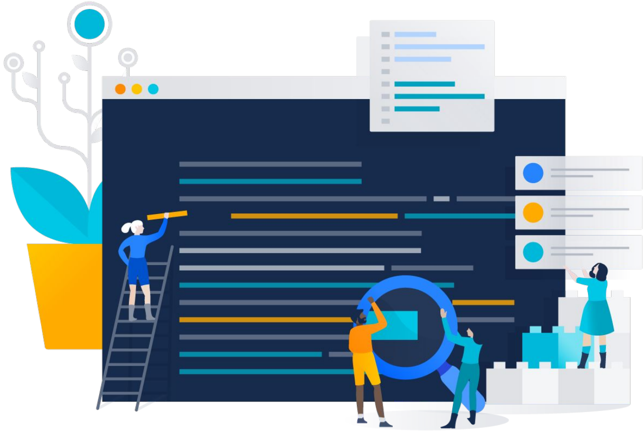
CS130: Software Engineering

Week 5



Agenda

1. Project Part B
2. Advanced Git Commands
3. Build Management
4. CI/CD
5. Software Development Methodologies
6. Team Time

CONGRATS
YOU FINISHED A TON OF WORK
SO YOU CAN WORK SOME MORE
imgflip.com

Part B

Should include:

1. Detailed description of the implemented units

Part B

Should include:

1. Detailed description of the implemented units
2. API design in a Javadoc-like manner

Part B

Should include:

1. Detailed description of the implemented units
2. API design in a Javadoc-like manner
3. Changeability matrix

Part B

Should include:

1. Detailed description of the implemented units
2. API design in a Javadoc-like manner
3. Changeability matrix
4. Screenshots of features that are tested and ready to release

Part B

Should include:

1. Detailed description of the implemented units
2. API design in a Javadoc-like manner
3. Changeability matrix
4. Screenshots of features that are tested and ready to release
5. Test scenarios as executable programs

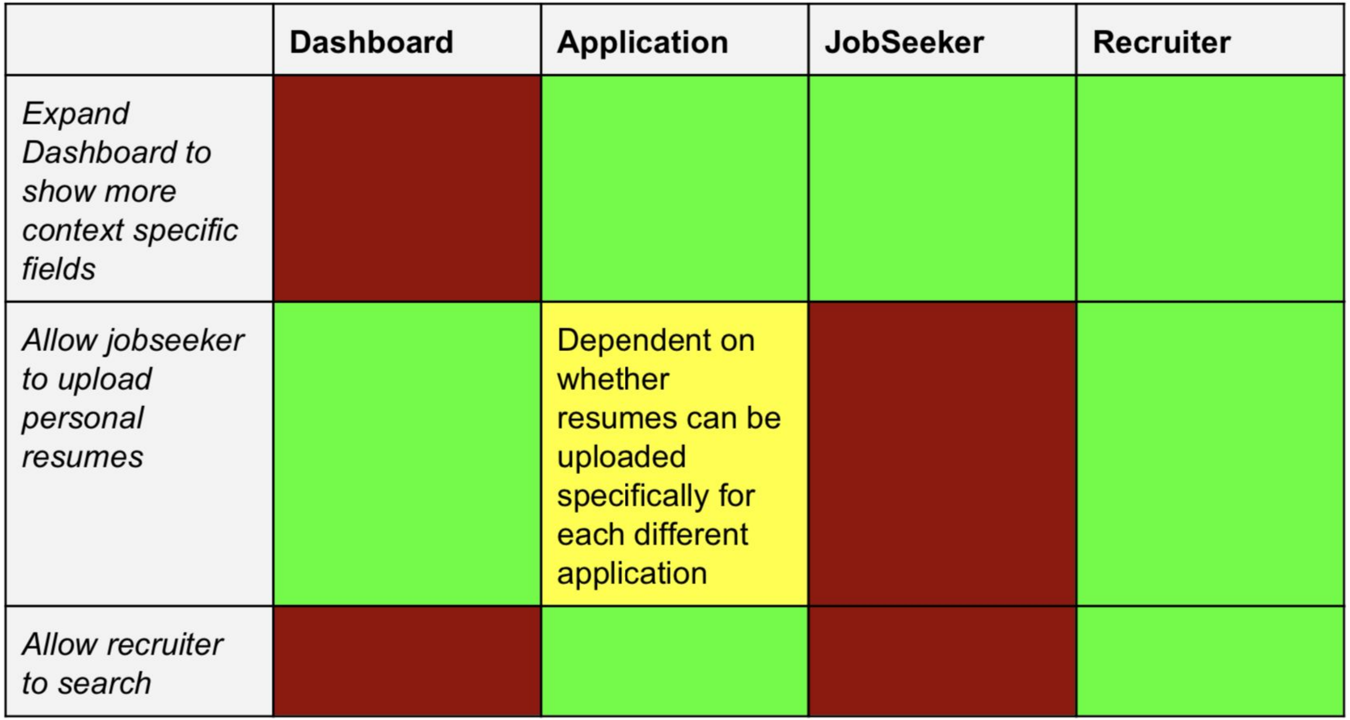
Let’s look at a sample project

Suppose you have four classes: Dashboard, Application, JobSeeker, Recruiter

Here’s how it goes:

