **Note**: If this is the first time you are starting **Cloud Shell** and you are presented with the **You have no storage mounted** message, select the subscription you are using in this lab, and select **Create storage**.
14. From the **Bash** session in the Cloud Shell pane, run the following to create variables representing the Azure region where you will deploy resources in this lab, the resource group containing these resources, as well as the names of these resources, including an Azure Container Registry instance, an Azure App Service plan name, an Azure web app name, an Azure SQL Database logical server name, and an Azure SQL database name:
15. From the Bash session in the Cloud Shell pane, run the following to create the resource group that will host Azure resources you deploy in this lab (replace the

<Azure_region> placeholder with the name of the Azure region, such as ‘eastus’, where you intend to deploy these resources):

16. CodeCopy

```
17.     LOCATION='<Azure_region>'
```

18. **Note**: You can identify Azure region names by running `az account list-locations -o table`

19. Run the following to create variables representing the names of Azure resources, including an Azure Container Registry instance, an Azure App Service plan name, an Azure web app name, an Azure SQL Database logical server name, and an Azure SQL database name:

20. CodeCopy

```
21.     RG_NAME='az400m1501a-RG'
    ACR_NAME=az400m151acr$RANDOM$RANDOM
    APP_SVC_PLAN='az400m1501a-app-svc-plan'
    WEB_APP_NAME=az400m151web$RANDOM$RANDOM
    SQLDB_SRV_NAME=az400m15sqlsrv$RANDOM$RANDOM
    SQLDB_NAME='az400m15sqldb'
```

22. **Note**: You can identify Azure region names by running `az account list-locations -o table`

23. Run the following to create all resources required Azure resources required for this lab:

24. CodeCopy

```
25.     az group create --name $RG_NAME --location $LOCATION
    az acr create --name $ACR_NAME --resource-group $RG_NAME --
   location $LOCATION --sku Standard --admin-enabled true
    az appservice plan create --name 'az400m1501a-app-svc-plan' --
   location $LOCATION --resource-group $RG_NAME --is-linux
    az webapp create --name $WEB_APP_NAME --resource-group $RG_NAME
   --plan $APP_SVC_PLAN --deployment-container-image-name
   elnably/dockerimagetest
    IMAGE_NAME=myhealth.web
    az webapp config container set --name $WEB_APP_NAME --resource-
   group $RG_NAME --docker-custom-image-name $IMAGE_NAME --docker-
   registry-server-url $ACR_NAME.azurecr.io/$IMAGE_NAME:latest --
   docker-registry-server-url Error! Hyperlink reference not valid. az sql
   server create --name $SQLDB_SRV_NAME --resource-group $RG_NAME --
```

```
location $LOCATION --admin-user sqladmin --admin-password
Pa55w.rd1234
 az sql db create --name $SQLDB_NAME --resource-group $RG_NAME --
server $SQLDB_SRV_NAME --service-objective S0 --no-wait
 az sql server firewall-rule create --name AllowAllAzure --
resource-group $RG_NAME --server $SQLDB_SRV_NAME --start-ip-
address 0.0.0.0 --end-ip-address 0.0.0.0
```

26. **Note**: Wait for the provisioning process to complete. This might take about 5 minutes.
27. Run the following to configure a connection string of the newly created Azure web app (replace the $SQLDB_SRV_NAME and $SQLDB_NAME placeholders with the values of the names of the Azure SQL Database logical server and its database instance, respectively):
28. CodeCopy

```
29.     CONNECTION_STRING="Data
Source=tcp:$SQLDB_SRV_NAME.database.windows.net,1433;Initial
Catalog=$SQLDB_NAME;User Id=sqladmin;Password=Pa55w.rd1234;"
 az webapp config connection-string set --name $WEB_APP_NAME --
resource-group $RG_NAME --connection-string-type SQLAzure --
settings defaultConnection="$CONNECTION_STRING"
```

30. In the web browser displaying the Azure portal, close the Cloud Shell pane, navigate to the **Resource groups** blade, and, on the **Resource groups** blade, select the **az400m1501a-RG** entry.
31. On the **az400m1501a-RG** resource group blade, review the listing of its resources.
32. **Note**: Record the name of the logical Azure SQL Database server. You will need it later in this lab.
33. On the **az400m1501a-RG** resource group blade, in the list of resources, click the entry representing the Container Registry instance.
34. On the container registry blade, in the vertical menu on the left side, in the **Settings** section, click **Access keys**.
35. On the **Access keys** blade of the container registry instance, identify the values of the **Registry name**, **Login server**, **Admin user**, and **password** entries.
36. **Note**: Record the values of **Registry name** and **Login server** (the registry names and the admin user name should match). You will need them later in this lab.

Exercise 2: Deploy a Docker container to Azure App Service web app using Azure DevOps

In this exercise, you will deploy a Docker container to Azure App Service web app by using Azure DevOps.

