**Learn continuous integration with GitHub Actions**

**Learn continuous integration with GitHub Actions**

* 1 hr 10 min
* Module
* 12 Units

Feedback

Advanced

Administrator

Developer

DevOps Engineer

Security Engineer

Security Operations Analyst

Service Adoption Specialist

Solution Architect

Technology Manager

Azure

Azure Artifacts

Azure Boards

Azure Cloud Services

Azure DevOps

Azure Pipelines

Azure Repos

Azure Test Plans

GitHub

This module details continuous integration using GitHub Actions and describes environment variables, artifacts, best practices, and how to secure your pipeline using encrypted variables and secrets.

**Learning objectives**

By the end of this module, you're able to:

* Implement Continuous Integration with GitHub Actions.
* Use environment variables.
* Share artifacts between jobs and use Git tags.
* Create and manage secrets.

[**Start**](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/1-introduction/)Add

**Prerequisites**

None

**This module is part of these learning paths**

* [AZ-400: Implement CI with Azure Pipelines and GitHub Actions](https://learn.microsoft.com/training/paths/az-400-implement-ci-azure-pipelines-github-actions/)
* [Explore Azure DevOps with GitHub to streamline your development process](https://learn.microsoft.com/training/paths/explore-azure-devops-with-github/)

**Module assessment**

Assess your understanding of this module. Sign in and answer all questions correctly to earn a pass designation on your profile.

[**Take the module assessment**](https://learn.microsoft.com/training/modules/learn-continuous-integration-github-actions/11-knowledge-check/)

* [Introduction](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/1-introduction)1 min
* [Describe continuous integration with actions](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/2-describe-continuous-integration-with-actions)3 min
* [Examine environment variables](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/3-examine-environment-variables)3 min
* [Share artifacts between jobs](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/4-share-artifacts-between-jobs)4 min
* [Examine Workflow badges](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/5-examine-workflow-badges)3 min
* [Describe best practices for creating actions](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/6-describe-best-practices-for-creating-actions)2 min
* [Mark releases with Git tags](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/7-mark-releases-git-tags)2 min
* [Create encrypted secrets](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/8-create-encrypted-secrets)3 min
* [Use secrets in a workflow](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/9-use-secrets-workflow)3 min
* [Implement GitHub Actions for CI/CD](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/10-implement-github-actions-for-ci-cd)40 min
* [Knowledge check](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/11-knowledge-check)5 min
* [Summary](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/12-summary)1 min

**Introduction**

Completed100 XP

* 1 minute

This module details continuous integration using GitHub Actions and describes environment variables, artifacts, best practices, and how to secure your pipeline using encrypted variables and secrets.

**Learning objectives**

After completing this module, students and professionals can:

* Implement Continuous Integration with GitHub Actions.
* Use environment variables.
* Share artifacts between jobs and use Git tags.
* Create and manage secrets.

**Prerequisites**

* Understanding of what DevOps is and its concepts.
* Familiarity with version control principles is helpful but isn't necessary.
* Beneficial to have experience in an organization that delivers software.

**Next unit: Describe continuous integration with actions**

**Describe continuous integration with actions**

Completed100 XP

* 3 minutes

It's an example of a basic continuous integration workflow created by using actions:

YAMLCopy

name: dotnet Build

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

strategy:

matrix:

node-version: [10.x]

steps:

- uses: actions/checkout@main

- uses: actions/setup-dotnet@v1

with:

dotnet-version: '3.1.x'

- run: dotnet build awesomeproject

* **On:** Specifies what will occur when code is pushed.
* **Jobs:** There's a single job called **build.**
* **Strategy:** It's being used to specify the Node.js version.
* **Steps:** Are doing a checkout of the code and setting up dotnet.
* **Run:** Is building the code.

**Next unit: Examine environment variables**

**Examine environment variables**

Completed100 XP

* 3 minutes

When using Actions to create CI or CD workflows, you'll typically need to pass variable values to the actions. It's done by using Environment Variables.

**Built-in environment variables**

GitHub provides a series of built-in environment variables. It all has a GITHUB\_ prefix.

**Note**

Setting that prefix for your variables will result in an error.