1. Each JobSeeker has a dashboard including the jobs they have applied to, the deadlines, and status
2. Each recruiter has a dashboard with job listing and candidates’ information
3. The recruiter receives applications and they can update the job status and application details
4. Application includes the user’s formatted resume, the job description, and relevant skills

Changeability matrix example



	Dashboard	Application	JobSeeker	Recruiter
Expand Dashboard to show more context specific fields	🟥	🟩	🟩	🟩
Allow jobseeker to upload personal resumes	🟩	Dependent on whether resumes can be uploaded specifically for each different application	🟥	🟩
Allow recruiter to search	🟥	🟩	🟥	🟩

Exercise

1. Take a few minutes to discuss with your team-mates some possible change scenarios (related to scaling, adding new features in the future, etc)

2. Once you have these changes, take a look at your class diagram and understand what classes would need to be modified for each scenario

3. Construct a changeability matrix for the above

Git 🔥

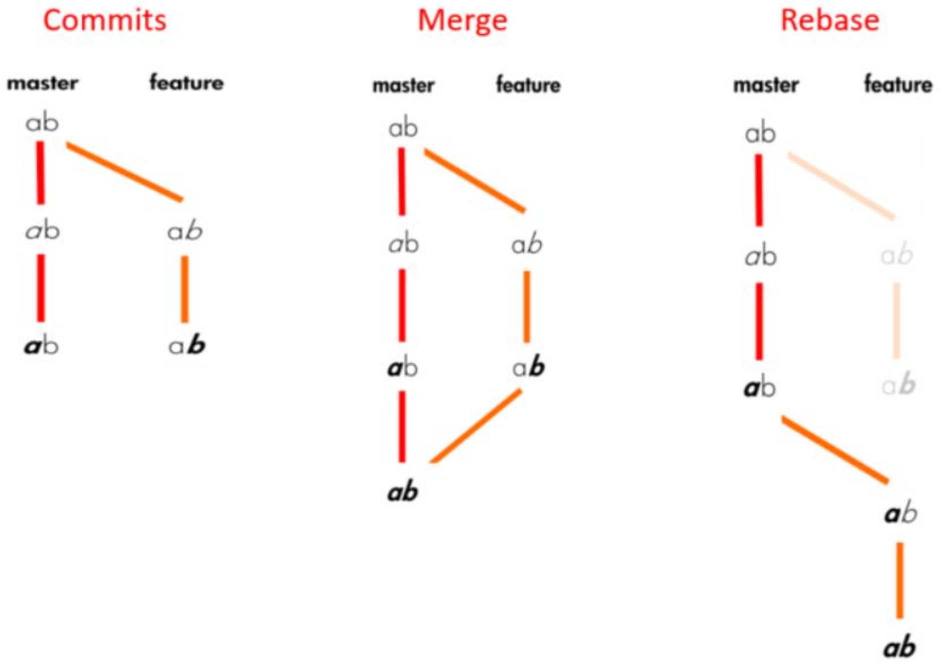
Has merge conflicts
ONLY DEVELOPER WORKING ON PROJECT
quickmeme.com

What’s the difference?

git-merge and git-rebase

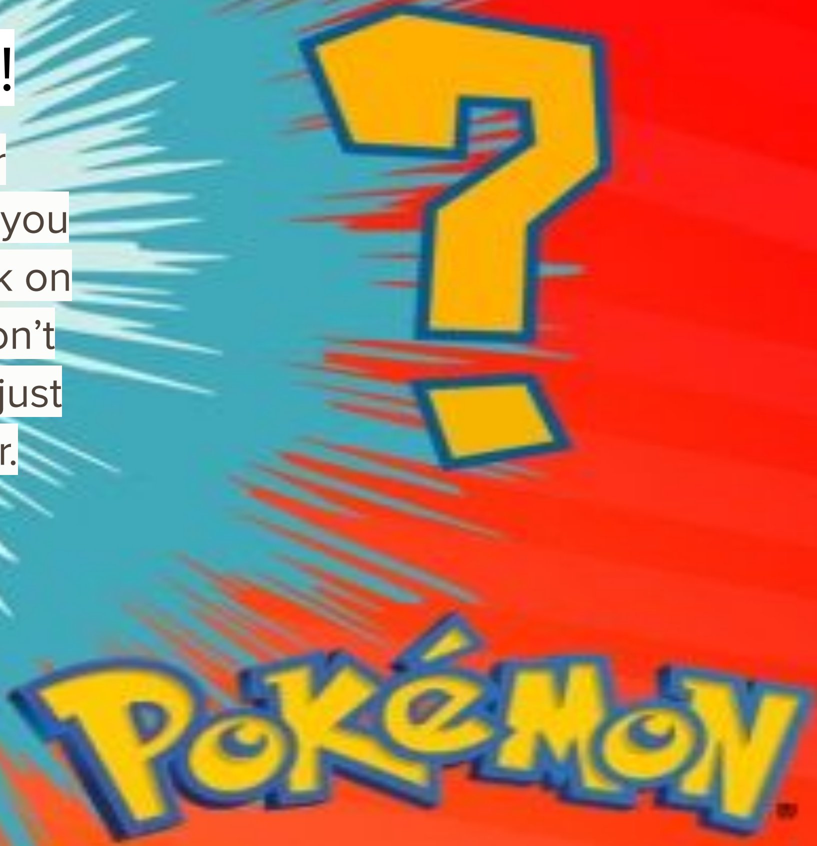
What’s the difference?

