GitHub Actions

Dr. Isaac Griffith

Idaho State University

Outcomes

After today's lecture you will be able to:

- Understand the basics of applying CI/CD fundamentals using GitHub Actions
- Describe and identify the components of GitHub Actions
- Execute a basic Gradle build using GitHub Actions

ROAR

GitHub Actions Basics

CS 3321

https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

GitHub Actions Components

- There are several components used
  - Workflows
  - Events
  - Jobs
  - Actions
  - Runners

Workflows

- **Workflow**: a configurable automated process that runs one or more jobs
  - defined using a YAML file checked into your repo
  - triggered by
    - an event in your repo
    - manually
    - at a defined schedule
- A repo may have multiple workflows each for a different set of steps and events
  - Additionally you can reuse workflows

Events

- A specific activity which triggers a workflow run
- Examples include:
  - pushing a commit
  - pull request
  - opening an issue
- A Full list can be found:
  - https://docs.github.com/en/actions/learn-github-actions/events-that-trigger-workflows

Jobs

- A set of **steps** in a workflow executing on the same runner
- Can be run as either
  - A shell script
  - An action
- Jobs will run in parallel, unless dependencies between jobs are defined

Actions

- **Action**: a custom application for GitHub actions that performs a complex but repeated task
  - These allow you to:
    - pull a repo from GitHub
    - setup a toolchain for the build
    - setup authentication with a cloud provider
    - …
  - You can define your own actions if you wish

Runners

- **Runner**: a server that executes workflows when triggered
  - Can run a single job at a time
  - GitHub provides the following types of runners
    - Ubuntu (*default*)
    - Windows
    - MacOS
  - If you need different runners or a specific hardware config you can host your own

Example Workflow

- All workflows are stored as YAML files in the `.github/workflows/` directory of your repo
  - workflows are separated into individual files
  - file naming requirements:
    - contains no spaces
    - ends in ".yml"

.github/workflows/learn-github-actions.yml

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
```

Essential Features

CS 3321

https://docs.github.com/en/actions/learn-github-actions/essential-features-of-github-actions

Workflow Variables

- GitHub Actions include default environment variables
- If you need your won, you can set them in the YAML workflow file
  - using and `env` section where each component is a pair
    - `VAR: value`

Example

```yaml
jobs:
  example-job:
    steps:
      - name: Connect to PostgreSQL
        run: node client.js
        env:
          POSTGRES_HOST: postgres
          POSTGRES_PORT: 5432
```

Running Additional Scripts

- You can use actions to execute other scripts or shell commands
- These are assigned to a runner, using the `run` command

Example

```
jobs:
  example-job:
    steps:
      -run: npm install -g bats
```

Example

```
jobs:
  example-job:
    steps:
      - name: Run build script
        run: ./.github/scripts/build.sh
        shell: bash
```

Artifacts

- **Artifacts** provide the ability to store and share data between jobs
- All actions and workflows called within a run have write access to that run's artifacts
- You can download an artifact from a separate workflow run using
  - `actions/download-artifact@v2`

```yaml
jobs:
  example-job:
    name: Save output
    steps:
      - shell: bash
        run: |
          expr 1 + 1 > output.log
      - name: Upload output file
        uses: actions/upload-artifact@v2
        with:
          name: output-log-file
          path: output.log
```

```yaml
jobs:
  example-job:
    steps:
      - name: Download a single artifact
        uses: actions/download-artifact@v2
        with:
          name: output-log-file
```

ROAR

Building/Testing Java with Gradle

CS 3321

https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-gradle

ROAR

Gradle Template

- GitHub provides a basic Gradle workflow template
  - You can choose this template when creating a new workflow
  - You can also manually add it to the `.github/workflows` directory of your repo
- This workflow does the following
  1. `checkout` step downloads a copy of your repo onto the runner
  2. `setup-java` configures the Java 11 JDK by Adoptium
  3. “Validate Gradle wrapper” - validates the wrapper JAR files
  4. “Build with Gradle” - runs `gradlew` to build, test, and package your code