Examples of built-in environment variables are:

**GITHUB\_WORKFLOW** is the name of the workflow.

**GITHUB\_ACTION** is the unique identifier for the action.

**GITHUB\_REPOSITORY** is the name of the repository (but also includes the name of the owner in owner/repo format)

**Using variables in workflows**

Variables are set in the YAML workflow files. They're passed to the actions that are in the step.

YAMLCopy

jobs:

verify-connection:

steps:

- name: Verify Connection to SQL Server

- run: node testconnection.js

env:

PROJECT\_SERVER: PH202323V

PROJECT\_DATABASE: HAMaster

For more information on environment variables, including a list of built-in environment variables, see [Environment variables](https://docs.github.com/actions/learn-github-actions/environment-variables).

**Next unit: Share artifacts between jobs**

**Share artifacts between jobs**

Completed100 XP

* 4 minutes

When using Actions to create CI or CD workflows, you'll often need to pass artifacts created by one job to another.

The most common ways to do it are by using the **upload-artifact** and **download-artifact** actions.

**Upload-artifact**

This action can upload one or more files from your workflow to be shared between jobs.

You can upload a specific file:

YAMLCopy

- uses: actions/upload-artifact

with:

name: harness-build-log

path: bin/output/logs/harness.log

You can upload an entire folder:

YAMLCopy

- uses: actions/upload-artifact

with:

name: harness-build-logs

path: bin/output/logs/

You can use wildcards:

YAMLCopy

- uses: actions/upload-artifact

with:

name: harness-build-logs

path: bin/output/logs/harness[ab]?/\*

You can specify multiple paths:

YAMLCopy

- uses: actions/upload-artifact

with:

name: harness-build-logs

path: |

bin/output/logs/harness.log

bin/output/logs/harnessbuild.txt

For more information on this action, see [upload-artifact.](https://github.com/actions/upload-artifact)

**Download-artifact**

There's a corresponding action for downloading (or retrieving) artifacts.

YAMLCopy

- uses: actions/download-artifact

with:

name: harness-build-log

If no path is specified, it's downloaded to the current directory.

For more information on this action, see [download-artifact.](https://github.com/actions/download-artifact)

**Artifact retention**

A default retention period can be set for the repository, organization, or enterprise.

You can set a custom retention period when uploading, but it can't exceed the defaults for the repository, organization, or enterprise.

YAMLCopy

- uses: actions/upload-artifact

with:

name: harness-build-log

path: bin/output/logs/harness.log

retention-days: 12

**Deleting artifacts**

You can delete artifacts directly in the GitHub UI.

For details, you can see: [Removing workflow artifacts](https://docs.github.com/actions/managing-workflow-runs/removing-workflow-artifacts).

**Next unit: Examine Workflow badges**

**Examine Workflow badges**

Completed100 XP

* 3 minutes

Badges can be used to show the status of a workflow within a repository.

They show if a workflow is currently passing or failing. While they can appear in several locations, they typically get added to the README.md file for the repository.

Badges are added by using URLs. The URLs are formed as follows:

<https://github.com/><OWNER>/<REPOSITORY>/actions/workflows/<WORKFLOW\_FILE>/badge.svg

Where:

* AAAAA is the account name.
* RRRRR is the repository name.
* WWWWW is the workflow name.



They usually indicate the status of the default branch but can be branch-specific. You do this by adding a URL query parameter:

?branch=BBBBB

where:

* BBBBB is the branch name.

For more information, see: [Adding a workflow status badge](https://docs.github.com/actions/monitoring-and-troubleshooting-workflows/adding-a-workflow-status-badge).

**Next unit: Describe best practices for creating actions**

**Describe best practices for creating actions**

Completed100 XP

* 2 minutes

It's essential to follow best practices when creating actions:

* Create chainable actions. Don't create large monolithic actions. Instead, create smaller functional actions that can be chained together.
* Version your actions like other code. Others might take dependencies on various versions of your actions. Allow them to specify versions.
* Provide the **latest** label. If others are happy to use the latest version of your action, make sure you provide the **latest** label that they can specify to get it.
* Add appropriate documentation. As with other codes, documentation helps others use your actions and can help avoid surprises about how they function.
* Add details **action.yml** metadata. At the root of your action, you'll have an **action.yml** file. Ensure it has been populated with author, icon, expected inputs, and outputs.
* Consider contributing to the marketplace. It's easier for us to work with actions when we all contribute to the marketplace. Help to avoid people needing to relearn the same issues endlessly.