git-merge and git-rebase

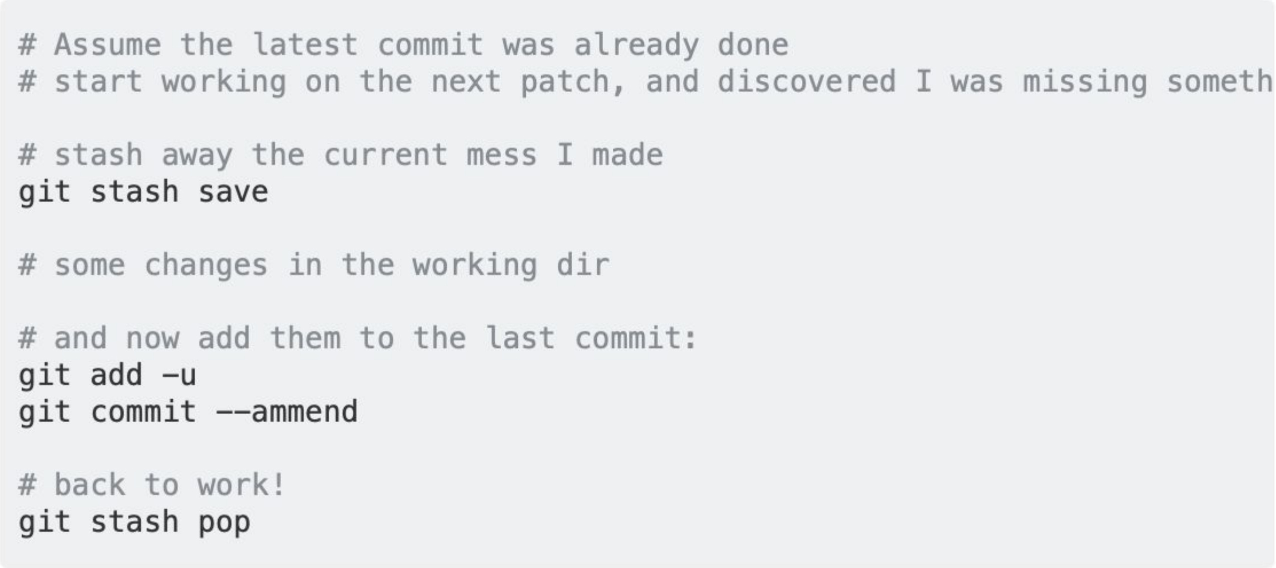


Guess the Git Command!

You've been working on part of your project, things are in a messy state and you want to switch branches for a bit to work on something else. The problem is, you don't want to do a commit of half-done work just so you can get back to this point later.



Ans: git stash

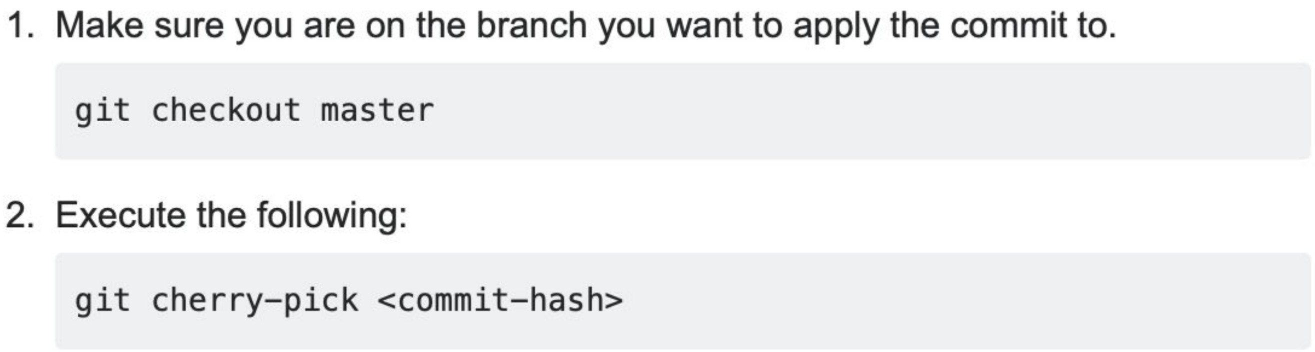


```
# Assume the latest commit was already done
# start working on the next patch, and discovered I was missing someth

# stash away the current mess I made
git stash save

# some changes in the working dir

# and now add them to the last commit:
git add -u
git commit --ammend

# back to work!
git stash pop
```

Guess the Git Command!

