Continuous Integration & Continuous Delivery

Your team has been assigned the task of automating a development project according to CI/CD principles. The project currently uses Python flask and already has some automation ready. However, there are some shortcomings in the project regarding tests, production configuration, operability, observability and security.

The managers have decided certain tools/solutions must be used, e.g.

- Git & GitHub products
- Kubernetes as deployment target

Management is very interested in the progress. They want you to continuously write a work diary during the course of the project in which it is clear who has done what.

ASSIGNMENT G

1. Add the teacher as a member in your `devops_cicd_final` repository.
2. Update the WORK_DIARY.md with the team member's first names

Inner Loop

The goal for the inner loop is to support the developer and help them be a better version of themselves.

The developers should be able to:

- Run Unit tests
- Run integration tests or e2e tests (the team decides)
- Easily install packages with pip for development purposes
- Read clear instruction in a README.md file how to get started
- Be able to run most of the test automation locally
- Build a docker image
- Using pre-commit to run unit test on a push

Outer loop

The goal of the outer loop is to support the team and maintain the quality of its deliverables. But also give quick feedback to the developer if something fails.

The CI/CD should be able to:

- Handle feature branches
    - Changes are merged into main with a Pull Request
    - The main branch is protected from direct push
    - The tests are required to pass before a merge is possible
- Run on every commit in a branch
    - Run linting e.g. pylint or flake8
    - Run unit tests
    - Run coverage calculations
    - If unit test and linting succeed
        - Run Docker build
        - Tag the image with:
            - branch name
            - short git sha
        - Push the image to the GitHub Container Registry
    - Run Integration tests or e2e tests
- Run on every commit in main
    - Run linting
    - Run test automation
    - If test automation succeed
        - Tag the image with:
            - latest
            - Push the image to the GitHub Container Registry

Prove that it works

Testing Inner loop

Create a small change to the flask server in a feature branch.

1. The change includes a new function or class
2. The change includes unit tests
3. The unit tests fails in the inner loop

When you present your work:
- The test should be fixed and you can push the code to GitHub.

Testing Outer loop

Create a small change to the flask server in a feature branch.

1. The change includes a new route with a feature of your choice
2. The change includes unit tests
3. The change includes a integration or e2e test of the new route
4. The integration or e2e tests should fail

When you present your work:

- Fix the code or tests failing
- Open a Pull Request
    - Show that "status checks" has passed
    - Show that a docker image is pushed to the container registry
    - Show the coverage log as a artifact or in the runner terminal log
- Merge the pull request
    - Show that a workflow runs for the main branch
    - Show that a image was uploaded to the Container Registry
- Deploy:
    - Update k8s to run your new image
    - Show in the browser that your image is running with the added feature

Hand in instructions G assignment

- Screenshot of a failing pre-commit test run
- Screenshot of your successful workflow run in GitHub
- Screenshot of your failing workflow run in GitHub
- Screenshot of your pull request with a successful "status check"
- Screenshot of your Github Packages showing your image(s)
- Screenshot of your flask server running in k8s
- Screenshot of your browser showing your FEATURE

All students should take their own screenshots, then compress all files to one zip, or tar.gz and upload it in the student portal. *DEADLINE 1/12*