Task 1: Configure Continuous Integration (CI) and Continuous Delivery (CD)

In this task, you will use the Azure DevOps project you generated in the previous exercise in order to implement a CI/CD pipeline that builds and deploys a Docker container to an Azure App Service web app.

37. On your lab computer, switch to the web browser window displaying the Azure DevOps portal with the **Deploying Docker containers to Azure App Service web apps** project open, in the vertical menu bar at the far left of the Azure DevOps portal, click **Repos**.

38. **Note**: You will first modify the references to the Docker image.

39. On the **Docker** repository pane, in the list of files, select **docker-compose.ci.build.yml**.
40. On the **docker-compose.ci.build.yml** pane, click **Edit**, replace line **5** that references the target Docker image with `image: az400mp/aspnetcore-build:1.0-2.0`, select **Commit** and, when prompted for confirmation, click **Commit** again.
41. On the **Docker** repository pane, in the list of files, navigate to the **src/MyHealth.Web** folder and select **Dockerfile**.
42. On the **Dockerfile** pane, click **Edit**, replace line **1** that references the base Docker image with `FROM az400mp/aspnetcore1.0:1.0.4`, select **Commit** and, when prompted for confirmation, click **Commit** again.
43. In the web browser window displaying the Azure DevOps portal with the **Deploying Docker containers to Azure App Service web apps** project open, in the vertical menu bar at the far left of the Azure DevOps portal, click **Pipelines**.

44. **Note**: You will now modify the build pipeline.

45. On the **Pipelines** pane, click the entry representing the **MHCDocker.build** pipeline and, on the **MHCDocker.build** pane, click **Edit**.

46. **Note**: The build pipeline consists of the following tasks

Tasks	Usage
Run services	prepares the build environment by restoring the required packages
Build services	builds the **myhealth.web** image
Push services	pushes the **myhealth.web** image tagged with **$(Build.BuildId)** to container registry
Publish Artifact	allows for sharing dacpac for database deployment through Azure DevOps artifacts

47. On the **MHCDocker.build** pipeline pane, ensure that the **Pipeline** entry is selected and, in the **Agent Specifications** drop-down list, select **ubuntu-18.04**.

48. On the **MHCDocker.build** pipeline pane, in the list of tasks of the pipeline, click the **Run services** task, on the **Docker Compose** pane on the right side, in the **Azure subscription** dropdown list, select the entry representing the Azure subscription you are using in this lab, and click **Authorize** to create the corresponding service connection. When prompted, sign in using the account with the Owner role in the Azure subscription and the Global Administrator role in the Azure AD tenant associated with the Azure subscription.

49. **Note**: This step creates an Azure service connection, which defines and secures a connection to the target Azure subscription, using Service Principal Authentication (SPA).

50. In the list of tasks of the pipeline, with the **Run services** task selected, on the **Docker Compose** pane on the right side, in the **Azure Container Registry** dropdown list, select the entry representing the ACR instance you created earlier in this lab (**refresh the list if needed** or type the name of the login server).
51. Repeat the previous two steps to configure the **Azure subscription** and **Azure Container Registry** settings in the **Build services**, and **Push services** tasks, but, this time, instead of selecting your Azure subscription, select the newly created service connection.
52. On the same **MHCDocker.build** pipeline pane, at the top of the pane, click the down-facing caret next to the **Save & queue** button, click **Save** to save the changes, and, when prompted again, click **Save**.

53. **Note**: Next you will modify the release pipeline.

54. In the web browser window displaying the Azure DevOps portal, in the vertical menu bar at the far left of the Azure DevOps portal, in the **Pipelines** section, click **Releases**.
55. On the **Pipelines / Releases** pane, ensure that the **MHCDocker.release** entry is selected and click **Edit**.
56. On the **All pipelines / MHCDocker.release** pane, in the rectangle representing the **Dev** stage of the deployment, click the **2 jobs, 2 tasks** link.

57. **Note**: The release pipeline consists of the following tasks

Tasks	Usage
Execute Azure SQL: DacpacTask	deploys the dacpac artifact including the target schema and data to the Azure SQL database
Azure App Service deploy	pulls the docker image generated during the build stage from the designated container registry and deploys the image to the Azure App Service web app

58. In the list of tasks of the pipeline, click the **Execute Azure SQL: DacpacTask** task, on the **Azure SQL Database deployment** pane on the right side, in the **Azure subscription**

dropdown list, select the entry representing the Azure service connection you created earlier in this task.

59. In the list of tasks of the pipeline, click the **Azure App Service deploy** task, on the **Azure App Service deploy** pane on the right side, in the **Azure subscription** dropdown list, select the entry representing the Azure service connection you created earlier in this task and, in the **App Service name** dropdown list, select the entry representing the Azure App Service web app you deployed earlier in this lab.