Used to choose a commit from one branch and apply it onto another.

Ans: git cherry-pick, used to choose a commit from one branch and apply it onto another.



1. Make sure you are on the branch you want to apply the commit to.

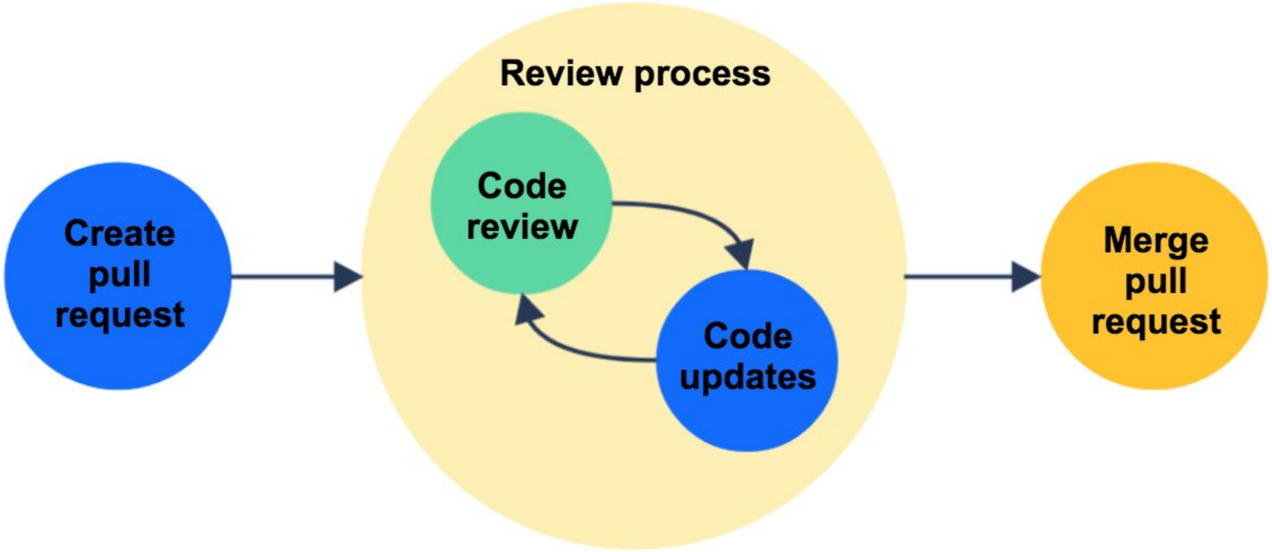
```
git checkout master
```

2. Execute the following:

```
git cherry-pick <commit-hash>
```

when you try to git-blame and you realize you wrote the broken code

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub



Code review

1. What are code reviews? When do they occur in the build management pipeline?

2. Linters - Tools that analyze source code to flag programming errors, bugs, stylistic errors, and suspicious constructs.

