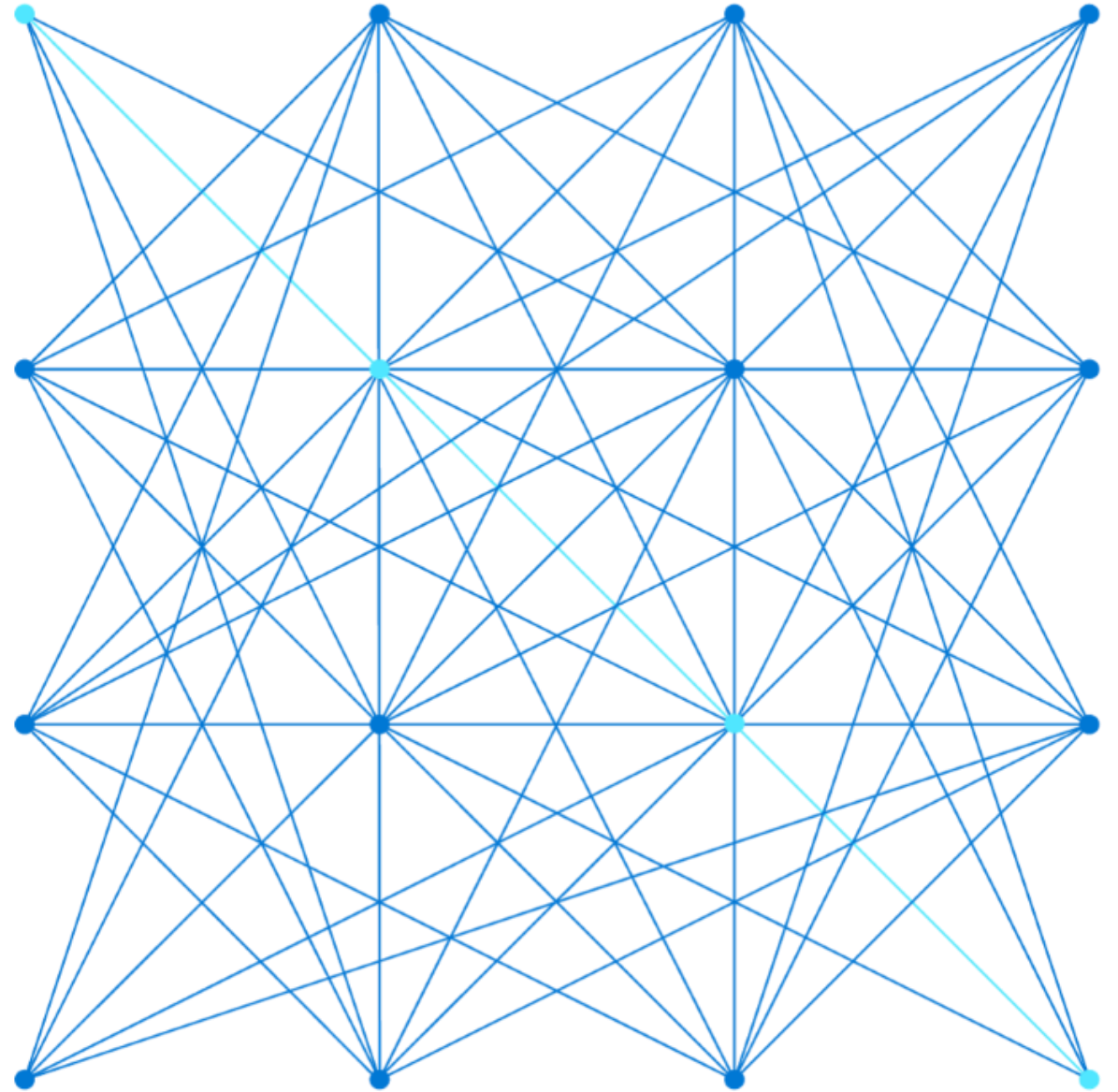


AZ-400.00

Module 8: Implementing Continuous Integration with GitHub Actions



Lesson 01: Module overview



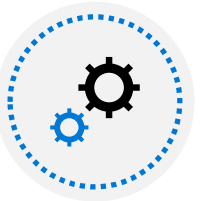
Module overview



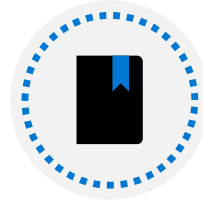
Lesson 1: Module overview



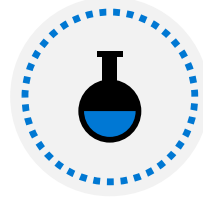
Lesson 2: GitHub Actions



Lesson 3: Continuous integration with GitHub Actions



Lesson 4: Securing secrets for GitHub Actions



Lesson 5: Lab



Lesson 6: Module review and takeaways

Learning objectives

After completing this module, students will be able to:



