Git, code review, and how to actually work in a team

- By Matt Wildman

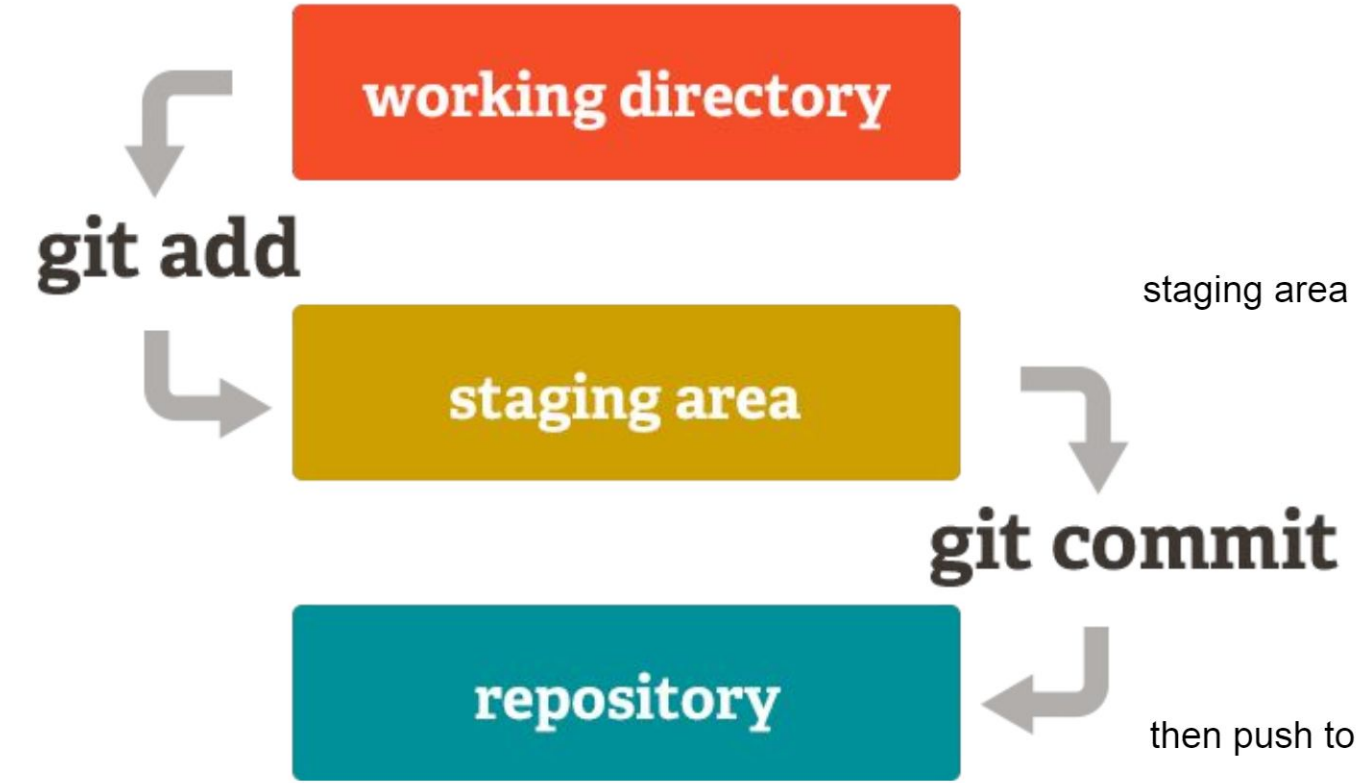
What’s the Point?

- Collaborating on code is hard
  - Easy to break other code
  - Large codebase that may not be documented
  - Project is potentially live, users will complain if it goes down
- Distributing tasks is hard
  - How to prioritize workload
  - Context switching is expensive
  - Potentially wasted development time

