GitHub Actions: Workflows in Your Repo

Brian Jablonsky

brianjablonsky.com

@brian_jablonsky

bjablonsky

linkedin.com/in/brianjablonsky

About Me

- Brian Jablonsky
- 10+ years of professional experience
- .NET Developer
- Microsoft MVP
- Co-organizer of NYC .NET Dev User Group
- @brian_jablonsky
- https://github.com/bjablonsky

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
- Event-driven
- GitHub Actions are workflows that contain one or more jobs
- Can be scheduled or triggered by an event
- Create your own workflows or use and customize community created Actions and workflows
- Workflows are stored as code 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:
  - Pull request
  - Push
  - Create or delete a branch or tag
  - Deployment
  - Create or update a Wiki page (Gollum)
  - Issues
  - 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

- Create a workflow
- Execute the Workflow in a Runner based off an event

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

- 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

- Steps
  - An individual task that can run commands (called actions)
  - Each step of the job is executed on the same runner to allow

How GitHub Actions Work

- Actions
  - Standalone commands
  - Create your own actions or use and customize actions from the community

How GitHub Actions Work

- Runners
  - Any machine with the GitHub Actions runner application installed
  - Can be hosted by GitHub or can host your own
  - Runners wait for an available job, then executes a job's actions and reports the progress/logs/results back to GitHub
  - Runners can run one job at a time

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 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 Pricing

- Public repositories: Free
- 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
- OS runner minute multiplier
  - Linux: 1
  - macOS: 10
  - Windows: 2
- 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

Creating a CI/CD Pipeline

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

Creating a CI/CD Pipeline

- In your workflow file:
  - Name your workflow

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

Creating a CI/CD Pipeline

- In your workflow file:
  - Define an event section so GitHub Actions knows when to fire it off

```
name: Build and deploy an ASP.NET Core app to Azure Web Apps
on:
  push:
    branches: [ master ]
```

Creating a CI/CD Pipeline

- In your workflow file:
  - Create a jobs section and name the job

```
name: Build and deploy an ASP.NET Core app to Azure Web Apps
on:
  push:
    branches: [ master ]
jobs:
  build-and-deploy:
```

Creating a CI/CD Pipeline

- In your workflow file:
  - Define what runner to use

```
name: Build and deploy an ASP.NET Core app to Azure Web Apps
on:
  push:
    branches: [ master ]
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
```

Creating a CI/CD Pipeline

- In your workflow file:
  - 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:
  - 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
```

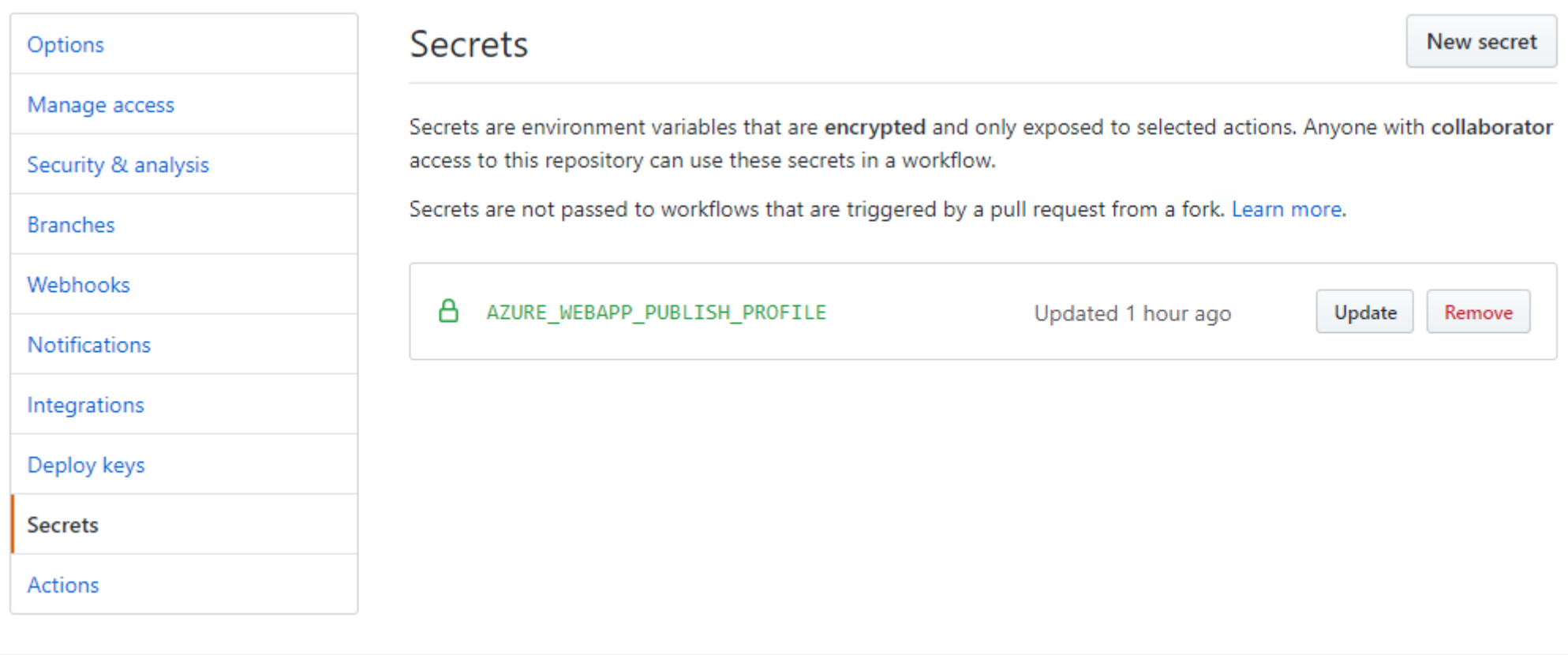
Creating a CI/CD Pipeline

- In your workflow file:
  - 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



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