Create and work with GitHub Actions and workflows



Implement Continuous Integration with GitHub Actions

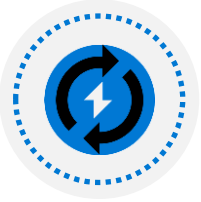
Lesson 02: GitHub Actions



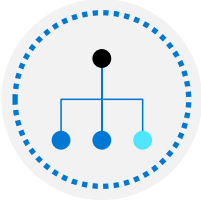
What are actions?



Automations within the GitHub environment



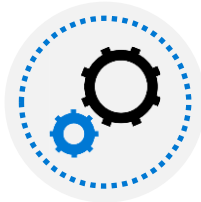
Often used to build CI/CD implementations



Based on YAML files living within GitHub repositories



Executed on GitHub or self-hosted runners



Large number of existing actions in the GitHub Marketplace

Actions flow

Events trigger workflows:

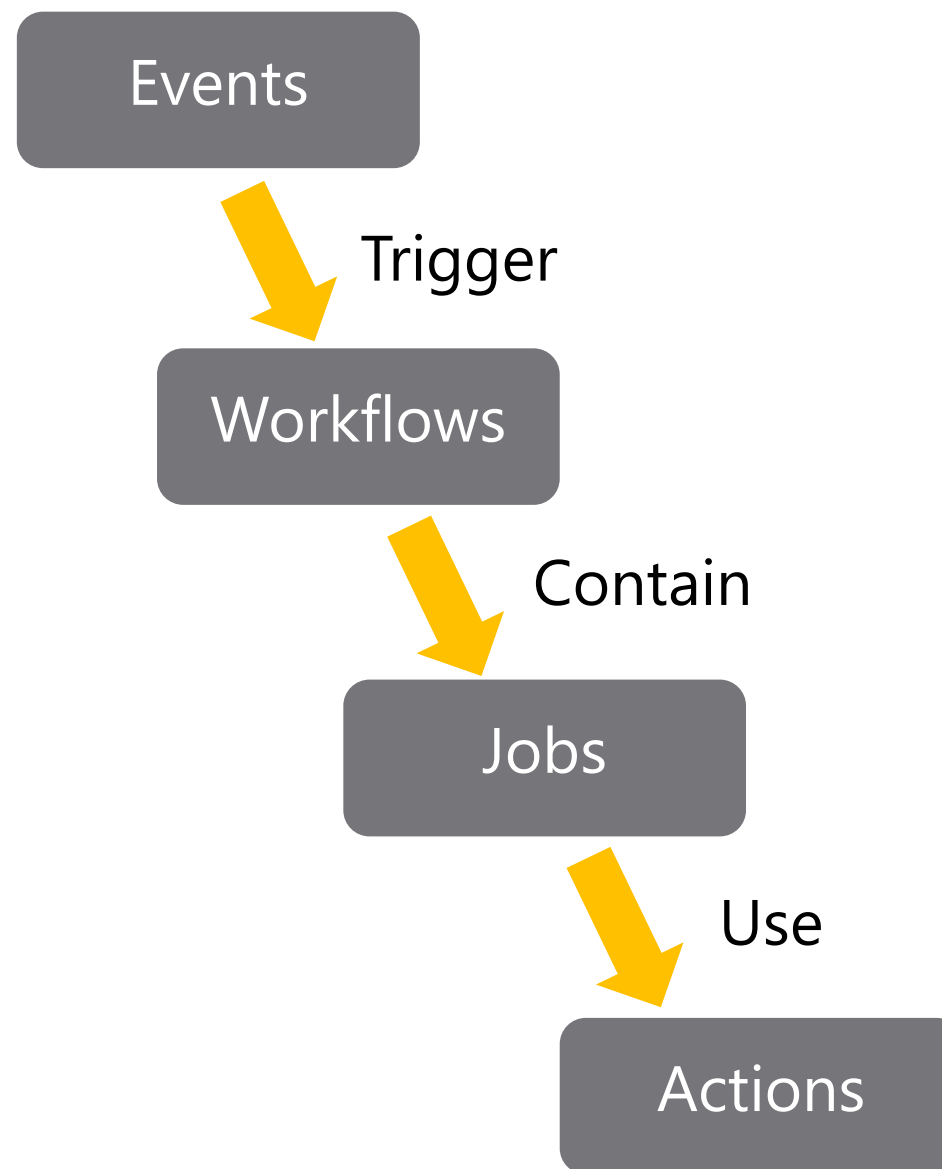
- Schedule, code, etc.

