

# GitHub Actions: Workflows in Your Repo

Brian Jablonsky

 [brianjablonsky.com](https://brianjablonsky.com)

 [@brian\\_jablonsky](https://twitter.com/brian_jablonsky)

 [bjablonsky](https://github.com/bjablonsky)

 [linkedin.com/in/brianjablonsky](https://linkedin.com/in/brianjablonsky)

# About Me

- Brian Jablonsky
- [@brian\\_jablonsky](#)
- <https://github.com/bjablonsky>
- Microsoft MVP
- Co-organizer of NYC .NET Dev User Group
- .NET Developer
- 10+ years of professional experience



# Overview

- What are GitHub Actions?
- How GitHub Actions Work
- Demo: Creating a simple CI/CD Pipeline
- Going Beyond a Simple Workflow

# What are GitHub Actions?

- An easy way to automate your software workflows straight from GitHub and your repo
- Actions are individual tasks that you can combine to create jobs and customize your workflow
- Create your own Actions or use and customize community created Actions
- Workflows are code stored in your GitHub repo

# Just Another CI/CD Tool?

- While great for CI/CD, it can do a lot more
- Supports a variety of workflow events:
  - Push/Pull request/PR comments
  - Create/delete a branch or tag
  - Deployment
  - Fork
  - Gollum (creates or updates a Wiki page)
  - Issues
  - Releases/Milestones
  - Schedules
  - And more!

# Should I Migrate to GitHub Actions?

- You don't need to migrate your existing CI/CD workflow to GitHub Actions
- You can use GitHub Actions in conjunction with your existing CI/CD and workflow tools

# How GitHub Actions Work

- Build Action(s)
- Create a Workflow based off Event(s)
  - Workflows are made up of Job(s)
  - Jobs are made up of Action(s)
- Execute the Workflow in Runner(s)

# How GitHub Actions Work

- Actions
  - Individual tasks that you can combine as steps to create jobs and workflows
  - Use your own actions or use and customize actions from the community
  - To use in a workflow, you must include it as step



# How GitHub Actions Work

- Jobs
  - A set of steps that execute on the same runner
  - Can define dependency rules for how jobs run in the Workflow
  - Jobs can run in parallel or sequentially

# How GitHub Actions Work

- Workflows
  - A configurable automated process set up in your repo
  - Workflows are made up of one or more jobs
  - Can be scheduled or activated by an event
  - YAML file that defines your workflow and lives in the root of your GitHub repo in the `.github/workflows` directory

# How GitHub Actions Work

- Runners
  - Any machine with the GitHub Actions runner application installed
  - Can be hosted by GitHub or host your own
  - Runners wait for an available job, then executes job's actions and reports the progress/logs/results back to GitHub
  - Runners can run one job at a time
  - GitHub hosted runner:
    - 2-core CPU, 7GB RAM, 14GB SSD
    - Windows supported versions: Windows Server 2019
    - Linux supported versions: Ubuntu 20.04, Ubuntu 18.04, Ubuntu 16.04
    - macOS support versions: macOS Catalina 10.15

# GitHub Actions Limits

- Job execution time: <6 hours (does not apply to self-hosted runners)
- Workflow run time: <72 hours
- Job queue time: <24 hours (does not apply to self-hosted runners)
- API requests: <1000 requests
- Concurrent jobs: (does not apply to self-hosted runners)
  - Free plan: 20 total concurrent jobs, max of 5 macOS jobs
  - Pro plan: 40 total concurrent jobs, max of 5 macOS jobs
  - Team plan: 60 total concurrent jobs, max of 5 macOS jobs
  - Enterprise plan: 180 total concurrent jobs, max of 50 macOS jobs
- Job matrix: <256 jobs per workflow run

# GitHub Actions Pricing

- Public repositories: Free
- OS runner minute multiplier
  - Linux: 1
  - macOS: 10
  - Windows: 2
- Private repositories:
  - Free: 2,000 minutes included/per month/per account
  - Pro: 3,000 minutes included/per month/per account
  - Team: 3,000 minutes included/per month/per account
  - Enterprise: 50,000 minutes included/per month/per account
- Charged to account that owns the repo
- Overage costs: \$0.008/minute