Git

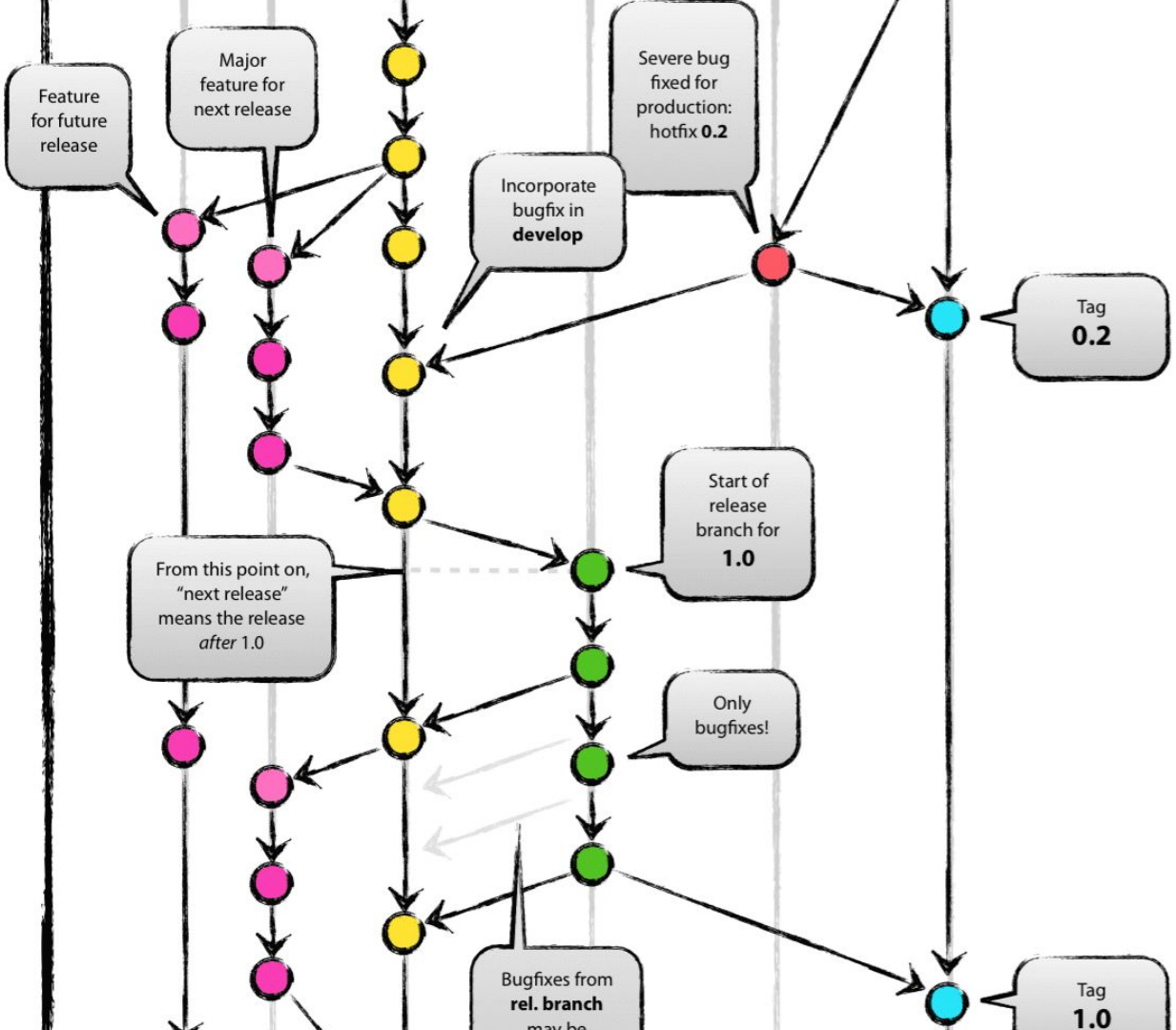
- Single contributor workflow is straightforward

