Not All Projects Use Pull-Requests

- https://github.com/torvalds/linux/pull/17#issuecomment-5654674

torvalds commented on May 11, 2012

I don't do github pull requests.

github throws away all the relevant information, like having even a valid email address for the person asking me to pull. The diffstat is also deficient and useless.

Git comes with a nice pull-request generation module, but github instead decided to replace it with their own totally inferior version. As a result, I consider github useless for these kinds of things. It's fine for *hosting*, but the pull requests and the online commit editing, are just pure garbage.

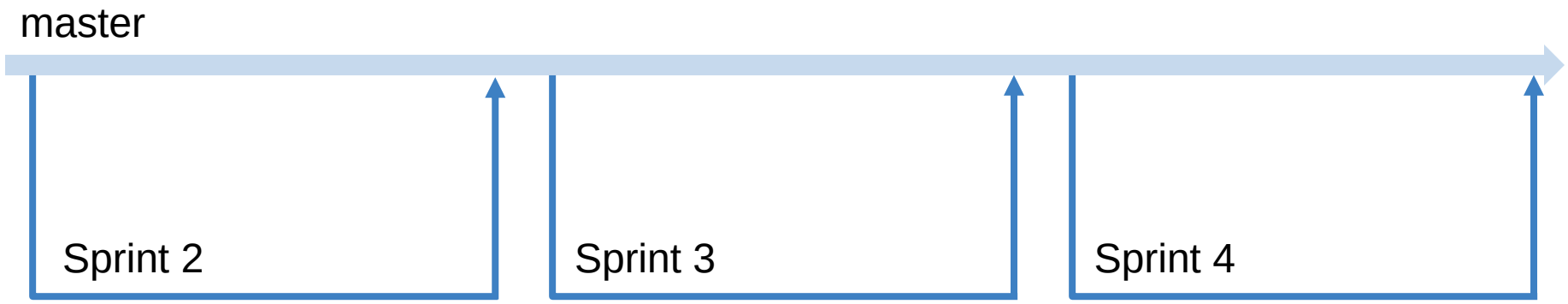
I've told github people about my concerns, they didn't think they mattered, so I gave up. Feel free to make a bugreport to github.

```
Linus
```