# GitHub Actions Pricing

- Public repositories: Free
- Private repositories:
  - Free: 500MB included/per account
  - Pro: 1GB included/per account
  - Team: 2GB included/per account
  - Enterprise: 50GB included/per account
- Charged to account that owns the repo
- Overage costs: \$0.25/GB

**Demo**

# Creating a CI/CD Pipeline

- Make sure GitHub Actions is enabled in your repo



# Creating a CI/CD Pipeline

- Create workflow YAML file in `.github/workflows` in the root of your repo

- Name your workflow

```
name: Build and deploy an ASP.NET Core app to Azure Web Apps
```

- Define an event section so GitHub Actions knows when to fire it off

```
on:
```

```
  push:
```

```
    branches: [ master ]
```

# Creating a CI/CD Pipeline

- In your workflow file, build the project
  - Create a jobs section

`jobs:`

# Creating a CI/CD Pipeline

- In your workflow file, build the project
  - Name the job

```
jobs:
```

```
  build-and-deploy:
```

# Creating a CI/CD Pipeline

- In your workflow file, build the project
  - Define what runner to use

```
jobs:
```

```
  build-and-deploy:
```

```
    runs-on: ubuntu-latest
```

# Creating a CI/CD Pipeline

- In your workflow file, build the project
  - Create the steps to checkout and build/test your project

```
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Setup .NET Core
        uses: actions/setup-dotnet@v1
        with:
          dotnet-version: '3.1.x'
      - name: Install dependencies
        run: dotnet restore ./src/GitHubActionsExample.sln
      - name: Build
        run: dotnet build --configuration Release --no-restore ./src/GitHubActionsExample.sln
      - name: Test
        run: dotnet test --no-restore --verbosity normal ./src/GitHubActionsExample.sln
```

# Creating a CI/CD Pipeline

- Build!

# Creating a CI/CD Pipeline

- In your workflow file, publish the project
  - Create the steps to publish your project

```
jobs:  
  build-and-deploy:  
    runs-on: ubuntu-latest  
    steps:  
    ...  
    - name: Publish  
      run: dotnet publish -c Release -o './myapp' ./src/GitHubActionsExample.sln
```