Workflow



Git

- Multiple contributor workflow isn't

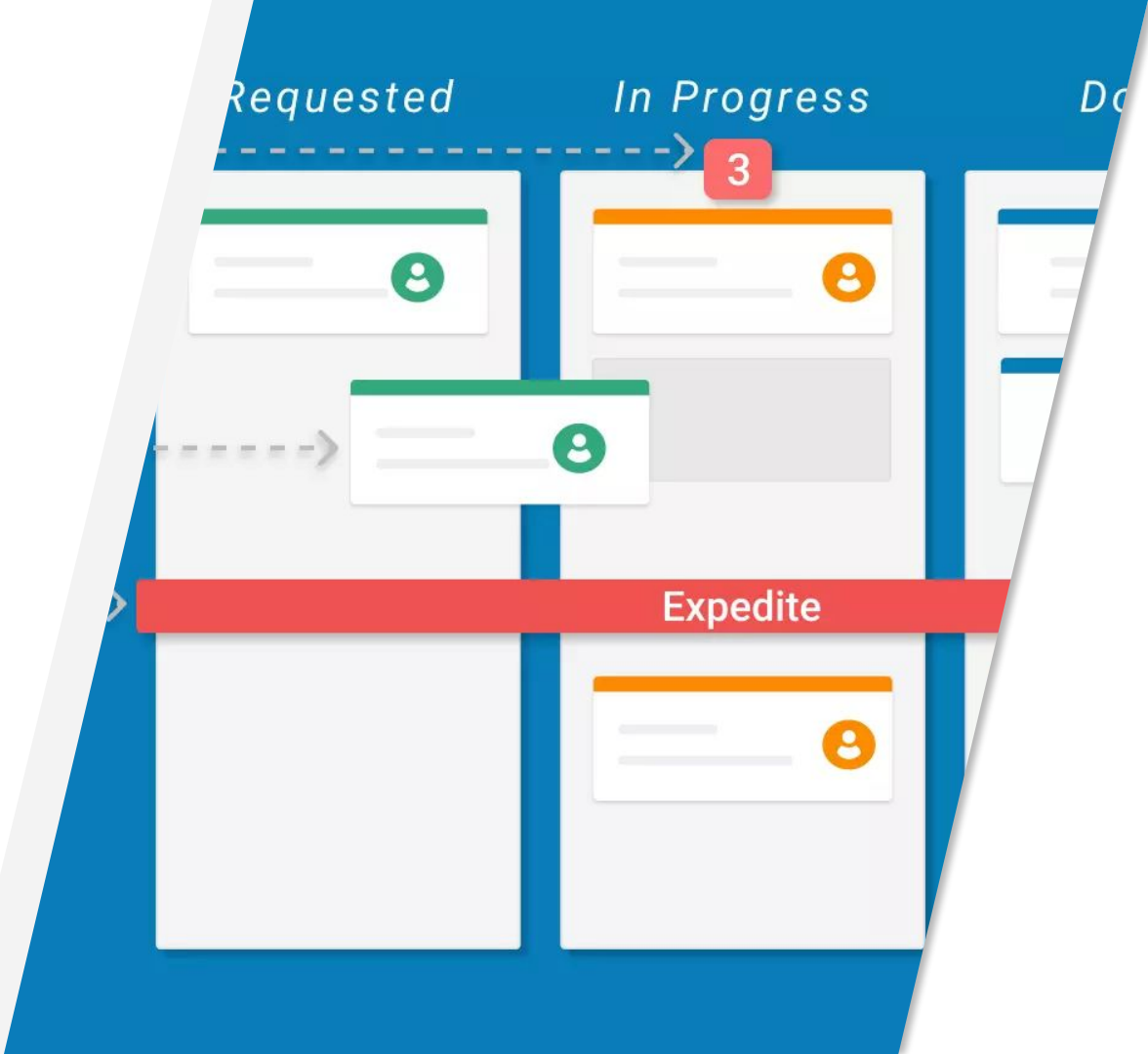


Typical Task Lifecycle

- Acquire task and clarify requirements
- Implement task on separate branch (With Tests!)
- Resolve conflicts with Development branch
- Submit pull request (PR)
- Get feedback through code review
- Fix and repeat until satisfied
- Merge into development branch (Preferably squash and merge/rebase)
- Deploy as necessary