**Next unit: Mark releases with Git tags**

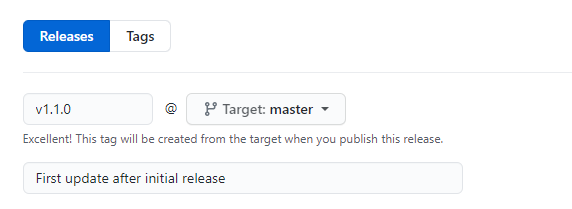
**Mark releases with Git tags**

Completed100 XP

* 2 minutes

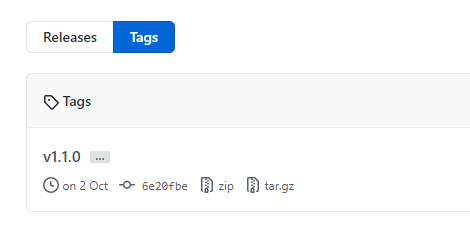
Releases are software iterations that can be packed for release.

In Git, releases are based on Git tags. These tags mark a point in the history of the repository. Tags are commonly assigned as releases are created.



Often these tags will contain version numbers, but they can have other values.

Tags can then be viewed in the history of a repository.



For more information on tags and releases, see: [About releases](https://docs.github.com/repositories/releasing-projects-on-github/about-releases).

**Next unit: Create encrypted secrets**

**Create encrypted secrets**

Completed100 XP

* 3 minutes

Actions often can use secrets within pipelines. Common examples are passwords or keys.

In GitHub actions, It's called **Secrets**.

**Secrets**

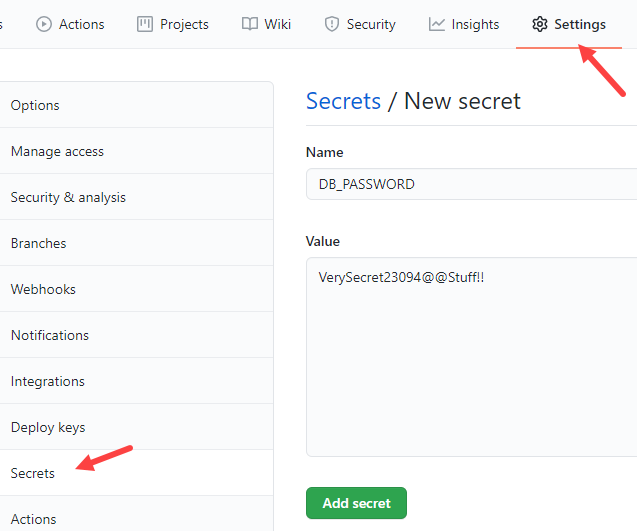
Secrets are similar to environment variables but encrypted. They can be created at two levels:

* Repository
* Organization

If secrets are created at the organization level, access policies can limit the repositories that can use them.

**Creating secrets for a repository**

To create secrets for a repository, you must be the repository's owner. From the repository **Settings**, choose **Secrets**, then **New Secret**.



For more information on creating secrets, see [Encrypted secrets](https://docs.github.com/actions/security-guides/encrypted-secrets).

**Next unit: Use secrets in a workflow**

**Use secrets in a workflow**

Completed100 XP

* 3 minutes

Secrets aren't passed automatically to the runners when workflows are executed. To make a secret available to an action, you must set the secret as an input or environment variable in the workflow file. To accomplish this, you can use the secrets context as illustrated in following example:

YAMLCopy

steps:

- name: Test Database Connectivity

with:

db\_username: ${{ secrets.DBUserName }}

db\_password: ${{ secrets.DBPassword }}

**Referencing secrets from the command line**

When referencing secrets from the command line, you can treat them like shell environment variables, as illustrated by the following example:

steps:

YAMLCopy

- shell: pwsh

env:

DB\_PASSWORD: ${{ secrets.DBPassword }}

run: |

db\_test "$env:DB\_PASSWORD"

It's important to follow this approach, which minimizes the risk of their accidental exposure. Secrets passed between processes from the command line can be viewed by using the PowerShell command and captured in security audit logs.

