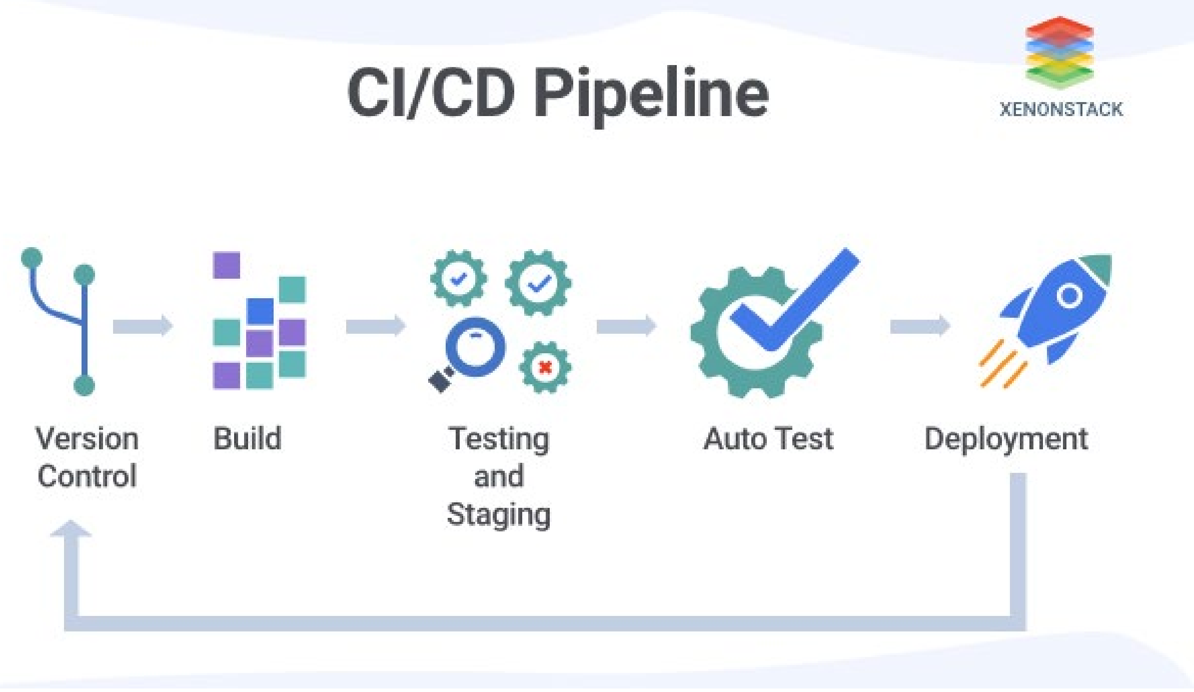
Continuous Integration

ERIN KEITH

CS 333 – TESTING AND DEVOPS



Pipeline

- Coding
  - Code development and reviews
  - Source code management
  - Code merging
  - *not IDEs (personal development environment)
- Building
  - Continuous integration tools
  - Build status
- Testing
  - Continuous testing tools

Building - Continuous Integration

- When code is integrated into a shared repository frequently
- Does it build?

Benefits

- Speeds up workflow
- Better communication and feedback
  - code is more accessible to team

Testing - Continuous Integration

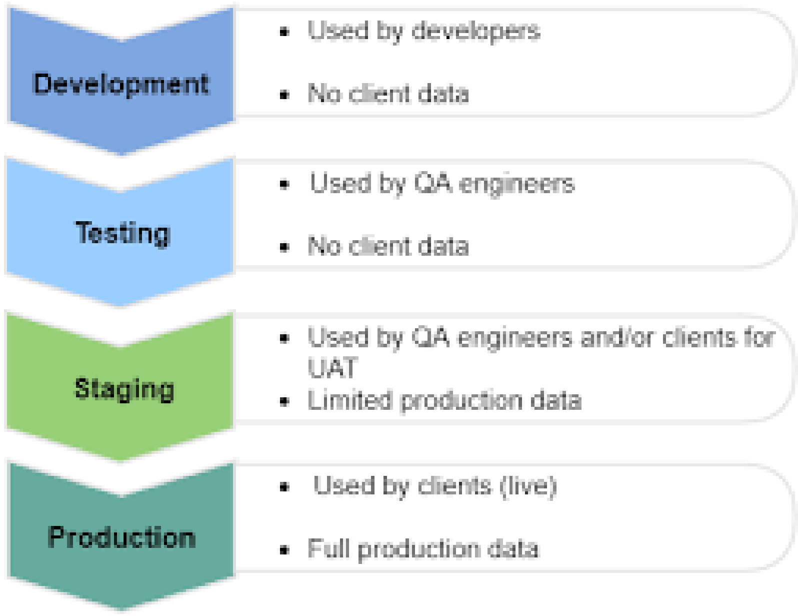
- Does it break tests?
  - All unit and integration tests are run across the project
- Every bug-fix commit should come with *at least* one test case
- Code analysis