Phabricator

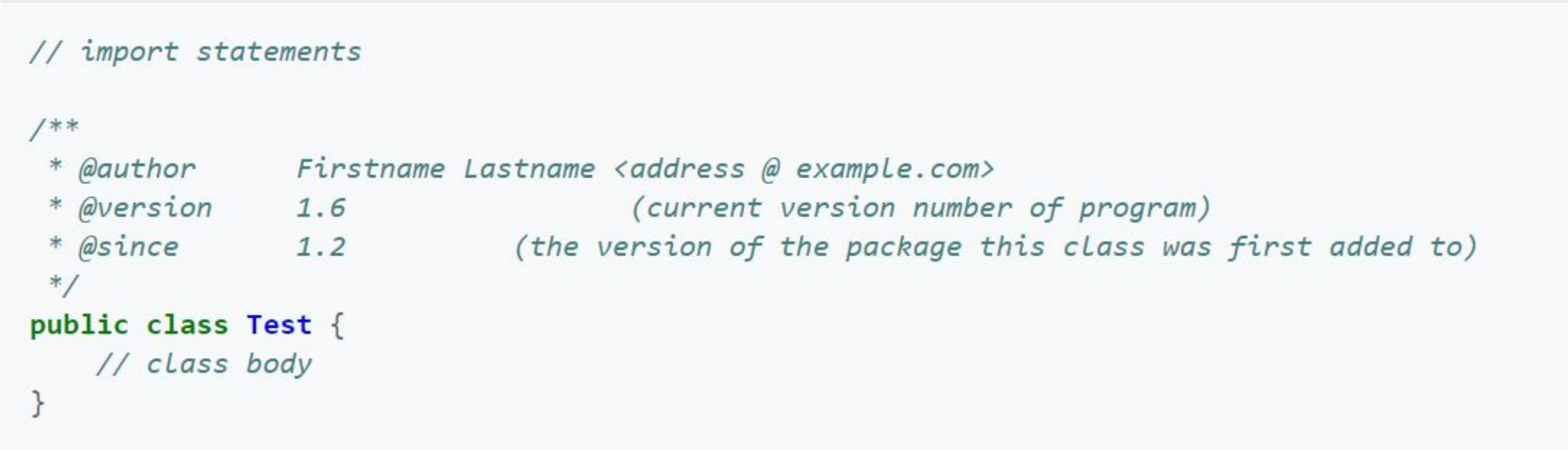
Gerrit Code Review

Collaborator
SMARTBEAR

JSLint
The JavaScript Code Quality Tool

Javadoc

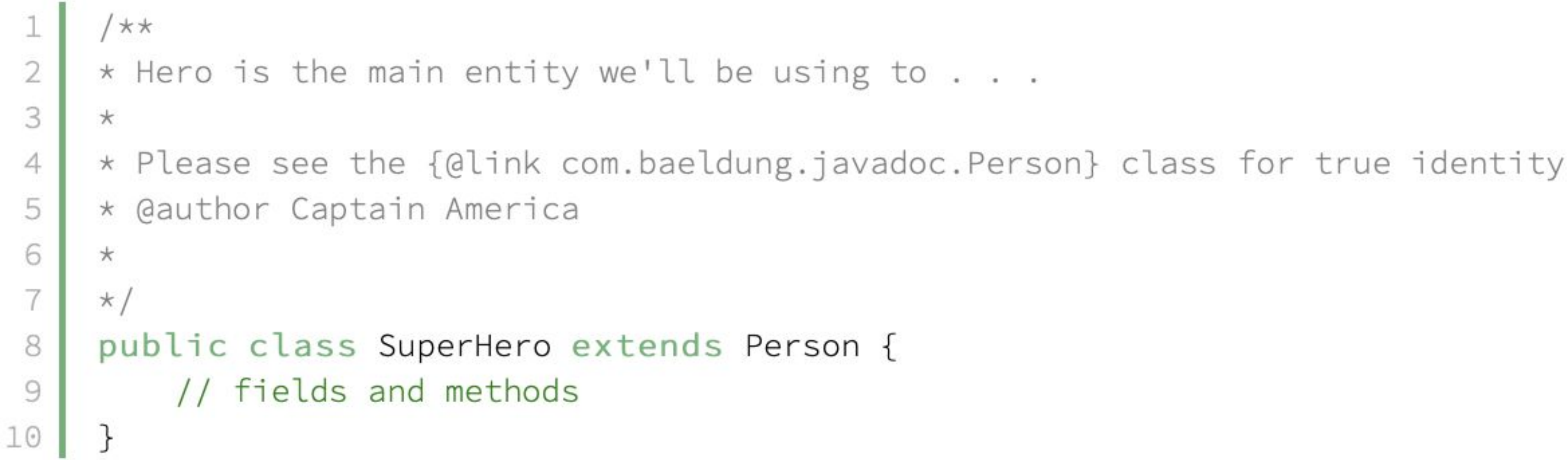
Javadoc is a documentation generator created for the Java language for generating API documentation in HTML format from Java source code.



```java
// import statements

/**
 * @author      Firstname Lastname <address @ example.com>
 * @version     1.6                 (current version number of program)
 * @since       1.2          (the version of the package this class was first added to)
 */
public class Test {
    // class body
}
```

Javadoc

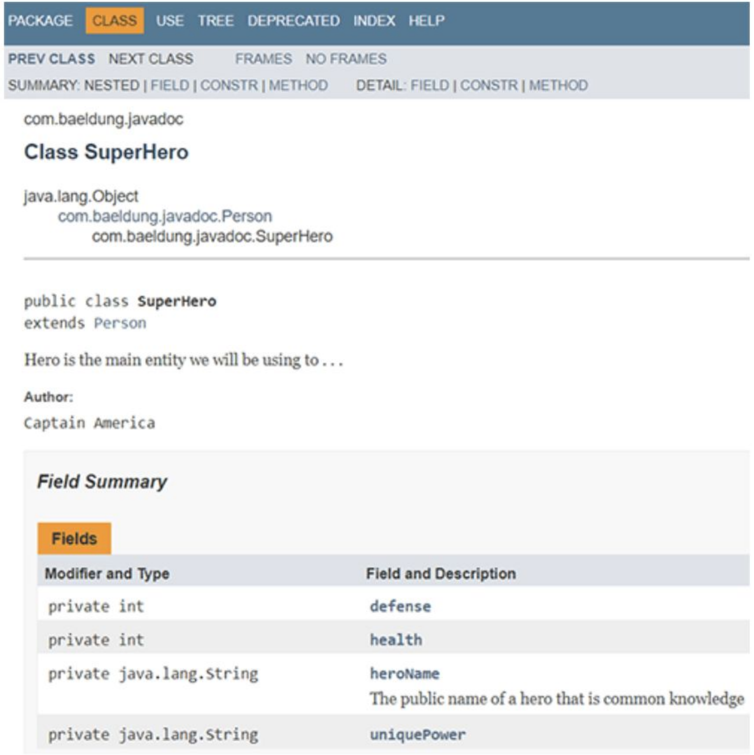
Let’s look at another example



```
/**
* Hero is the main entity we'll be using to . . .
*
* Please see the {@link com.baeldung.javadoc.Person} class for true identity
* @author Captain America
*
*/
public class SuperHero extends Person {
    // fields and methods
}
```

Javadoc

This generates:



com.baeldung.javadoc

Class SuperHero

java.lang.Object
    com.baeldung.javadoc.Person
        com.baeldung.javadoc.SuperHero

```
public class SuperHero
extends Person
```

Hero is the main entity we will be using to . . .