Workflows contain jobs:

- May contain multiples

Jobs use actions:

- Configured within steps



Workflows

Define the required automation:

Events, Jobs

Written in YAML

Stored in **.github/workflows**

Starter workflows available

```
# .github/workflows/build.yml
name: Node Build

on: [push]

jobs:
  mainbuild:

    runs-on: ${{ matrix.os }}

    strategy:
      matrix:
        node-version: [12.x]
        os: [windows-latest]

    steps:
      - uses: actions/checkout@v1
      - name: Run node.js on latest Windows
        uses: actions/setup-node@v1
        with:
          node-version: ${{ matrix.node-version }}
      - name: Install NPM and build
        run: |
          npm ci
          npm run build
```

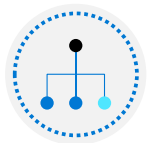
Standard workflow syntax elements



Name: the name of the workflow (optional but recommended)



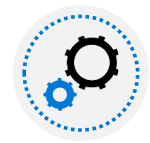
On: the event that triggers the workflow



Jobs: the list of jobs to be executed



Runs-on: the target runner



Steps: the list of steps to execute



Uses: a predefined action to retrieve



Run: tells the job to execute a command on the runner

Events

Defined by the **on** clause

- Scheduled events
- Code events
- Manual events
- Webhook events
- External events

```
on:  
  gollum
```

```
on:  
  schedule:  
    - cron: '0 8-17 * * 1-5'
```

```
on:  
  pull_request
```

```
on:  
  [push, pull_request]
```

```
on:  
  pull_request:  
    branches:  
      - develop
```

Jobs

Workflows contain one or more jobs. Jobs contain a set of steps to execute in order.

```
jobs:
  startup:
    runs-on: ubuntu-latest
    steps:
      - run: ./setup_server_configuration.sh
  build:
    steps:
      - run: ./build_new_server.sh
```

Jobs run in parallel by default but can be configured with dependencies.

```
jobs:
  startup:
    runs-on: ubuntu-latest
    steps:
      - run: ./setup_server_configuration.sh
  build:
    needs: startup ←
    steps:
      - run: ./build_new_server.sh
```

Runners



Job steps execute on runners:

Shell scripts

Actions

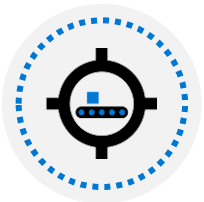


For JavaScript code, Node.js hosted runners:

Windows

MacOS

Linux



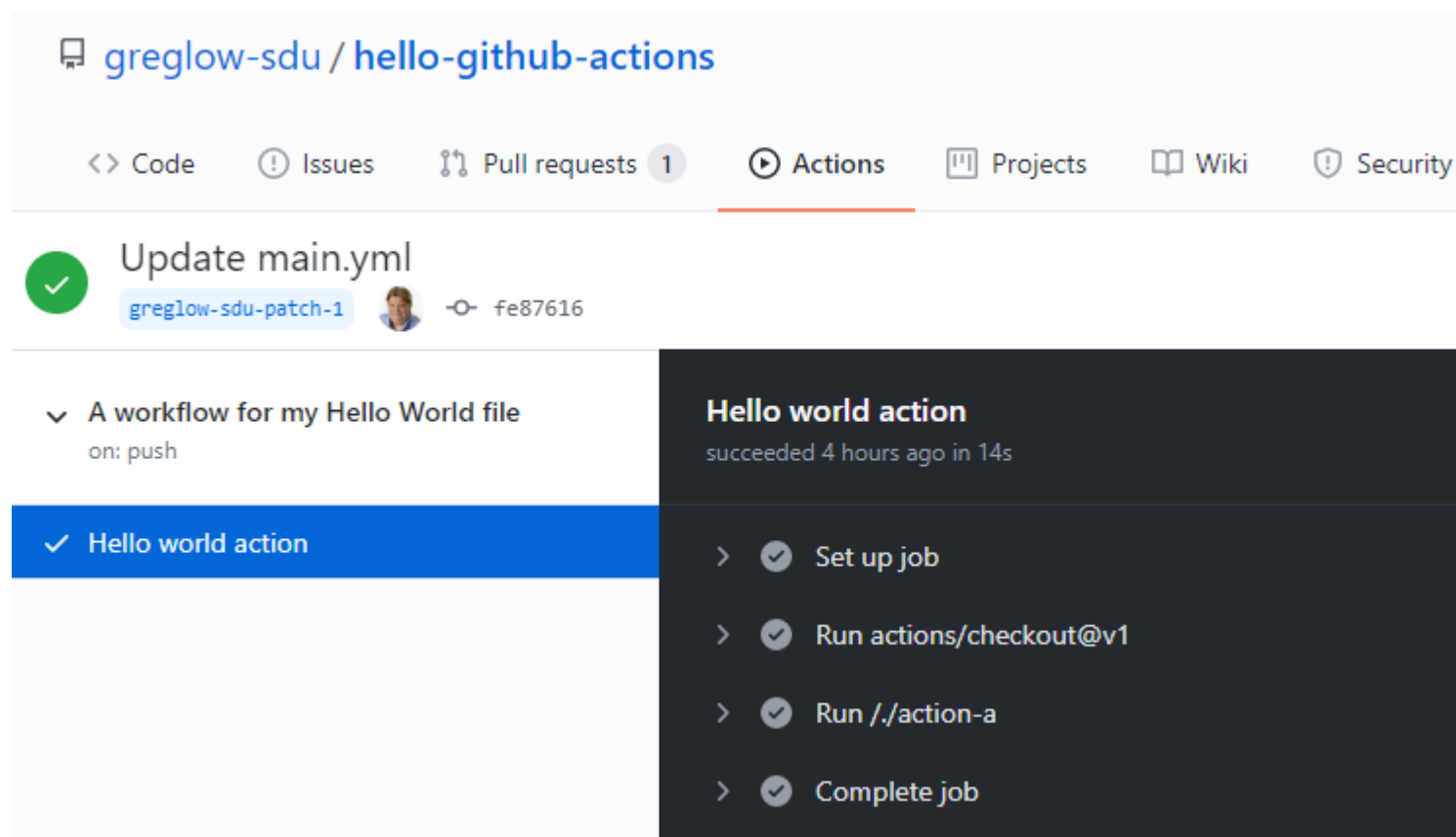
For other languages or options:

Docker containers on Linux

Self-hosted – but do not use for public repos

Console output from actions

Console output from actions is available directly in the GitHub UI



The screenshot shows the GitHub Actions interface for the repository `greglow-sdu / hello-github-actions`. The `Actions` tab is selected, showing a workflow named `Update main.yml` with a green checkmark icon. Below the workflow name, it says `greglow-sdu-patch-1` and `fe87616`. The workflow is triggered by `on: push`. The workflow steps are listed in a sidebar, with `Hello world action` selected. The console output for `Hello world action` is displayed on the right, showing four steps: `Set up job`, `Run actions/checkout@v1`, `Run ./action-a`, and `Complete job`, all with green checkmarks indicating success. The overall status of the workflow is `succeeded 4 hours ago in 14s`.

greglow-sdu / hello-github-actions

<> Code ⓘ Issues 🔗 Pull requests 1 🎬 Actions 📁 Projects 📖 Wiki 🛡 Security

✓ Update main.yml
greglow-sdu-patch-1 fe87616

✓ A workflow for my Hello World file
on: push

✓ Hello world action

Hello world action
succeeded 4 hours ago in 14s

- > ✓ Set up job
- > ✓ Run actions/checkout@v1
- > ✓ Run ./action-a
- > ✓ Complete job

Release management for actions

Specific releases of actions can be requested:

- Using tags
- Using branches
- Using SHA hashes

```
steps:  
  - uses: actions/install-timer@v2.0.1
```

```
steps:  
  - uses: actions/install-timer@develop
```

```
steps:  
  - uses: actions/install-timer@327239021f7cc39fe7327647b213799853a9e
```

Testing an action

DEMO (you can do later)

az-p-hello-github-actions

Lesson 03: Continuous integration with GitHub Actions



Continuous integration with actions

Example workflow shown:

- executes when code is pushed
- runs on latest ubuntu
- uses Node.js version 10

Steps:

- check out the code
- set up dotnet
- build the project

```
name: dotnet Build

on: [push]

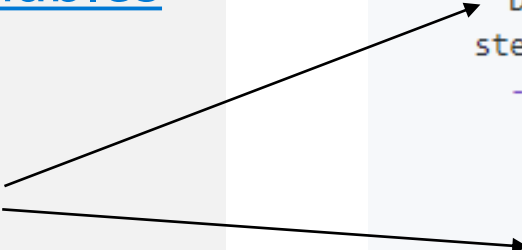
jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node-version: [10.x]
    steps:
      - uses: actions/checkout@main
      - uses: actions/setup-dotnet@v1
        with:
          dotnet-version: '3.1.x'
      - run: dotnet build awesomeproject
```

Environment variables

CI workflows usually need environment variables

- [GitHub default variables](#) (GITHUB_ prefix)
- Custom variables

```
jobs:
  weekday_job:
    runs-on: ubuntu-latest
    env:
      DAY_OF_WEEK: Mon
    steps:
      - name: "Hello world when it's Monday"
        if: ${{ env.DAY_OF_WEEK == 'Mon' }}
        run: echo "Hello $FIRST_NAME $middle_name $Last_Name, today is Monday!"
    env:
      FIRST_NAME: Mona
      middle_name: The
      Last_Name: Octocat
```



Passing artifacts between jobs

CI workflows need to be able to pass artifacts between jobs. Standard actions:

- upload-artifact
- download-artifact

Can configure retention

```
- uses: actions/upload-artifact
  with:
    name: harness-build-log
    path: bin/output/logs/harness.log
```

specific file

```
- uses: actions/upload-artifact
  with:
    name: harness-build-logs
    path: bin/output/logs/
```

entire folder

```
- uses: actions/upload-artifact
  with:
    name: harness-build-logs
    path: bin/output/logs/harness[ab]?/*
```

wildcards

```
- uses: actions/upload-artifact
  with:
    name: harness-build-logs
    path: |
      bin/output/logs/harness.log
      bin/output/logs/harnessbuild.txt
```

multiple paths

Workflow badges

Show status of a workflow within a repository

Typically added to README.md

Can be branch specific

Added via URLs

[https://github.com/**accountname**/**repositoryname**/workflows/**workflowname**/badge.svg](https://github.com/accountname/repositoryname/workflows/workflowname/badge.svg)

Spaces in names must be encoded as %20



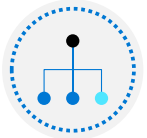
Best practices for creating actions (#mw. i.e. you [create an action](#) that other people can use)



Create chainable actions – avoid monolithic actions



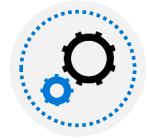
Version your actions like other code



Provide a **latest** label



Add appropriate documentation: like README.md



Add detailed **action.yml** [metadata](#)



Consider contributing to the [marketplace](#)

Marking releases with **Git tags**

Releases are based on **Git tags**

Tags mark a specific point in repository history

Tags are viewed in the repository history

Releases

Tags

v1.1.0

@

 Target: master ▼

Excellent! This tag will be created from the target when you publish this release.

First update after initial release

Releases

Tags

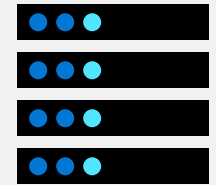
 Tags

v1.1.0

...