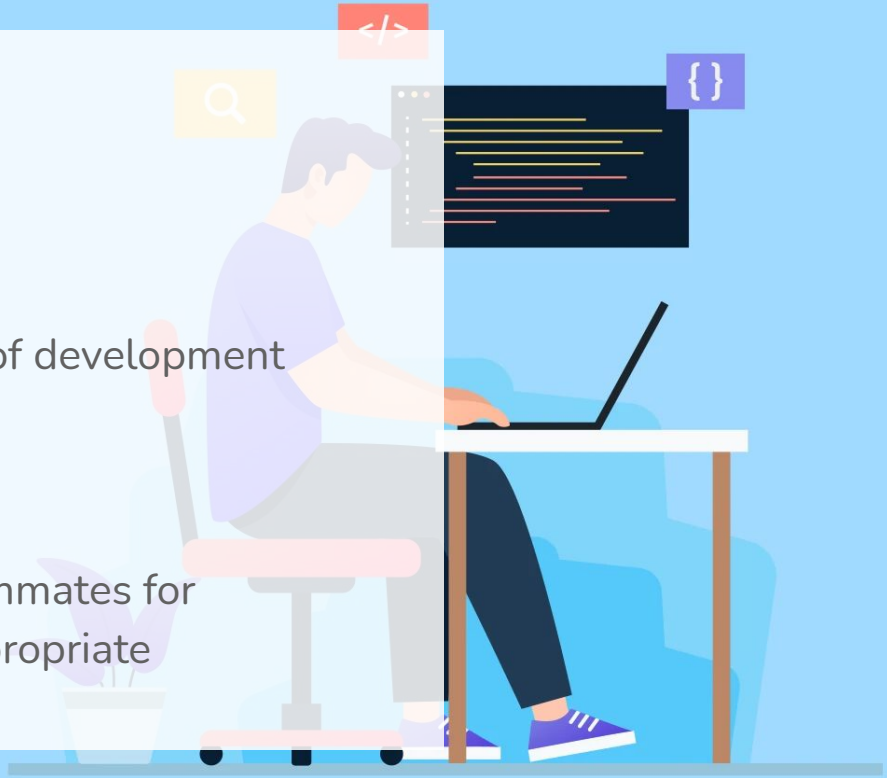
Acquiring Task

- Sometimes will be given by manager, sometimes from kanban board
- Don’t forget to clarify what is actually being requested!!!

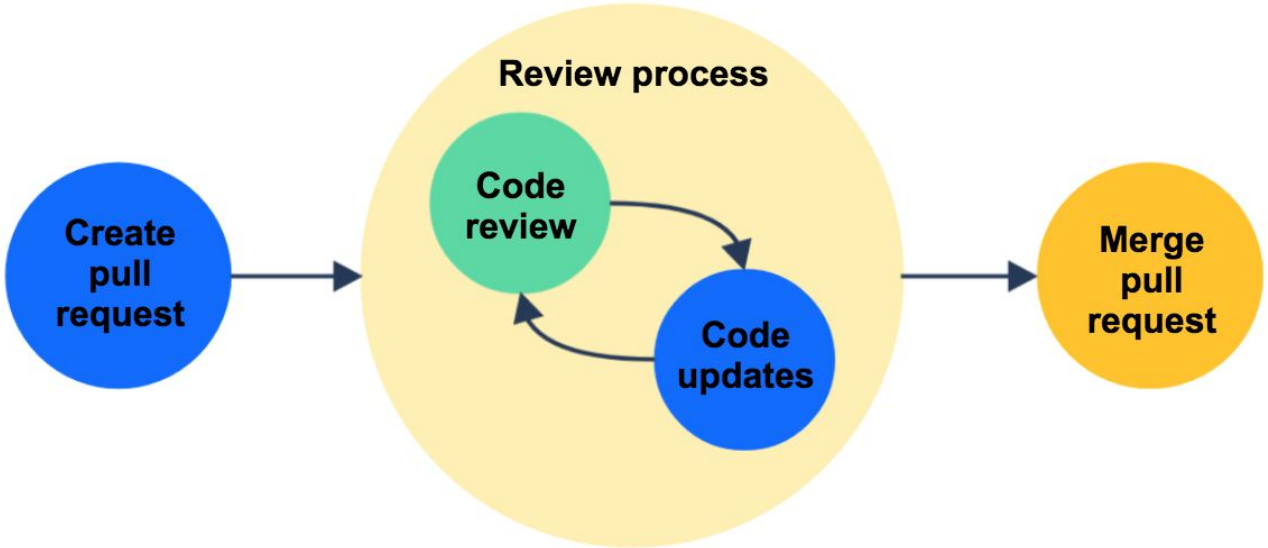


Implement

- Make branch off of development branch
- Code :)
- Don’t forget tests
- Reach out to teammates for assistance as appropriate