Author:

`Captain America`

Field Summary

Fields

Modifier and Type	Field and Description
`private int`	`defense`
`private int`	`health`
`private java.lang.String`	`heroName` The public name of a hero that is common knowledge
`private java.lang.String`	`uniquePower`

Build Management

Consists of tools that build, compile, package, and version code so that it can be redistributed as needed.

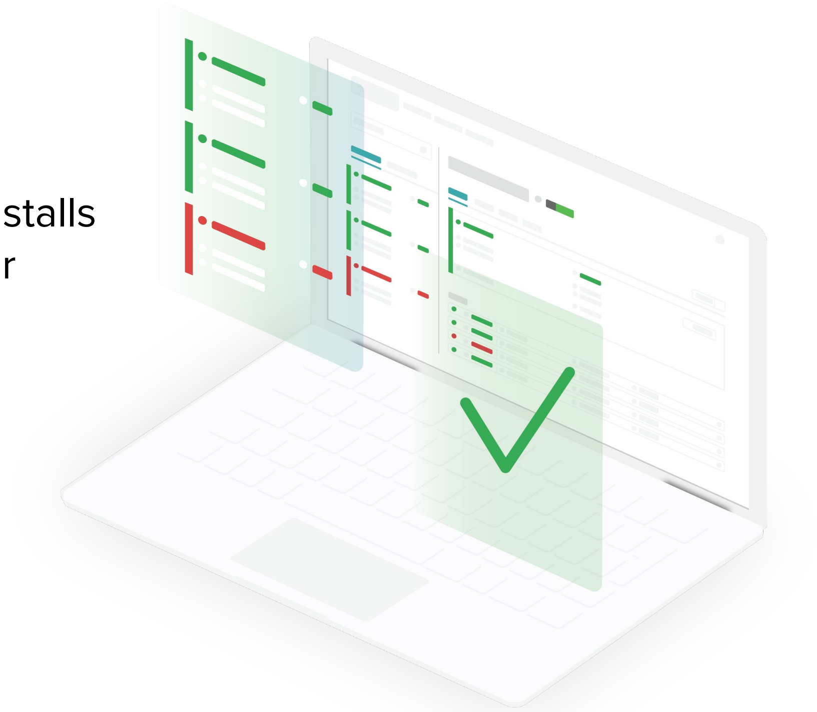
Build Management

pip v/s conda

Build Management

pip v/s conda

1. pip installs python packages whereas conda installs packages which may contain packages in other languages



Build Management

pip v/s conda

1. pip installs python packages whereas conda installs packages which may contain packages in other languages

2. Conda has the ability to create isolated environments that can contain different versions of python

Build Management

pip v/s conda

1. pip installs python packages whereas conda installs packages which may contain packages in other languages