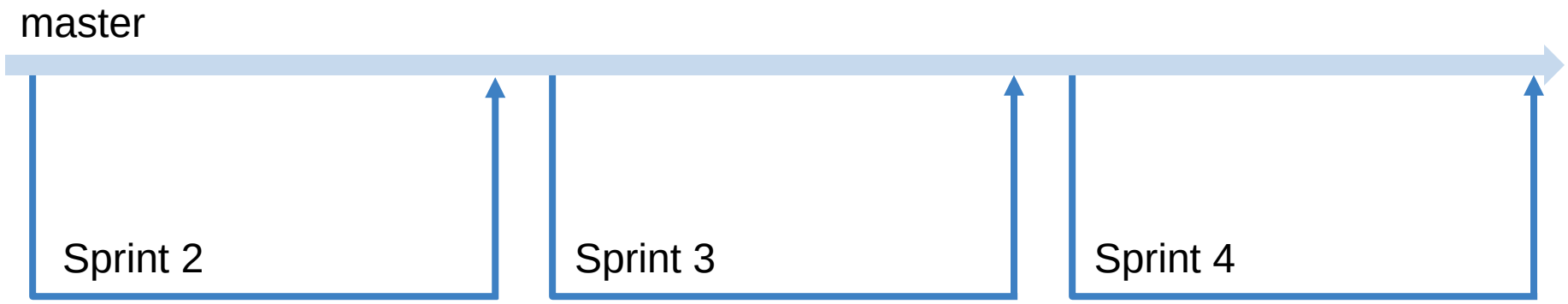
A GitHub-Based Sprint-Branch Workflow

- There are many workflows — this is just an example

A GitHub-Based Sprint-Branch Workflow



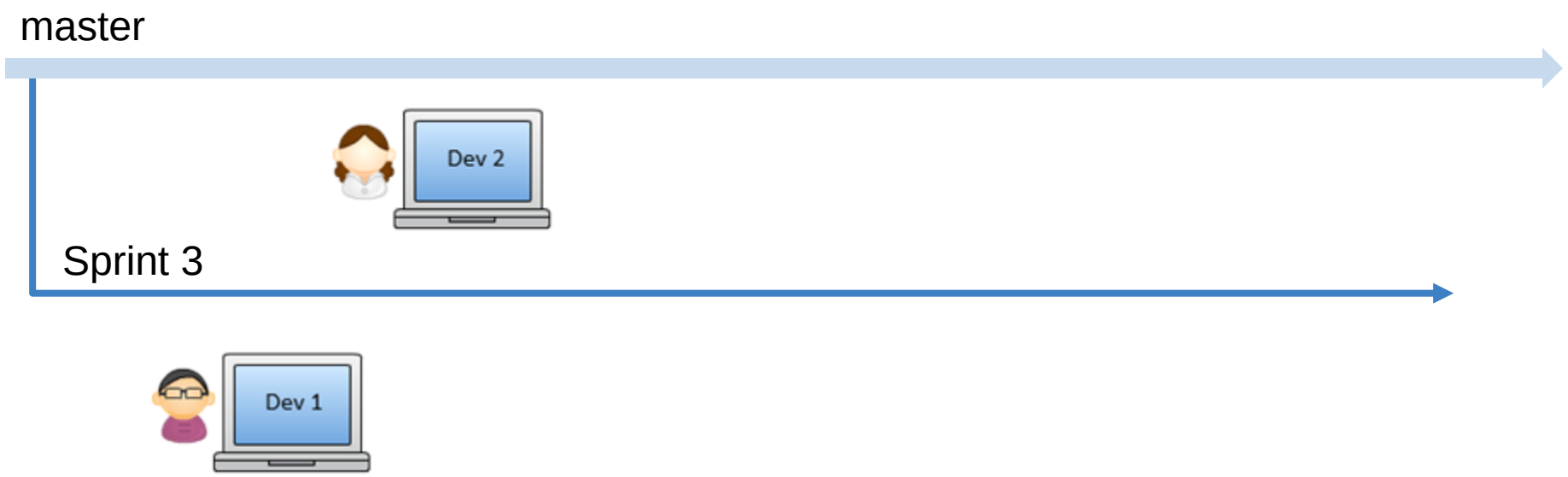
A GitHub-Based Sprint-Branch Workflow



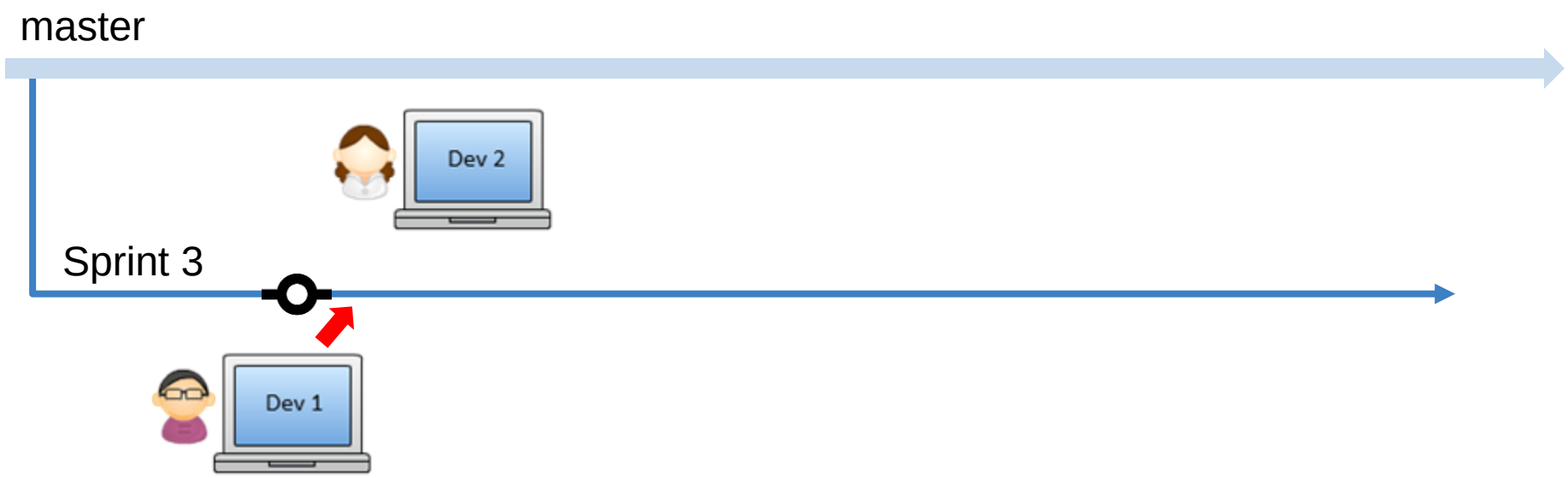
- The *master* (AKA *main*) branch always contains fully integrated and tested code, ready to release to customers
- Tasks will be implemented on a *sprint branch*
  - e.g., "Sprint3"
- If something goes wrong during the sprint, you still have working code on the *master branch*

