

Lab requirements

- This lab requires **Microsoft Edge** or an [Azure DevOps supported browser](#).
- 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](#).
- If you don't already have a GitHub account that you can use for this lab, follow instructions available at [Signing up for a new GitHub account](#) to create one.

Lab overview

In this lab, you'll learn how to implement a GitHub Action workflow that deploys an Azure web app using DevOps Starter.

Objectives

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

- Implement a GitHub Action workflow by using DevOps Starter.
- Explain the basic characteristics of GitHub Action workflows.

Estimated timing: 30 minutes

Instructions

Exercise 1: Create a DevOps Starter project

In this exercise, you will use DevOps Starter to facilitate provisioning of a number of resources, including:

- A GitHub repository hosting:
 - Code of a sample .NET Core web site.
 - Azure Resource Manager templates that deploy an Azure web app hosting the web site code.
 - A workflow that builds, deploys, and tests the web site.
- An Azure web app, automatically deployed by using the GitHub workflow.

Task 1: Create DevOps Starter project

In this task, you will create an Azure DevOps Starter project that automatically sets up a GitHub repository, as well as creates and triggers a GitHub workflow that deploys an Azure web app, based on the content of the GitHub repository.

1. From the lab computer, start a web browser, navigate to the [Azure Portal](#), and sign in with the user account that has at least the Contributor role in the Azure subscription you are using in this lab.
2. In the Azure portal, search for and select the **DevOps Starter** resource type and, on the **DevOps Starter** blade, click either **+ Add**, **+ New**, or **+ Create**.

3. On the **Start fresh with a new application** page of the **DevOps Starter** blade, click the **here** link in the **Setting up DevOps starter with GitHub, click here** text.
4. **Note:** This will display the **DevOps starter settings** blade.
5. On the **DevOps starter settings** blade, ensure that the **GitHub** tile is selected and click **Done**.
6. Back on the **DevOps Starter** blade, click **Next: Framework >**.
7. On the **Choose an application framework** page of the **DevOps Starter** blade, select the **ASP.NET Core** tile and click **Next: Service >**.
8. On the **Select an Azure service to deploy the application** page of the **DevOps Starter** blade, ensure that the **Windows Web App** tile is selected and click **Next: Create >**.
9. On the **Select Repository and Subscription** page of the **DevOps Starter** blade, click **Authorize**.
10. **Note:** This will display the **Authorize Azure GitHub Actions** pop-up web browser window.
11. In the **Authorize Azure GitHub Actions** pop-up window, review the required permissions and click **Authorize AzureGithubActions**.
12. **Note:** This will redirect the web pop-up browser window to the Azure DevOps site, prompting you for your Azure DevOps information.
13. When prompted, in the pop-up web browser window, click **Continue**.
14. Back on the **Select Repository and Subscription** page of the **DevOps Starter** blade, specify the following settings and click **Review + Create**:

Setting	Value
Organization	the name of GitHub account
Repository	az400m08l01
Subscription	the name of the Azure subscription you are using for this lab
Web app name	any valid, globally unique host name in the azurewebsites.net DNS namespace
Location	the name of any Azure region in which you can provision an Azure web app. East US recommended

15. **Note:** Some **Location** may fail due to unavailable resources . **East US** recommended.
Note: Wait for the provisioning to complete. This should take about 1 minute.
16. On the **Deploy_DevOps_Project_az400m08l01 | Overview** blade, click **Go to Resource**.
17. On the **az400m08l01** blade, on the **GitHub Workflow** tile, click **Authorize**.
18. On the **GitHub Authorization** blade, click **Authorize** again.
19. Back on the **az400m08l01** blade, monitor the progress of actions on the **GitHub Workflow** tile.

20. **Note:** Wait for the build, deployment, and functional tests jobs of the GitHub workflow to complete. This should take about 5 minutes.

Task 2: Review the results of creating the DevOps Starter project

In this task, you will review the results of creating the DevOps Starter project.

21. In the web browser window displaying the Azure portal, on the **az400m08l01** blade, review the **GitHub Workflow** section and verify that the **build**, **deploy**, and **Functional tests** jobs completed successfully.
22. On the **az400m08l01** blade, review the **Azure resources** section and verify that they include an App Service web app instance and the corresponding Application Insights resource.
23. At the top of the **az400m08l01** blade, note the link to the **Workflow file** and the GitHub repository you created in the previous task.
24. At the top of the **az400m08l01** blade, click the link to the GitHub repository.
25. On the GitHub repository page, note the three folders labeled:
 - **.github\workflows** - containing the YAML-formatted workflow file
 - **Application** - containing the code of the sample web site
 - **ArmTemplates** - containing the Azure Resource Manager templates that the workflow uses to provision Azure resources
26. On the GitHub repository page, click **.github/workflows** and then, click the **devops-starter-workflow.yml** entry.
27. On the GitHub repository page displaying the content of the **devops-starter-workflow.yml**, review its content and note that it contains the **build**, **deploy**, and **Functional tests** job definitions.
28. On the GitHub repository page, in the toolbar, click **Actions**.
29. On the GitHub repository page, on the **Actions** tab, in the **All workflows** section, click the entry representing the most recent workflow run.
30. On the workflow run page, review the workflow status, as well as the listing of **Annotations** and **Artifacts**.
31. On the GitHub repository page, in the toolbar, click **Settings** and, on the **Settings** tab, click **Secrets**.
32. On the **Action secrets** pane, note the **AZURE_CREDENTIALS** entry, representing the credentials necessary to access the target Azure subscription.
33. Navigate to the **az400m08l01/Application/aspnet-core-dotnet-core/Pages/Index.cshtml** GitHub repository page and, in the upper right corner, click the pencil icon to switch to the edit mode.
34. Change the line 19 to `<div class="description line-1"> GitHub Workflow has been successfully updated</div>`.
35. Scroll down to the bottom of the page and click **Commit changes**.
36. On the GitHub repository page, in the toolbar, click **Actions**.
37. In the **All workflows** section, click the **Update Index.cshtml** entry.

38. In the **devops-starter-workflow.yml** section, monitor the progress of deployment and verify that it completed successfully.

39. **Note:** If the **action using “azure/CLI@1” fails**, commit the following change to the **devops-starter-workflow.yml** file (change the default azure cli version) and verify it completes successfully:

40. CodeCopy

```
41.     ```\n- name: Deploy ARM Template\n  uses: azure/CLI@v1\n  continue-on-error: false\n  with:\n    azcliversion: 2.29.2\n    inlineScript: |\n      az group create --name "$" --location "$"\n      az deployment group create --resource-group "$" --\ntemplate-file ./ArmTemplates/windows-webapp-template.json --\nparameters webAppName="$" hostingPlanName="$"\nappInsightsLocation="$" sku="$"\n    ```\n
```

42. Switch to the browser window displaying the DevOps Starter blade in the Azure portal and click the **Browse** link next to the **Application endpoint** entry.

43. In the newly opened web browser window, verify that the updated text representing the change you committed in the GitHub repo is displayed on the web app home page.

Exercise 2: 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.

44. In the Azure portal, open the **Bash** shell session within the **Cloud Shell** pane.

45. List all resource groups created throughout the labs of this module by running the following command:

46. ShellCopy

47. `az group list --query "[?starts_with(name,'az400m08l01')].name" --output tsv`

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

49. ShellCopy

50. `az group list --query "[?starts_with(name,'az400m08l01')].name" --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'`

51. **Note:** The command executes asynchronously (as determined by the `--no-wait` 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 implemented a GitHub Action workflow that deploys an Azure web app by using DevOps Starter.