Continuous Deployment for MkDocs with GitHub Actions

June 20, 2020

MkDocs is a great documentation tool, but every time you make an update, there needs to be a deployment process in order to get your updates published to GitHub Pages. To fix this, I decided to learn how to use GitHub actions to automatically update the published site when an update is pushed or merged into the master branch. If you are new to MkDocs, I certainly recommend my previous blog Getting Started with MkDocs

What are GitHub Actions?

GitHub Actions are a CI/CD tool that is tied directly into GitHub's ecosystem. This allows you to set up workflows that can be triggered through many different events that might take place in your repository, such as a merge or push to a specific branch. In this case, we will trigger the generation and deployment of documentation through MkDocs on every push and merge into the master branch. There are also many pre-built actions available here.

The Action

All GitHub Actions are defined using YAML. There are two main parts that needs to be defined, consisting of when the action should run, and then what should occur when the action is triggered. To start out, lets create a `.github` directory in the outermost directory of your repository, followed by a `workflows` directory within the `.github` one. Then create a `main.yml` file, which is where the action will be defined.

When to Run The Action

I am using this action to update my personal portfolio site as well as some other repositories that only hold documentation and GitHub Pages content. Therefore, I want this action to update with any push or pull request into the master branch. We define this with the following:

```
on:
  push:
    branches: [master]
  pull_request:
    branches: [master]
```

Defining The Jobs and Steps

After we decide when we want GitHub to run the action, the next step is to define the jobs and steps that need to occur in each job. For this, we will only have one job, called build. The tasks will be as follows:

1. Checkout the master branch - this uses a built-in action available to all GitHub Actions to clone the repository into the newly spun-up container.
2. Setup Python 3.7 - this also uses a built-in action that is available to all GitHub Actions.
3. Install pip and mkdocs-material - the `run` keyword is used for terminal commands that you wish to use. the `pip install mkdocs-material` also automatically installs mkdocs and any other package dependencies needed.
4. Run `mkdocs gh-deploy` - this builds the files and deploys them to GitHub pages.

The full result can be seen below:

```
name: Build Documentation using MkDocs

# 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]

jobs:
  build:
    name: Build and Deploy Documentation
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Master
        uses: actions/checkout@v2

      - name: Set up Python 3.7
        uses: actions/setup-python@v2
        with:
          python-version: '3.x'

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install mkdocs-material

      - name: Deploy
        run: |
          git pull
          mkdocs gh-deploy
```

Conclusion

After creating the file above in the `.github/workflows` directory, that should be it! Simply commit your changes locally and push to the GitHub repository. You can view the status of the build on the **Actions** tab on your repository's GitHub page.