Github Actions

This course will contain all Bash Scripts and Yaml and Github actions.

Actions, a developer productivity and CI/CD suite of tools from GitHub, offers developers a new way to automate workflows for their web projects. Each action—which is a set of custom instructions—can be combined with other actions to create a workflow.

You can have tasks that run on a specific timeframe, control what happens when someone interacts with the GitHub repository, and more. In this brief course, learn about the main features of GitHub Actions, as well as the possibilities that it offers you and your organization.

Important Links

- Github actions - Linkedin Learning
- Cron Commands
- Events that trigger workflows
- Github context
- Using Git Secrets to create environment variables
- Medium - Use Github Actions to deploy to Heroku

Github Workflows

- Go into a special folder .git
- You can create 20 workflows
- Will respond for a specific event
- Templates provided by Github
- Y.A.M.L format - JSON format without any pronunciation

How to setup a workflow

- Goto your github repository
- Click on ▶️ **actions**
- Select **Setup my workflow**
- Give your workflow .yml file a name
- Github stores it under ``` `.github/workflows/firstaction.yml ```

```
# This is a basic workflow to help you get started with Actions
name: CI
```

```
# Controls when the action will run. Triggers the workflow on
push or pull request
# events but only for the master branch
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

# A workflow run is made up of one or more jobs that can run
sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest

    # Steps represent a sequence of tasks that will be executed
as part of the job
    steps:
      # Checks-out your repository under $GITHUB_WORKSPACE, so
your job can access it
      - uses: actions/checkout@v2

      # Runs a single command using the runners shell
      - name: Run a one-line script
        run: echo Hello, world!

      # Runs a set of commands using the runners shell
      - name: Run a multi-line script
        run: |
          echo Add other actions to build,
          echo test, and deploy your project.
```

Actions - Trigger Events

There are a different type of events that you can call when writing your Workflow

- Workflow - Called during a **merge** with a targeted branch OR **pull_request** on a targeted branch or a **fork**
    - You can use **Specifiers** to indicate during the status of a workflow
    - i.e., if a PR was successfully closed, or a merge happened
- Webhook - Some action that Github performs - start repo

- Scheduled - Uses cron jobs to schedule an event on a particular date/time.
- Repository Dispatch event - using the keyword `uses`

Workflow Jobs

Although you can only create 20 workflows in an existing repository. You can however write 1 or more jobs in each workflow

- Each job is identified by a Unique Id
- Naming Format: Begin with letter or _. can only contain Alpha-numeric letters
- Jobs run in parallel by default
- To sequentially run Jobs, use the `needs` key. which takes a single id or an array
- `runs-on`: Operating systems that you can specify to run your job.
    - ubuntu-latest
    - ubuntu-18.04
    - ubuntu-16.04
    - windows-latest
    - windows-2019
    - windows-2016
    - macOS-latest
    - macOS-10.14
- `env`: Place where you can define variables per Job.
- `steps`: sequence of tasks that a Job performs
    - Each Job can have multiples `steps`
    - Each step can have multiple `name` & `run` key value pairs
    - Each step can have single `uses` keyword

Variables, Contexts & Expressions

Variables

Variables can be defined using the `env` keyword

- You can have Global variables
- As well as Job specific variables

```
# This is a basic workflow to help you get started with Actions
name: CI
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]
# This is a GLOBAL variable called app_name
env:
    app_name: "B.O.R.E.D"
jobs:
  # This workflow contains a single job called "build"
  build:
    runs-on: ubuntu-latest
    # This variable is specific to build job
    env:
        version: "1.0.0"
    # Steps represent a sequence of tasks that will be executed
as part of the job
    steps:
      # Checks-out your repository under $GITHUB_WORKSPACE, so
your job can access it
      - uses: actions/checkout@v2

      # Runs multiple commands using the runners shell
      - name: Run a one-line script
        run: |
        echo ${{app_name}} is the name of the app!
        echo ${{version}} is specific to the Job - build!
```

Context

These are GLOBAL OBJECT which contain information. If you need info on repository, job, application, step...etc., use one of these objects

- Learn all about different Context Objects here

CONTEXT OBJECT	DESCRIPTION
github	The top-level context available during any job or step in a workflow.
github.actor	The login of the user that initiated the workflow run.
github.event_name	The name of the event that triggered the workflow run.
github.job	The `job_id` of the current job.
job.services..id	The id of the service container.
job.status	The current status of the job. Possible values are `success`, `failure`, or `cancelled`.
steps	This context changes for each step in a job. You can access this context from any step in a job.

Expressions

You can use `if` clauses in your Job tasks and trigger events conditionally

```
steps:
  - uses: actions/hello-world-javascript-action@v1.1
    if: ${{ <expression> }} echo ${{app_name}} is the name of the
app!
```

Secrets - Encrypted Variables in Github

Secrets are encrypted GLOBAL variables in a particular Github repository. They are perfect for holding environment variables which contain sensitive information like DB username & password
You can store any of the following by creating a secret.

- Encrypted
- Ready only
- can be accessed just like variables - `${{ secrets.my_secret_password }}`

Pros

- One really plus point about Secrets is that they also DONT show up even during the logging of a Workflow run
- Ideal for Storing sensitive information

Cons

- Once created, they are encrypted and dont show up. So if you tend to forget the variable you need to delete and create a new one.

The following Job (.yml script) stores a secret (encrypted variable) and create an environment variable

```
- name: Checkout the repository
  uses: actions/checkout@v2

- name: Create .env file
  run: |
    touch .env
    echo DB_URI=${{ secrets.DB_URI }} >> .env

- name: Delete gitignore
  run: |
    rm .gitignore

- name: Push .env to directory
  run: |
    git config user.email "EMAIL"
    git config user.name "USERNAME"
    git add .
    git commit -m "environment variables"
```

Using 3rd Party Plugins

You can refer 3rd party plugins (like Deploy to Heroku, AWS) using the `uses` keyword. Here is the code snippet to deploy your github repo to Heroku project

- You need to setup a Secret by name `HEROKU_API_KEY`
- Follow the steps in this link - Medium Article
- this uses 2 external plugins
    - **actions/checkout@v2** - This is a official GITHUB action which checks out repo contents onto a specified server.
    - **akhileshns/heroku-deploy@v3.0.0** - 3rd party plugin for deploying to Heroku

```
name: Deploy
on:
  push:
    branches:
      - master
```

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: akhileshns/heroku-deploy@v3.0.0
        with:
          heroku_api_key: ${{secrets.HEROKU_API_KEY}}
          heroku_app_name: "your_app_name"
          heroku_email: "your e-mail address associated with
Heroku"
          usedocker: true
```