**Practical Exercise 27 - Setting Up Our Docker Custom Action**

**Exercise Description**

**In this practical exercise, our goal is to explore how to setup a Docker custom action in GitHub Actions.**

The goal of our Docker custom action is to ping a website to check for its availability, and to abstract that behind an easy-to-use reusable action. **This and the following exercises require Python installed locally, just make sure you have any version later than 3.9.x**. Here are the instructions for the exercise:

1. Create the folder .github/actions/docker-ping-url. This folder is where we will host all the files for our Docker custom action.
2. Add the action.yaml file:
   1. Create a file named action.yaml under the folder .github/actions/docker-ping-url.
   2. In the action.yaml file, add the following properties:
      1. A name of Ping URL;
      2. A description of "Ping URL until maximum trials have exceeded. If result is not 200 until then, fails the action.";
      3. The action should receive three inputs:
         1. The first one, named url, should be required, and have a description of URL to ping.
         2. The second one, named max\_trials, should not be required, have a description of Maximum number of trials until action fails, and default to '10'.
         3. The third one, named delay, should not be required, have a description of Delay in seconds between trials, and default to '5'.
      4. Add a top-level runs key. This is the core of defining our Docker custom action. For a Docker custom action, the runs key has the following shape:
         * 1. runs:
           2. using: docker
           3. image: Dockerfile

where:

* + - 1. using: docker defines that the action will be run using Docker.
      2. image: <Dockerfile or image> defines which file will be used to build the Docker image, or which image will be used for execution.

1. Set up the local environment and dependencies for the Python script:
   1. Open a terminal and change into this directory.
   2. Create a Python virtual environment by running python -m venv venv. This will create a new directory named venv within the docker-ping-url folder, where all the dependencies of our Python scripts will be installed. This is recommended over installing dependencies globally.
      1. Add a new line to the root .gitignore file containing venv. This will prevent the venv directory from being committed into the repository.
   3. Activate the virtual environment by running . venv/bin/activate (on Mac and Linux), or the equivalent command on Windows.
   4. Run the command pip install "requests==2.31.0" to install the necessary dependencies for the Python script.
   5. Once the dependencies are installed, run pip freeze > requirements.txt to save a list of dependencies and their respective versions to a file to be used later on by Docker when building the image.
2. Create a main.py file under the folder .github/actions/docker-ping-url and add the following code to the file:
   1. if \_\_name\_\_ == "\_\_main\_\_":
   2. print("Hello world")
3. Add a Dockerfile under the folder .github/actions/docker-ping-url. The Dockerfile should use python:alpine3.19 as the base image, and execute the necessary steps to install the dependencies and run the Python script in main.py. If you are not familiar with Docker and Dockerfile, the code for the Dockerfile is available in the resources section of this lecture.
   1. Additionally, add a .dockerignore file under the folder .github/actions/docker-ping-url. The file should contain a single line containing venv. This is to prevent Docker from copying the venv directory when building the image.
4. Create a file named 17-3-custom-actions-docker.yaml under the .github/workflows folder at the root of your repository.
5. Name the workflow 17 - 3 - Custom Actions - Docker.
6. Add the following triggers with event filters and activity types to your workflow:
   1. workflow\_dispatch: additionally, the workflow\_dispatch trigger should receive one input, named url, of type string and with a default of 'https://www.google.com'
7. Add a single job named ping-url to the workflow.
   1. It should run on ubuntu-latest.
   2. It should contain two steps:
      1. The first step should checkout the code.
      2. The second step, named Ping URL, should use the recently created Docker custom action. To reference a custom action created in the same repository as the workflow, you can simply provide the path of the directory where the action.yaml file is located. In this case, this would be ./.github/actions/docker-ping-url. Make sure to pass the url input to the custom action.
8. Commit the changes and push the code. Trigger the workflow from the UI and take a few moments to inspect the output of the workflow run.

Resources for this lecture

* Code for Dockerfile: <https://github.com/lm-academy/github-actions-course/blob/5106bbc41f7dd67ba662e4093cb7e5d1d184829a/.github/actions/docker-ping-url/Dockerfile>