GitHub, Travis CI

GitHub

- GitHub Classroom is an tool for creating programming exercises or problem sets and distributing them to students enrolled in a course. This will be used as an example porject to showcase a workflow for developing web applications.
    - The first step is to fork the main `classroom` repository and clone it onto a computer.
    - When making improvements and changes to this repository, a good paradigm to follow is the feature-branch development strategy, in which a new branch is made for each feature, bug report, etc., which is merged back into the master branch as soon as itâ€™s done and ready to be released. This is one example of a continuous deployment strategy.
        - A completely different development strategy is released-based development, which is more common in desktop applications or applications that are shipped out. In these applications, developers will build up, for example, the version 1.0 branch, while also working on a 1.1 branch, even though the latter will only be released 6 months later, perhaps.
    - Once a feature has been developed in a separate branch, those changes should be pushed to the personal, forked version of the repository. Then, a pull request should be opened on the main repository. This repository is then sent to Travis as a build to be tested. Any co-developers can also review the pull request, provide feedback, and discuss the changes.
    - Once the pull request is reviewed and accepted, the next step is to actually deploy the new version of the application. Ideally, deployment should be continual. The more time spent building up a separate branch without integration and deploying it, the harder it will be to integrate and deploy it successfully in the future.
        - One way to progressively deploy a large feature is with feature toggles. A feature toggle consists of a break in the code where one of two versions can be chosen based on certain variables. This can be used to deploy new features without impacting all users at once. This could be used, for example, to let a small group of beta testers try out the new feature. This feature can now be continually deployed and developed, and when the time comes, it can be deployed to an increasing number of users. Ultimately, the feature toggle can be removed as the feature becomes fully deployed.

Travis CI

- To review, continuous integration is a development strategy that revolves around continually adding code to a codebase. CI can also refer to a tool that helps facilitate this process. Travis is one such tool.
- A large part of CI is testing (test-driven development). Tests can be either manual or automatic, functional or non-functional. Manual tests involve simply checking features by hand. A more preferred approach is to write automated tests, which are scripts a machine can run to verify an assertion. Functional tests check a specific functionality of an application and ensures that it meets all the expected requirements. From bottom to top, the different levels of testing can be thought of as unit testing, component testing (check whole packages), integration testing (check that packages work together), and end-to-end testing (check the entire flow). Tests should ensure that things both work and fail as they should. Testing should happen as development progresses, not at the end.
- Another part of the CI workflow is the build. A build is any complete, tested version of an application (it may have failed or passed those tests), but build can also be verb that describes the testing process.
- CI systems, as opposed to simply testing on a single machine, help to ensure reproducability and facilitate collaboration. They lead to tidy deploys that have already been tested in an environment as close to the production environment as possible and a faster development flow due to improved confidence in code and pull requests. CI systems allow for automation for many parts of the development process.
- CD is a natural result of CI. Because every commit is tested, every commit is deployable.