**Using secrets in if: conditionals**

Since secrets can't be directly referenced in if: conditionals, as a workaround, consider setting secrets as job-level environment variables, then referencing the environment variables to conditionally run steps in the job, as in the following example:

YAMLCopy

name: Run a step if a secret has been set

on: push

jobs:

samplejob:

runs-on: ubuntu-latest

env:

job\_secret: ${{ secrets.JobSecret }}

steps:

- if: ${{ env.job\_secret != '' }}

run: echo 'the secret has a value set.'

- if: ${{ env.job\_secret == '' }}

run: echo ' the secret does not have a value set.'

**Limitations**

Workflows can use up to 100 secrets, regardless of their location. Secrets are limited to 48 KB in size. To use secrets that are larger than 48 KB, you can use a workaround to store secrets in your repository and save the decryption passphrase as a secret on GitHub. For details regarding this procedure, refer to [Storing large secrets](https://docs.github.com/en/actions/security-guides/using-secrets-in-github-actions#storing-large-secrets) topic in GitHub documentation.

Note that with the exception of GITHUB\_TOKEN, secrets aren't passed to the runner when a workflow is triggered from a forked repository.

**Next unit: Implement GitHub Actions for CI/CD**

[**Previous**](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/8-create-encrypted-secrets/)

**Implement GitHub Actions for CI/CD**

Completed100 XP

* 40 minutes

**Estimated time:** 40 minutes.

**Scenario**

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

**Objectives**

After completing this lab, you'll be able to:

* Implement a GitHub Action workflow for CI/CD.
* Explain the basic characteristics of GitHub Action workflows.

**Requirements**

* This lab requires **Microsoft Edge** or an [Azure DevOps-supported browser](https://learn.microsoft.com/en-us/azure/devops/server/compatibility).
* Identify an existing Azure subscription or create a new one.
* Verify that you have a Microsoft or Entra account with the Contributor or the Owner role in the Azure subscription. For details, refer to [List Azure role assignments using the Azure portal](https://learn.microsoft.com/en-us/azure/active-directory/roles/manage-roles-portal).
* If you don't already have a GitHub account that you can use for this lab, follow the instructions available at [Signing up for a new GitHub account](https://docs.github.com/get-started/signing-up-for-github/signing-up-for-a-new-github-account) to create one.

**Exercises**

During this lab, you'll complete the following exercises:

* Exercise 0: Import eShopOnWeb to your GitHub Repository.
* Exercise 1: Setup your GitHub Repository and Azure access.
* Exercise 2: Remove the Azure lab resources.

[Screenshot of a launch button, which will take you to the lab.](https://go.microsoft.com/fwlink/?linkid=2270303)

**Next unit: Knowledge check**

[**Previous**](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/9-use-secrets-workflow/)

**Knowledge check**

Completed200 XP

* **Module assessment**
* 5 minutes

 Answer 100% of questions correctly in order to pass. [**Retake**](https://learn.microsoft.com/en-us/training/modules/learn-continuous-integration-github-actions/11-knowledge-check)

Dismiss alert

Choose the best response for each question.

**Check your knowledge**

Top of Form

**1.**

**Which of the following choices is where the database passwords that are needed in a CI pipeline should be stored?**

Repo.

**Incorrect. Secrets are similar to environment variables but encrypted.**

action.yml.

Encrypted Secrets.

**Correct. Secrets are similar to environment variables but encrypted.**

**2.**

**Which of the following files is the metadata for an action held?**

workflow.yml.

**Incorrect. Add details action.yml metadata. At the root of your action, you'll have an action.yml file. Make sure it has been populated with author, icon, any expected inputs, and outputs.**

action.yml.

**Correct. Add details action.yml metadata. At the root of your action, you'll have an action.yml file. Make sure it has been populated with author, icon, any expected inputs, and outputs.**

meta.yml.

**3.**

**Which of the following choices is how can the status of a workflow be shown in a repository?**

Using Badges.

**Correct. Badges can be used to show the status of a workflow within a repository.**

Status Files.

**Incorrect. Badges can be used to show the status of a workflow within a repository.**

Conversation Tab.

Bottom of Form

**All units complete:**