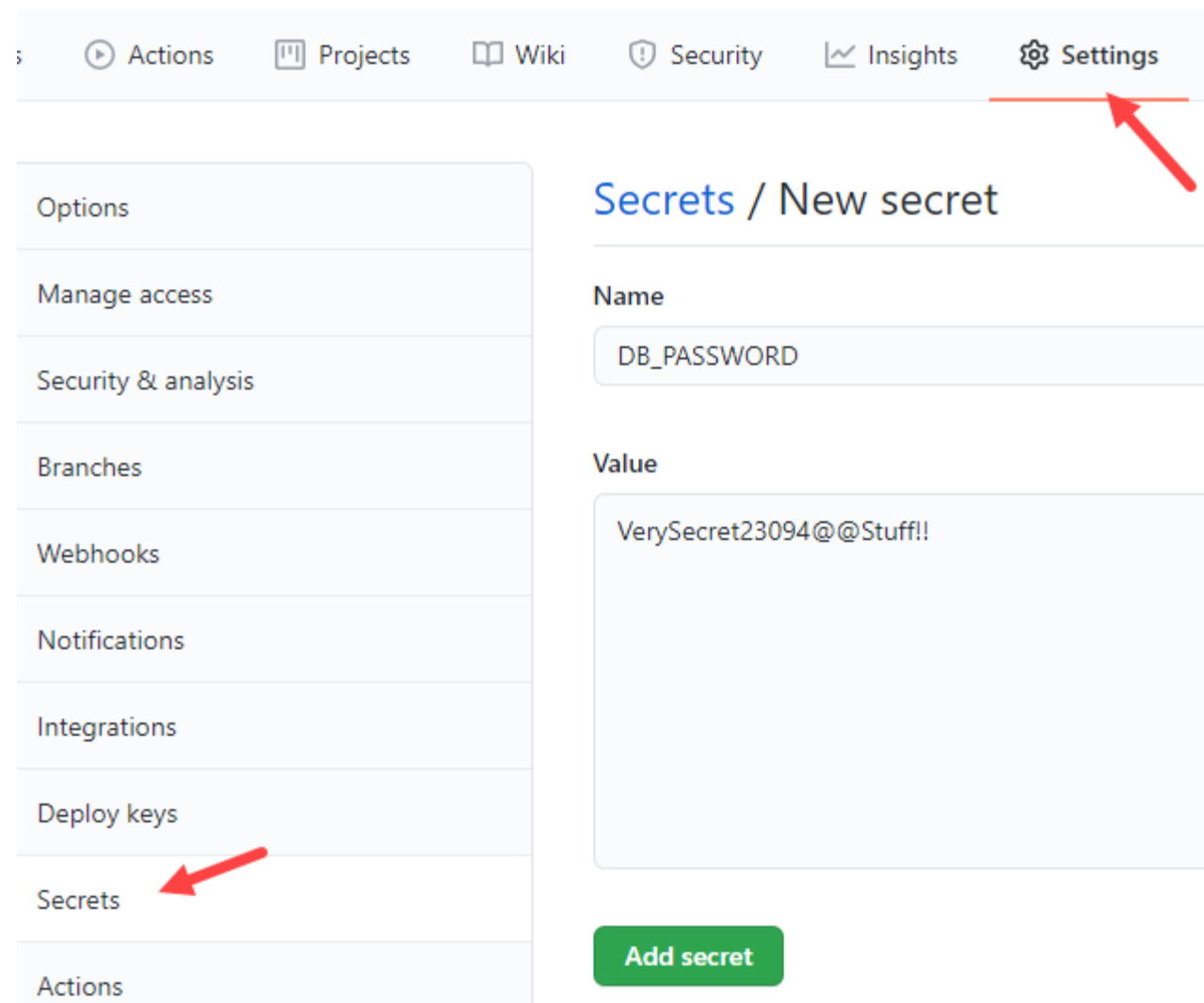
 on 2 Oct  6e20fbe  zip  tar.gz

Lesson 04: Securing secrets for GitHub Actions



Creating encrypted secrets

- Like environment variable but encrypted
- Created at organization, repository, or repository environment level
- Created/assigned in the GitHub UI



The screenshot shows the GitHub Settings page for an organization. The top navigation bar includes links for Actions, Projects, Wiki, Security, Insights, and Settings. The Settings page has a left sidebar with various configuration options, and the main content area shows the 'Secrets' section with a 'New secret' form.

Navigation links: Actions, Projects, Wiki, Security, Insights, **Settings**

Left sidebar options: Options, Manage access, Security & analysis, Branches, Webhooks, Notifications, Integrations, Deploy keys, **Secrets**, Actions

Secrets / New secret

Name: DB_PASSWORD

Value: VerySecret23094@@Stuff!!

Add secret

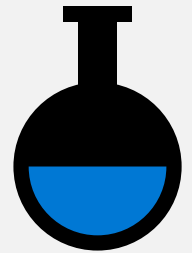
Using secrets in a workflow

- Secrets not automatically passed to runners
- Can be passed as inputs or as environment variables
- Avoid passing secrets in command-line arguments

```
steps:  
  - name: Test Database Connectivity  
    with:  
      db_username: ${ secrets.DBUserName }  
      db_password: ${ secrets.DBPassword }
```

```
steps:  
  - shell: pwsh  
    env:  
      DB_PASSWORD: ${ secrets.DBPassword }  
    run: |  
      db_test "$env:DB_PASSWORD"
```

Lesson 05: Lab



Implementing GitHub Actions by using DevOps Starter



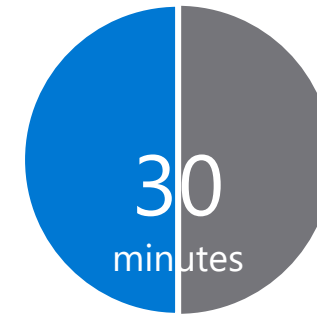
Lab overview:

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

Objectives:

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

Duration:



Lesson 06: Module review and takeaways



What did you learn?



Create and work with GitHub Actions and workflows



Implement Continuous Integration with GitHub Actions

Module review questions

1 True or False: Self-hosted runners should be used with public repos.

2 Database passwords that are needed in a CI pipeline should be stored where?

3 The metadata for an action is held in which file?

4 How can the status of a workflow be shown in a repository ?