```yaml
name: Java CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK 11
        uses: actions/setup-java@v2
        with:
          java-version: '11'
          distribution: 'adopt'
      - name: Validate Gradle wrapper
        uses: gradle/wrapper-validation-action@
          e6e38bacfdf1a337459f332974bb2327a31aaf4b
      - name: Build with Gradle
        run: ./gradlew build
```

Selecting Your OS

- The template uses `ubuntu-latest` runners.
- If you don’t want to use linux for you can change to a different OS
  - `runs-on: windows-latest` for windows
  - `runs-on: macos-latest` for mac
- Additionally, you can choose to run jobs using Docker containers

Selecting JVM

- You can choose to use a different JVM from a different distribution and for a different architecture:
  - JVM Versions available are based on the specific distribution selected
    - Major versions: `8`, `11`, `16`, `17`
    - Specific Versions: `17.0`, `11.0`, `11.0.4`
    - You can select basically any version since 8
  - Distributions
    - `temurin` from Eclipse
    - `zulu` from Zulu OpenJDK
    - `adopt` from Adopt OpenJDK
    - `liberica` from Liberica JDK
    - `microsoft` from MS Build of OpenJDK
  - Architectures: `x86` or 'x64'

```yaml
steps:
  - uses: actions/checkout@v2
  - uses: actions/setup-jdk@v2
    with:
      distribution: 'adopt'
      java-version: '17'
      check-latest: true
      distribution: 'x64'
```

- For more info: setup-java

Using Multiple JVMs

Idaho State University | Computer Science

- You can also test against multiple Java versions

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        java: ['8', '11', '13', '15', '17']
    name: Java ${{ matrix.Java }} sample
    steps:
      - uses: actions/checkout@v2
      - name: Setup java
        uses: actions/setup-java@v2
        with:
          distribution: '<distribution>'
          java-version: ${{ matrix.java }}
      - run: java -cp java HelloWorldApp
```

ROAR

Building and Testing

- We use the same commands that we use on our local machine to build and test the code
- By default the template will run the `build` task
  - Downloads dependencies
  - Builds classes
  - Runs tests
  - Packages classes into JAR file
- But we can run different commands if we wish
  - We can also have a separate gradle file for CI

```yaml
steps:
  - uses: actions/checkout@v2
  - uses: actions/setup-java@v2
    with:
      java-version: '11'
      distribution: 'adopt'
  - name: Validate Gradle wrapper
    uses: gradle/wrapper-validation-action@
      e6e38bacfdf1a337459f332974bb2327a31aaf4b
  - name: Run the Gradle package task
    run: ./gradlew -b ci.gradle package
```

Packaging Workflow Data

- Often we want to provide the capability to download the constructed artifacts
- You can use the `upload-artifact` action to upload the contents of a directory
- For example to upload the results of packaging

```yaml
steps:
  - uses: actions/checkout@v2
  - uses: actions/setup-java@v2
    with:
      java-version: '17'
      distribution: 'adopt'
  - name: Validate Gradle wrapper
    uses: gradle/wrapper-validation-action@
      e6e38bacfdf1a337459f332974bb2327a31aaf4b
  - run: ./gradlew package
  - uses: actions/upload-artifact@v2
    with:
      name: Package
      path: build/distributions
```

A Full Example

- This example will use gradle to build, test, package, and upload a distribution of your application on a push of a release

```yaml
name: Java Deployment CI
on: [push]
jobs:
  prod-release:
    if: ${{ github.ref == 'refs/heads/main' }}
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK 17
        uses: actions/setup-java@v2
        with:
          java-version: '17'
          distribution: 'adopt'
      - name: Validate Gradle wrapper
        uses: gradle/wrapper-validation-action@
          e6e38bacfdf1a337459f332974bb2327a31aaf4b
      - name: Build with Gradle
        run: ./gradlew package
      - name: Upload artifacts
        with:
          name: Package
          path: build/distributions
```

```yaml
  dev-build:
    if: ${{ github.ref == 'refs/heads/develop' }}
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK 17
        uses: actions/setup-java@v2
        with:
          java-version: '17'
          distribution: 'adopt'
      - name: Validate Gradle wrapper
        uses: gradle/wrapper-validation-action@
          e6e38bacfdf1a337459f332974bb2327a31aaf4b
      - name: Build with Gradle
        run: ./gradlew build
```

For Next Time

- Review the Reading
- Review this Lecture
- Come to Class

- For more info on GitHub Actions
  - See: `https://docs.github.com/en/actions`

ROAR

Are there any questions?

ROAR