Programming in our Sprint-Branch Workflow

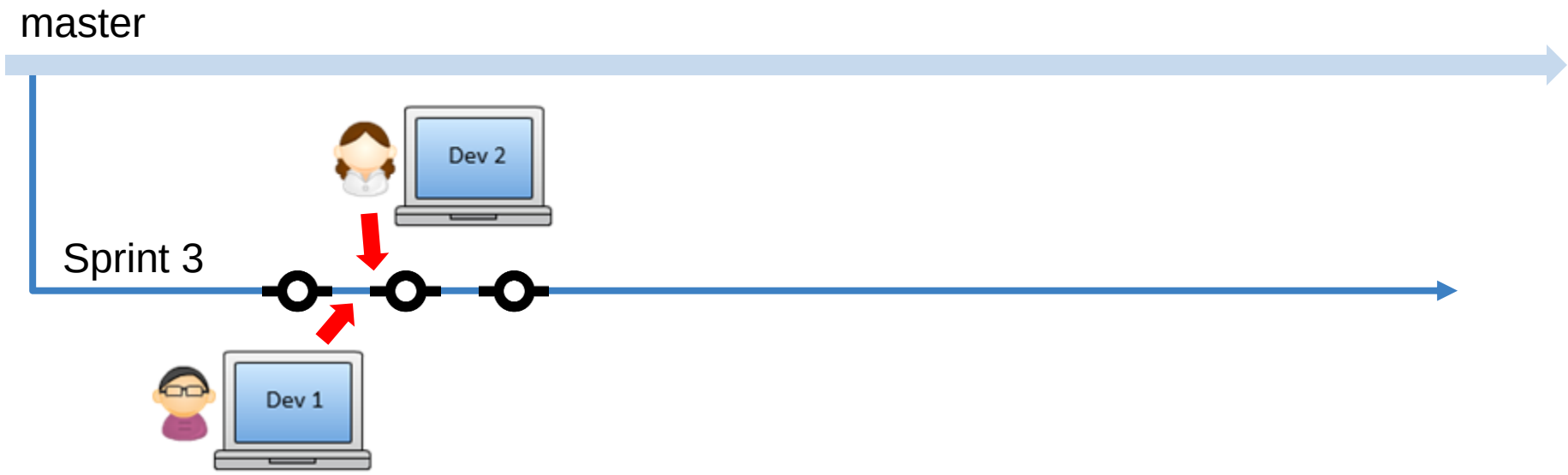
Programming in our Sprint-Branch Workflow