Benefits

- Reduces fear of breaking codebase
- Reduces risk
  - tests are run automatically

Testing - Environments

1. Developer creates a feature branch for every new feature.
2. The feature branch is (automatically) deployed on our **TEST environment** with every commit for the developer to test.
3. When the developer is done with deployment and the feature is ready to be tested he merges the develop branch on the feature branch and deploys the feature branch that contains all the latest develop changes on **TEST**.
4. The tester tests on **TEST**. When he is done he "accepts" the story and merges the feature branch on develop. Since the developer had previously merged the develop branch on feature we normally don't expect too many conflicts. However, if that's the case the developer can help. This is a tricky step, I think the best way to avoid it is to keep features as small/specific as possible. Different features have to be eventually merged, one way or another. Of course the size of the team plays a role on this step's complexity.
5. The develop branch is also (automatically) deployed on **TEST**. We have a policy that even though the features branch builds can fail the develop branch should never fail.
6. Once we have reached a feature freeze we create a release from develop. This is automatically deployed on **STAGING**. Extensive end to end tests take place on there before the production deployment. (ok maybe I exaggerate a bit they are not very extensive but I think they should be). Ideally beta testers/colleagues i.e. real users should test there.

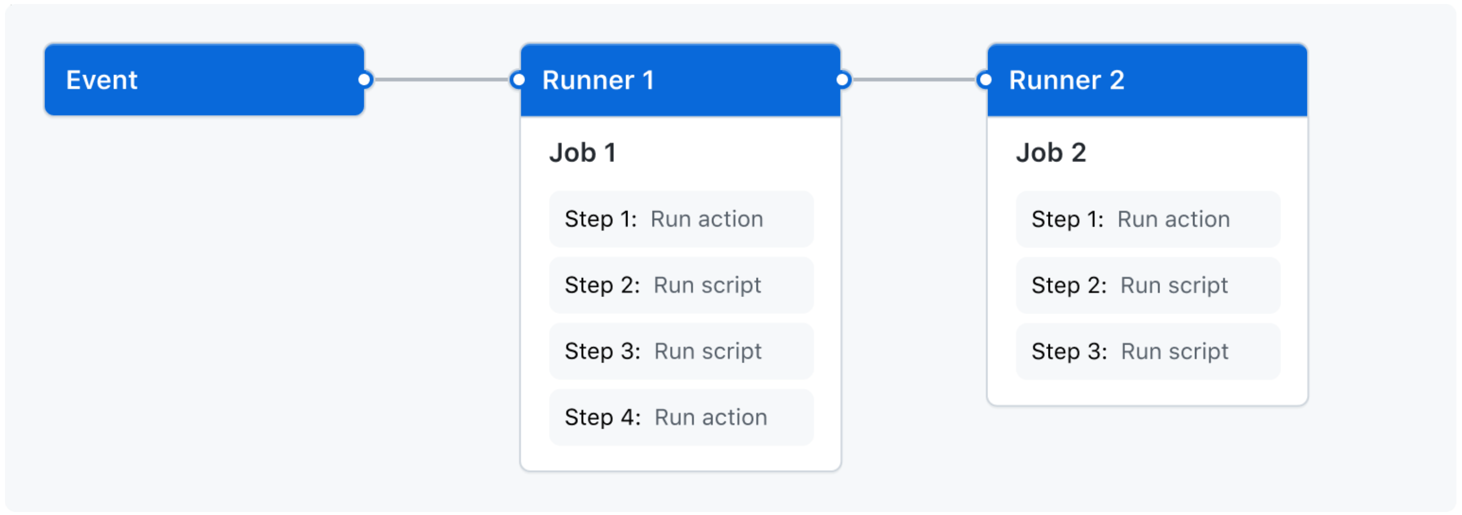


Variations of a Theme

- Continuous Integration
  - Integrating developer changes as soon as possible
  - Check in and test work as tasks are finished
- Continuous Deployment
  - Keeping your application deployable at all times
  - Product is in a complete working state at all times
- Continuous Delivery
  - Codebase is deployable automatically
  - Any additional tools are always ready
    - Installers, containers, etc.

Continuous Integration

- Use “jobs” for different tasks
  - Build
  - Run unit tests
  - Run integration tests
  - Deploy “artifact”



GitHub Actions

Questions?

Let’s Do Something!

Create a workflow to run the existing tests in your Activity 4 repository.

- Use a new branch you create (creating the branch isn't part of the workflow).
- You're done when the workflow finishes with a green checkmark after pushing a change.
- Include the linter, but don't worry about packaging and publishing.

Resources

- You can work through this tutorial as a "hello world" type introduction to get you started:
- This tutorial is more specific for this activity:
  - It walks you through understanding all the different parts of the .yml file so you can update it to do the Activity.