2. Conda has the ability to create isolated that can contain different versions of python

3. Pip makes no effort to ensure that the dependencies of all packages are fulfilled simultaneously

Continuous Integration / Continuous Deployment

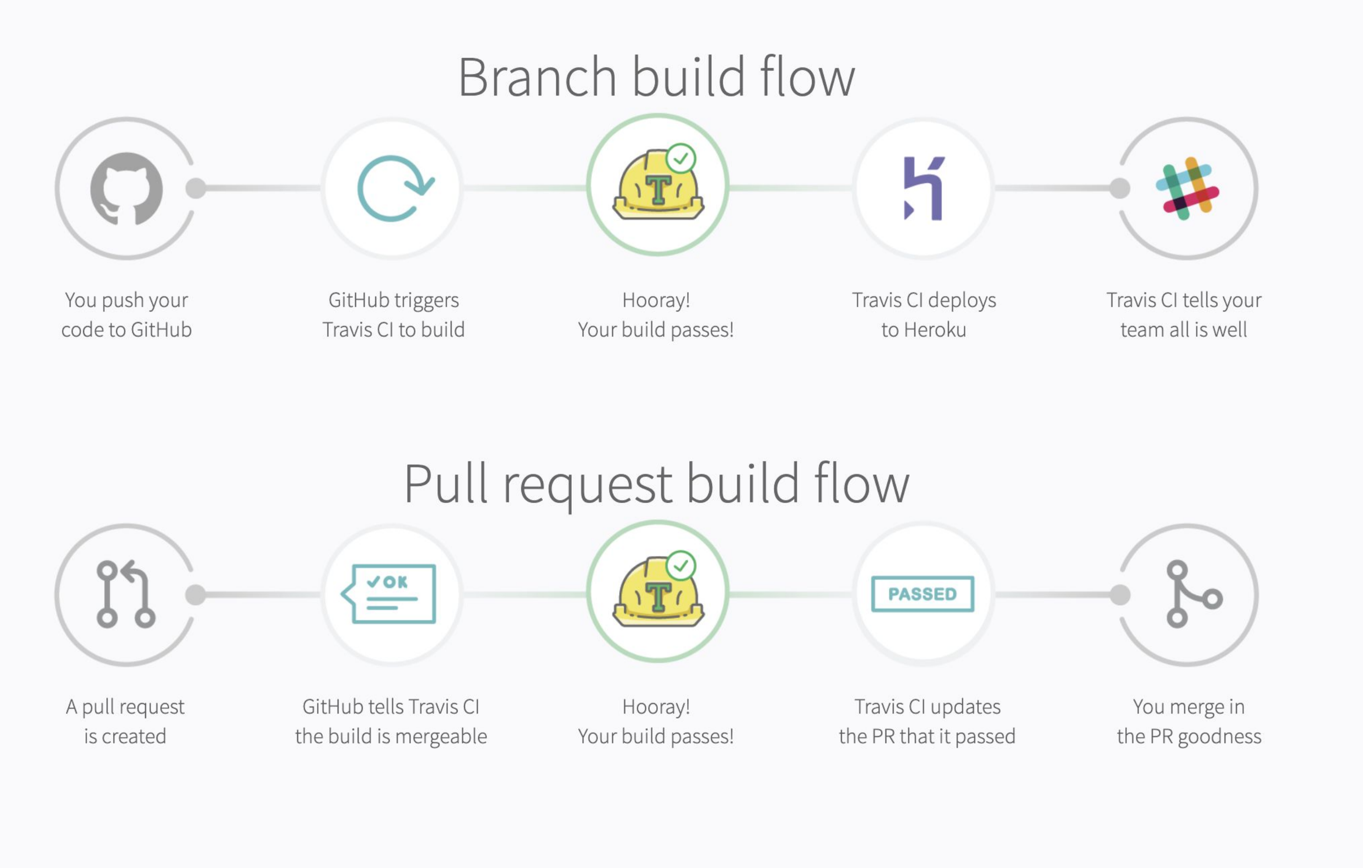
Continuous Integration (CI) is the practice of merging all developers' working copies to a shared mainline several times a day.

Continuous Deployment (CD) is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

Some CI/CD Tools

Travis CI Tutorial:
https://docs.travis-ci.com/user/tutorial/

Jenkins Sample Tutorial:
https://jenkins.io/doc/tutorials/build-a-node-js-and-react-app-with-npm/



Branch build flow

You push your code to GitHub

GitHub triggers Travis CI to build

Hooray! Your build passes!

Travis CI deploys to Heroku

Travis CI tells your team all is well

Pull request build flow

A pull request is created

GitHub tells Travis CI the build is mergeable

Hooray! Your build passes!

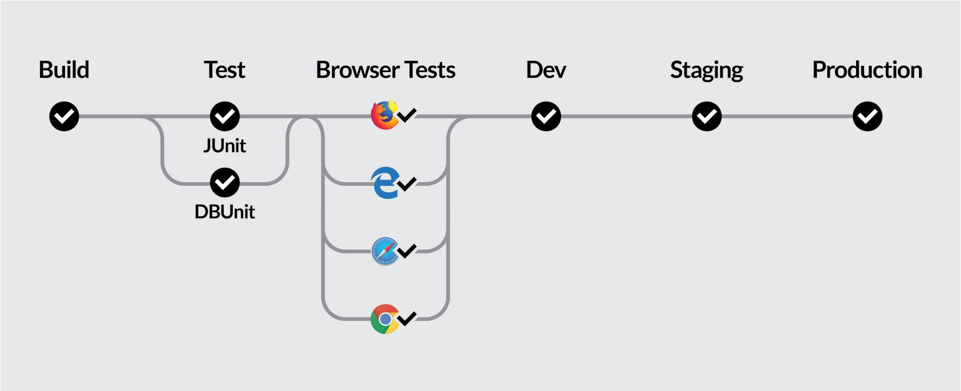
Travis CI updates the PR that it passed

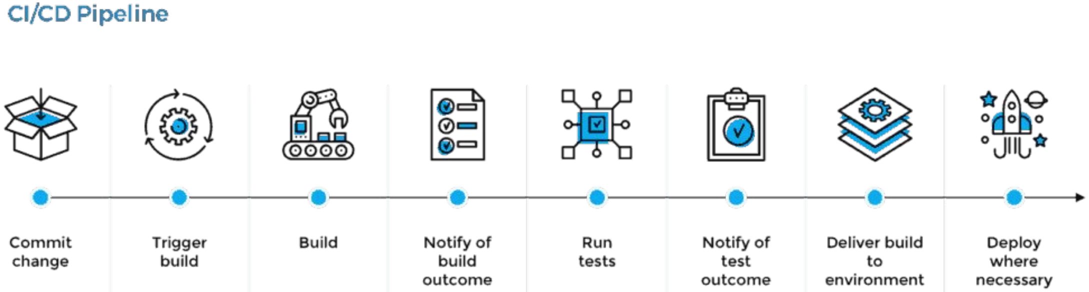
You merge in the PR goodness

Software Deployment - Good Practices

1. Reproducible Builds
2. Rollback Strategy
3. Continuous Integration and Delivery
4. Distribution in Phases
5. Documentation of Releases

Sample Build Pipelines





Let’s do some practice!