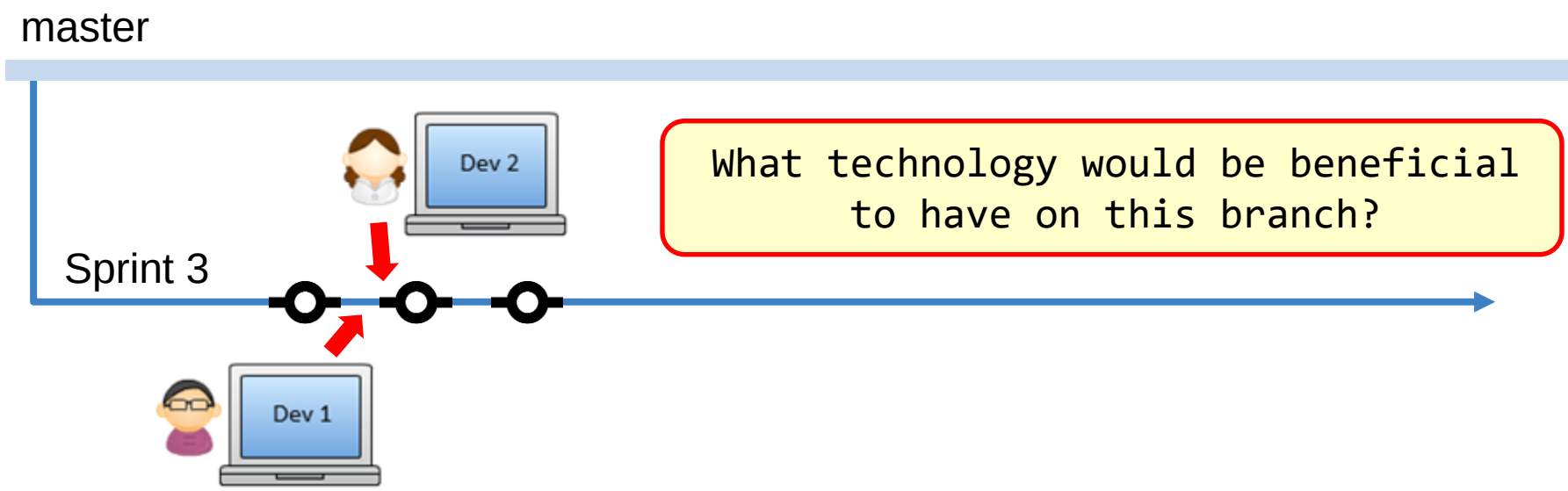
Programming in our Sprint-Branch Workflow



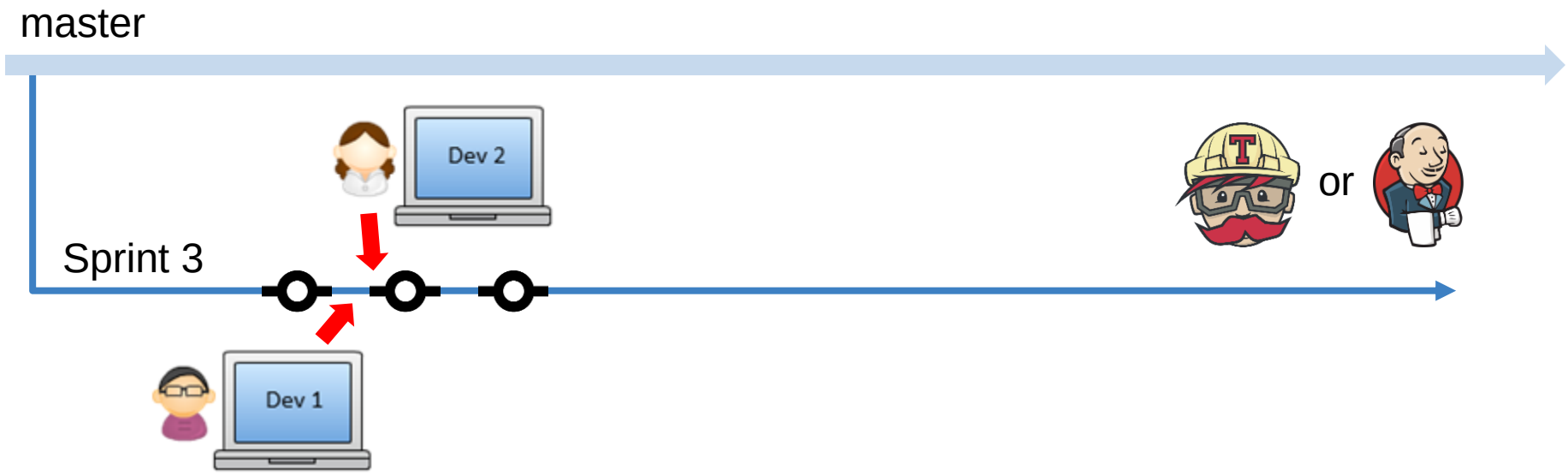
Programming in our Sprint-Branch Workflow



