Building Docker Images

Next step is to have our application packaged as a docker image for easy distribution.

We have some requirements for our pipeline step:

- Should build our application as a docker image.
- Should tag the image with both the git sha and "latest".
- Should push the image to docker registry.

In order for this to work, we need three environment variables:

- `docker_username` the username for docker registry.
- `docker_password` the password for docker registry.
- `GIT_COMMIT` the name of the git commit that is being built.

You can set these environment variables as global variables in your workflow through the `env` section.

```
env:
  docker_username: <your docker username>
  docker_password: <your docker password>
  GIT_COMMIT: <your git commit>
```

The two scripts: `ci/build-docker.sh` and `ci/push-docker.sh` expects all three environment variables to be set.

Build-in environment variables

Many of the common information pices for a build is set in default environment variables.

Examples of these are:

- The name of the repository
- The name of the branch
- The SHA of the commit

You can see the ones you can use directly inside a step here: https://docs.github.com/en/actions/learn-github-actions/environment-variables#default-environment-variables

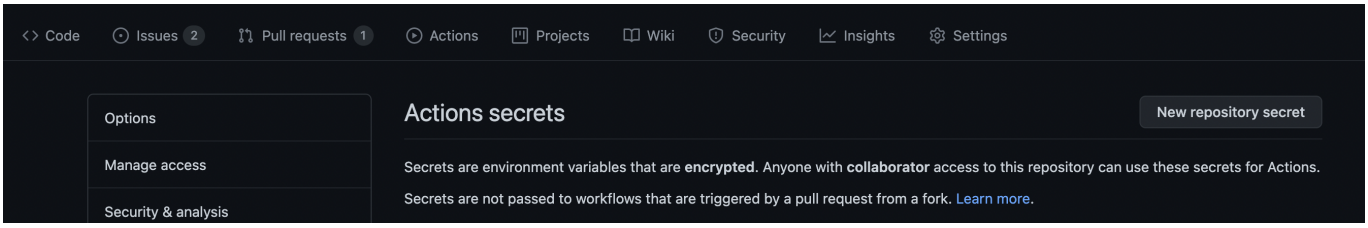
Github Actions also has a list of contexts.

Contexts are a way to access information about workflow runs, runner environments, jobs, and steps. Each context is an object that contains properties, which can be strings or other objects. You can see them here: https://docs.github.com/en/actions/learn-github-actions/contexts#about-contexts

The default environment variables that GitHub sets are available to every step in a workflow. Contexts are also available before the steps, as when defining the `env` section of the workflow.

Tasks

- To start Docker credentials should be stored as repository secrets at Github Actions Repository. Please go to `Settings > Secrets > New repository secret` to add them.



- Add a new job named `Docker-image` that requires the `Build` to be completed.

```
Docker-image:
  runs-on: ubuntu-latest
  needs: [Build]
```

- Add a new step to your `Build` job which uploads the compiled code found in `app/build/libs/app-0.1-all.jar`.

💡 *if you forgot how to do it, head over to storing artifacts*

- Add a step in `Docker-image` which downloads the build.
- Add `docker_username` and `docker_password` as environmental variables on top of the workflow file.

```
env:
  docker_username: ${{ secrets.DOCKER_USERNAME }}
  docker_password: ${{ secrets.DOCKER_PASSWORD }}
```

- Add GIT_COMMIT environment variable as well.

Tip! it needs the same "wrapping" (`${{}}`) as the other environment variables, and can be found in the `github` context

- Run the `ci/build-docker.sh` and `ci/push-docker.sh` scripts.

Ready steps looks like:

```
    - name: build docker
      run: chmod +x ci/build-docker.sh && ci/build-docker.sh
    - name: push docker
      run: chmod +x ci/push-docker.sh && ci/push-docker.sh
```

> Hint: Remember that the job needs to run on specified system and is based on the results from previous jobs.

- See that the image is built and pushed to the docker hub registry.

Using actions instead of scrtipts

The above job can be also done by using actions: `docker/login-action@v1` and `docker/build-push-action@v2`, what will provide the same functionality. You can find it in the example below:

```
on: push
jobs:
  build-and-push-latest:
    runs-on: ubuntu-latest
    steps:
      - name: Login to DockerHub
        uses: docker/login-action@v1
        with:
          username: ${{ env.docker_username }}
          password: ${{ env.docker_password }}
      - name: Build and push
        uses: docker/build-push-action@v2
        with:
          push: true
          tags: $docker_username/micronaut-app:1.0-${GIT_COMMIT::8}
```

Solution

If you strugle and need to see the whole ***Solution*** you can extend the section below.

▶ Solution

```
name: Java CI
on: push
env: # Set the secret as an input
  docker_username: ${{ secrets.DOCKER_USERNAME }}
  docker_password: ${{ secrets.DOCKER_PASSWORD }}
  GIT_COMMIT: ${{ github.sha }}
jobs:
  Clone-down:
    name: Clone down repo
    runs-on: ubuntu-latest
    container: gradle:6-jdk11
    steps:
    - uses: actions/checkout@v2
    - name: Upload Repo
      uses: actions/upload-artifact@v2
      with:
        name: code
        path: .
  Build:
    runs-on: ubuntu-latest
    needs: Clone-down
```

```
      container: gradle:6-jdk11
      steps:
      - name: Download code
        uses: actions/download-artifact@v2
        with:
          name: code
          path: .
      - name: Build with Gradle
        run: chmod +x ci/build-app.sh && ci/build-app.sh
      - name: Test with Gradle
        run: chmod +x ci/unit-test-app.sh && ci/unit-test-app.sh
      - name: Upload Repo
        uses: actions/upload-artifact@v2
        with:
          name: code
          path: .
    Docker-image:
      runs-on: ubuntu-latest
      needs: [Build,Test]
      steps:
      - name: Download code
        uses: actions/download-artifact@v1
        with:
          name: code
          path: .
      - name: build docker
        run: chmod +x ci/build-docker.sh && ci/build-docker.sh
      - name: push docker
        run: chmod +x ci/push-docker.sh && ci/push-docker.sh
```

Results

You should be able to see your docker image on your DockerHub account as:

Explore 〉 paulinadubas/micronaut-app 〉

paulinadubas/micronaut-app ☆

By paulinadubas • Updated 2 months ago

Container