Create a build pipeline for your project. Try incorporating the following:

1. Automated Testing
2. Staging or Dev Environment
3. CI/CD
4. Monitoring

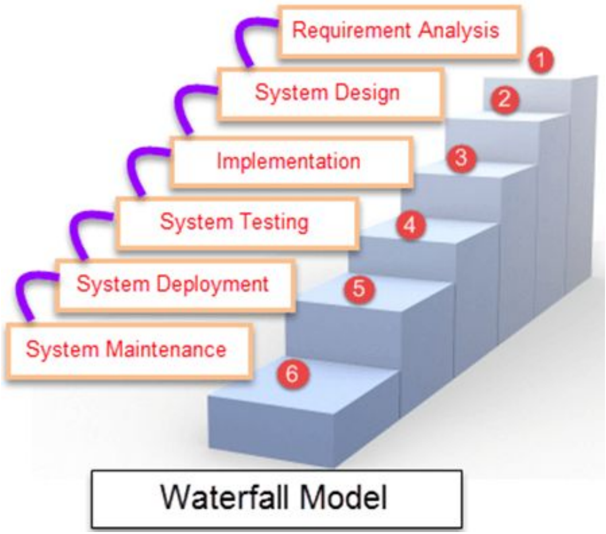
Think of the technologies you would use to actually implement each step

Software Methodologies

- Waterfall
- Agile
- Lean

Waterfall

Waterfall Model is a sequential model that divides software development into different phases. Each phase is designed for performing specific activity during SDLC phase.

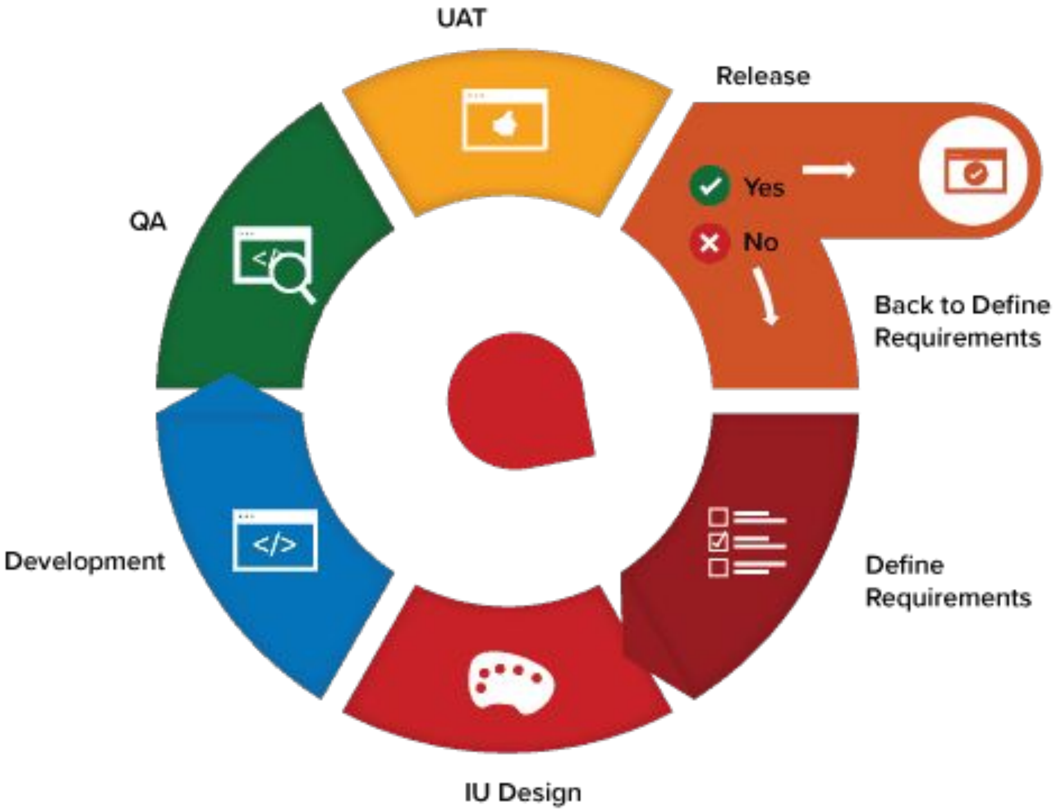


Agile

Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

Each timebox is like a mini software project that includes all the tasks necessary to release the mini-increment of new functionality:

- planning,
- requirements analysis,
- design,
- coding,
- testing, and
- documentation.


UAT
Release
Yes
No
Back to Define Requirements
QA
Development
IU Design
Define Requirements

Lean

Lean development can be summarized by seven principles, very close in concept to lean manufacturing principles

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. Optimize the whole

Team Time!