# Creating a CI/CD Pipeline

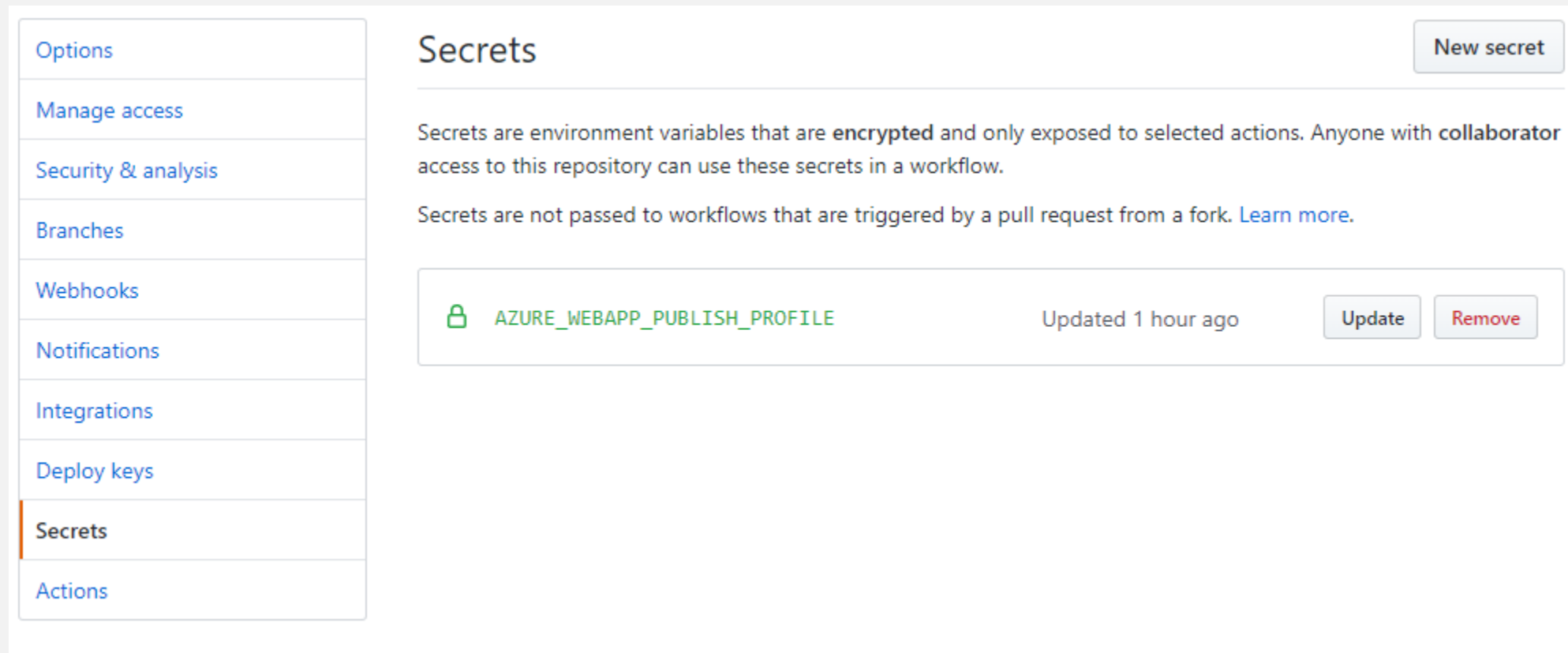
- In your workflow file, deploy the project to Azure Web Apps
  - Create the steps to deploy your project

```
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
    ...
    - name: Publish
      run: dotnet publish -c Release -o './myapp' ./src/GitHubActionsExample.sln
    - name: Run Azure webapp deploy action using publish profile credentials
      uses: azure/webapps-deploy@v2
      with:
        app-name: 'GithubActionsTest-NYC'
        publish-profile: '${{ secrets.AZURE_WEBAPP_PUBLISH_PROFILE }}'
        package: './myapp'
```




# Creating a CI/CD Pipeline

- In your GitHub repo, add a secret for the publish profile



The screenshot displays the GitHub interface for managing repository secrets. On the left, a sidebar contains navigation links: Options, Manage access, Security & analysis, Branches, Webhooks, Notifications, Integrations, Deploy keys, Secrets (highlighted with an orange bar), and Actions. The main content area is titled 'Secrets' and includes a 'New secret' button in the top right corner. Below the title, explanatory text states: 'Secrets are environment variables that are encrypted and only exposed to selected actions. Anyone with collaborator access to this repository can use these secrets in a workflow.' and 'Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more.](#)'. A table below lists the existing secrets:

Secret Name	Last Updated	Actions
 AZURE_WEBAPP_PUBLISH_PROFILE	Updated 1 hour ago	<button>Update</button> <button>Remove</button>

# Creating CI/CD Pipeline

- Deploy!

# Going Beyond a Simple Workflow

- Utilize environment variables for configurable workflows
- Use build matrixes to test across multiple systems/platforms/languages
- Create complex workflows that utilize existing workflows and jobs

# Going Beyond a Simple Workflow

- Create workflow templates
- Cache dependencies and store artifacts to make your workflow run more efficient
- Host your own runners

# Going Beyond a Simple Workflow

- Build your own Actions
  - Currently supports Docker and JavaScript
  - Docker containers run on Linux
  - JavaScript can run on Linux, macOS, and Windows

# Additional Information

- Link to slides and source code
  - <https://github.com/bjablonsky/GithubActions>
- GitHub Actions Docs
  - <https://help.github.com/en/actions>
- GitHub Actions Learning Lab
  - <https://lab.github.com/githubtraining/github-actions:-hello-world>