PR and Code Review



Code Review In Depth

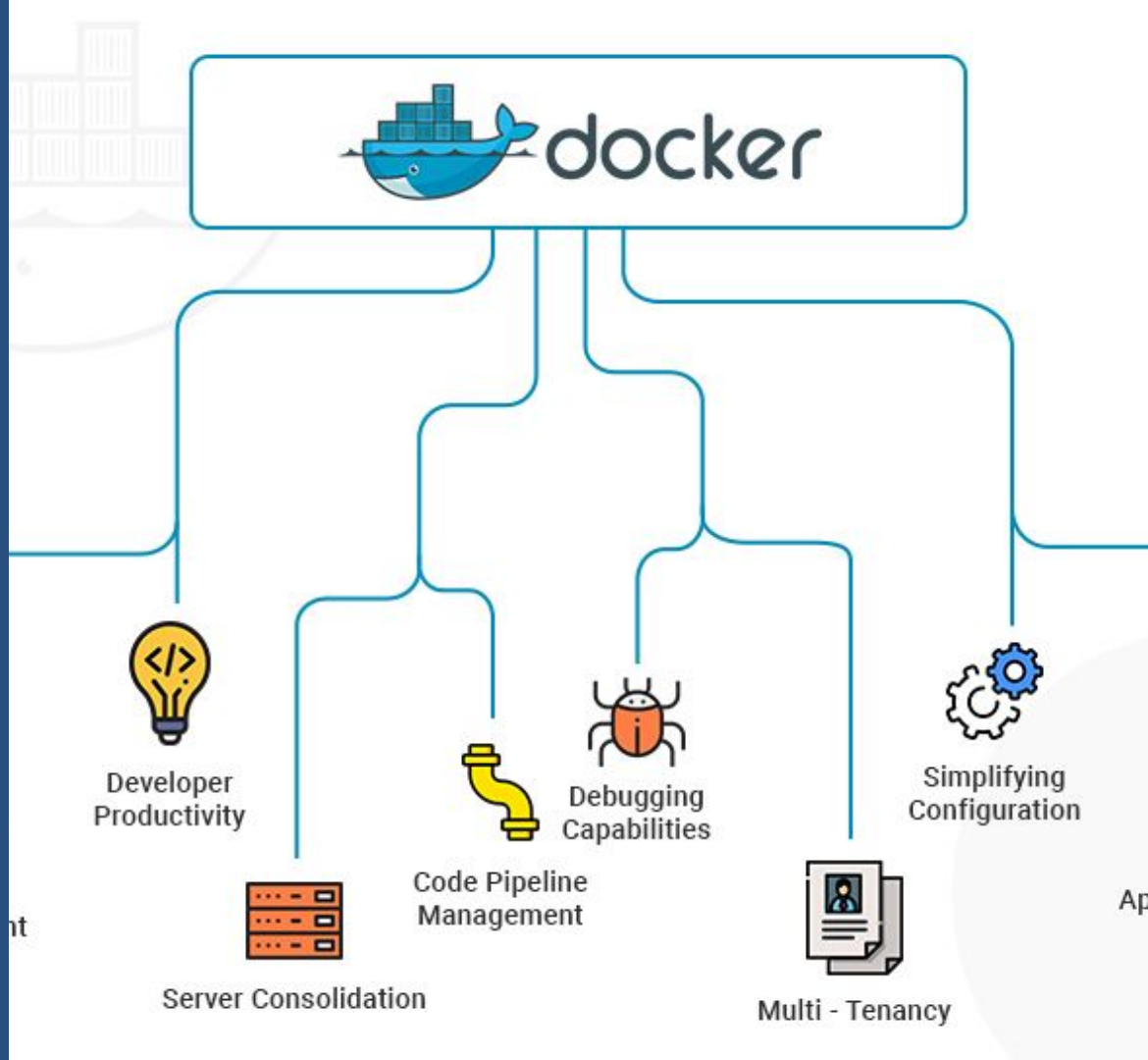
- A cycle that may happen multiple times per PR
- Don’t take things personally
- Potential Feedback
  - Code Style
  - Code Clarity
  - Refactoring
  - Missing Test Cases
  - Commenting
  - Correctness