60. **Note**: Next you will need to configure the agent pool information required for deployment.

61. Select the **DB deployment** job, on the **Agent job** pane on the right side, in the **Agent pool** dropdown list, select **Azure Pipelines** and, next, in the **Agent Specification** dropdown list, select **windows-2019**.
62. Select the **Web App deployment** job, on the **Agent job** pane on the right side, in the **Agent pool** dropdown list, select **Azure Pipelines** and, next, in the **Agent Specification** dropdown list, select **ubuntu-18.04**.
63. At the top of the pane, click the **Variables** header.
64. In the list of pipeline variables, set the values of the following variables:

Variable	Value
ACR	Azure container registry login name you recorded in the previous exercise of this lab, including the **azurecr.io** suffix
DatabaseName	**az400m15sqldb**
Password	**Pa55w.rd1234**
SQLadmin	**sqladmin**
SQLserver	the name of the Azure SQL Database logical server you recorded in the previous exercise of this lab, including the **database.windows.net** suffix

65. In the upper right corner of the pane, click the **Save** button to save the changes, and, when prompted again, click **OK**.

Task 2: Trigger build and release pipelines by using code commit
In this exercise, you will trigger the build and release pipelines by using code commit.

66. In the web browser window displaying the Azure DevOps portal, in the vertical menu bar at the far left of the Azure DevOps portal, click **Repos**.

67. **Note**: This will automatically display the **Files** pane.

68. On the **Files** pane, navigate to the **src/MyHealth.Web/Views/Home** folder and click the entry representing the **Index.cshtml** file and then click **Edit** to open it for editing.

69. On the **Index.cshtml** pane, in line **28**, change **JOIN US** to **CONTACT US** and then, in the upper right corner of the pane, click **Commit** and, when prompted for confirmation, click **Commit** again.

70. **Note**: This action would initiate an automatic build for the source code.

71. In the web browser window displaying the Azure DevOps portal, in the vertical menu bar at the far left of the Azure DevOps portal, click **Pipelines**.
72. On the **Pipelines** pane, click the entry representing the pipeline run triggered by the commit.
73. On the **MHCDocker.build** pane, click the entry representing the pipeline run.
74. On the **Summary** tab of the pipeline run, in the **Jobs** section, click the **Docker** entry and, on the resulting pane, monitor the progress of individual tasks until the job successfully completes.

75. **Note**: The build generates a Docker image and pushes it to Azure Container Registry. Once the build completes, you will be able to review its summary.

76. In the web browser window displaying the Azure DevOps portal, in the vertical menu bar at the far left of the Azure DevOps portal, in the **Pipelines** section, click **Releases**.
77. On the **Releases** pane, click the entry representing the latest release triggered by the successful build.
78. On the **MHCDocker.release > Release-1** pane, select the rectangle representing the **Dev** stage.
79. On the **MHCDocker.release > Release-1 > Dev** pane, monitor the progress of the release tasks until their successful completion.

80. **Note**: The release deploys the docker image generated by the build process to the App Service web app. Once the release completes, you can review its summary and logs.

81. After the release pipeline completes, switch to the web browser window displaying the Azure Portal and navigate to the blade of the Azure App Service web app you provisioned earlier in this lab.
82. On the App Service web app, click the **URL** link entry representing the target web app.

83. **Note**: This will automatically open a new web browser tab displaying the target web site.

84. Verify that the target web app displays the HealthClinic.biz web site, including the change that you applied to trigger the CI/CD pipeline.

Exercise 3: Remove the Azure lab resources

In this exercise, you will remove the Azure resources provisioned in this lab to eliminate unexpected charges.

Note: Remember to remove any newly created Azure resources that you no longer use. Removing unused resources ensures you will not see unexpected charges.

Task 1: Remove the Azure lab resources

In this task, you will use Azure Cloud Shell to remove the Azure resources provisioned in this lab to eliminate unnecessary charges.

85. In the Azure portal, open the **Bash** shell session within the **Cloud Shell** pane.
86. List all resource groups created throughout the labs of this module by running the following command:

87. ShellCopy

```
88.     az group list --query
   "[?starts_with(name,'az400m1501')].name" --output tsv
```

89. Delete all resource groups you created throughout the labs of this module by running the following command:

90. ShellCopy

```
91.     az group list --query
   "[?starts_with(name,'az400m1501')].[name]" --output tsv | xargs -
   L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

92. **Note**: The command executes asynchronously (as determined by the –nowait parameter), so while you will be able to run another Azure CLI command immediately afterwards within the same Bash session, it will take a few minutes before the resource groups are actually removed.

Review

In this lab, you used an Azure DevOps CI/CD pipeline to build a custom Docker image, pushed it to Azure Container Registry, and deployed it as a container to Azure App Service by using Azure DevOps.