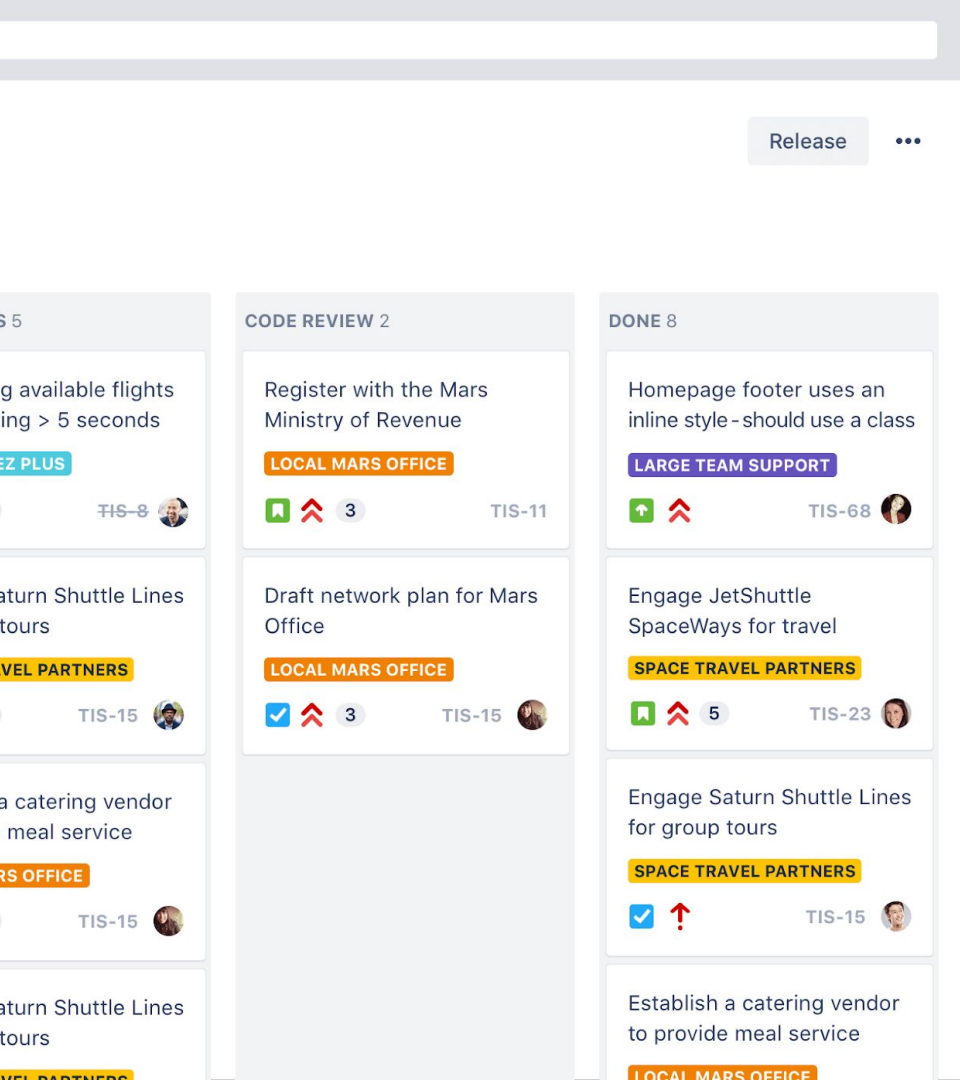
Merge!

- Once you've got the requisite number of ok's, you'll be cleared to bring your feature branch into the development branch
- Teams will have a preference on how you do this
    - My personal preference is squash and merge

Deployment!

- This varies extremely from team to team
- Often there may be alpha and QA environments to deploy to first for testing
- You may need to test on an alpha environment before PR is approved
- Security team may run analysis tools before deployment
- May also require additional bureaucracy





Done?

- Mark task completed in Jira board or equivalent
- Unless you broke something this task is probably done
  - If something goes down in production then someone will probably @ you
  - Then move onto hotfix section

Hotfix :(

- Don’t panic, it happens to everyone
- Figure out what went wrong
  - Try to get logs or records leading up to issue
- Take your time to fix it right
  - Write tests to cover the case that caused the issue
- Code Review (Not even hotfixes are exempt)
- Teams will likely have an expedited means to deploy your fix