Programming in our Sprint-Branch Workflow

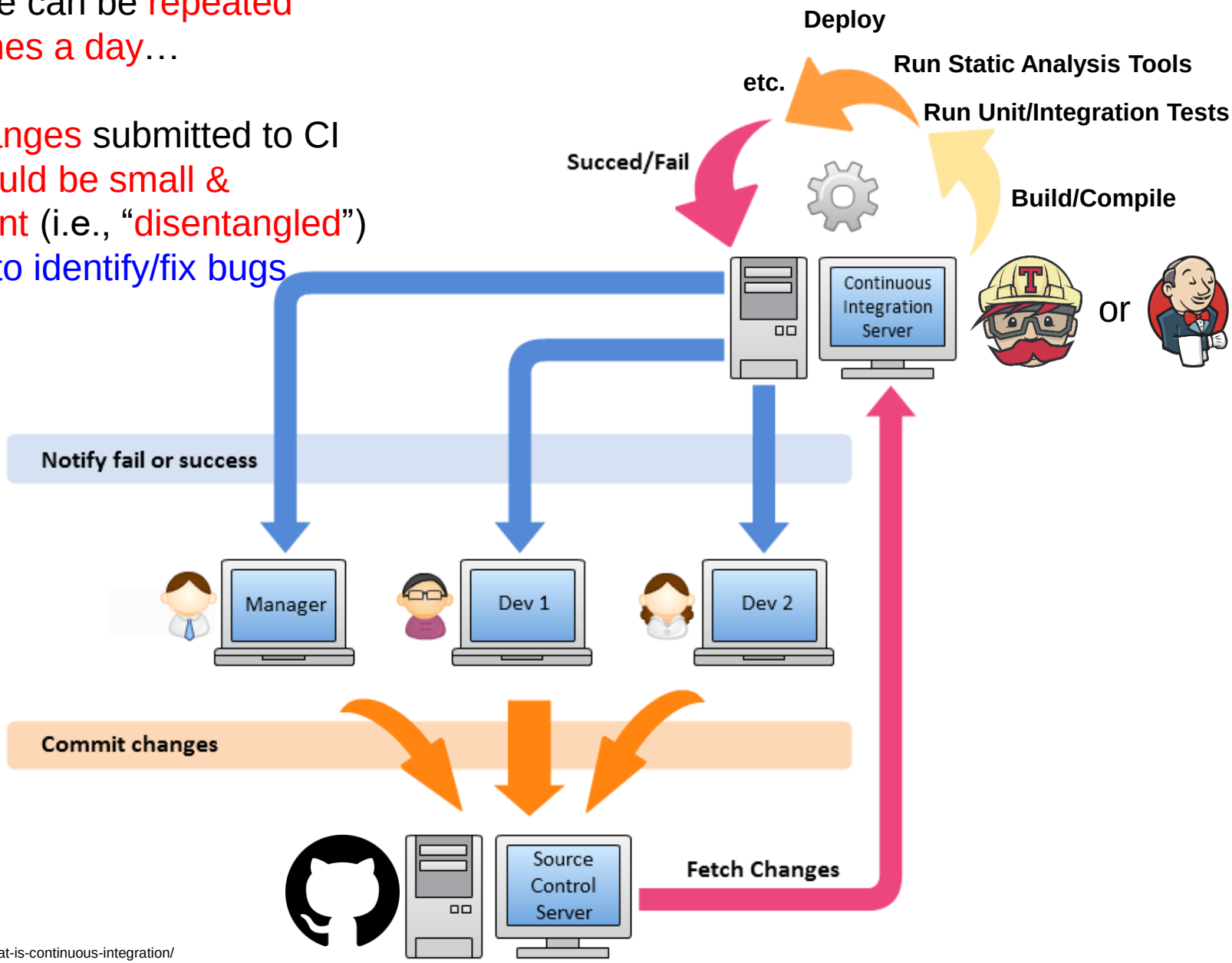


Programming in our Sprint-Branch Workflow



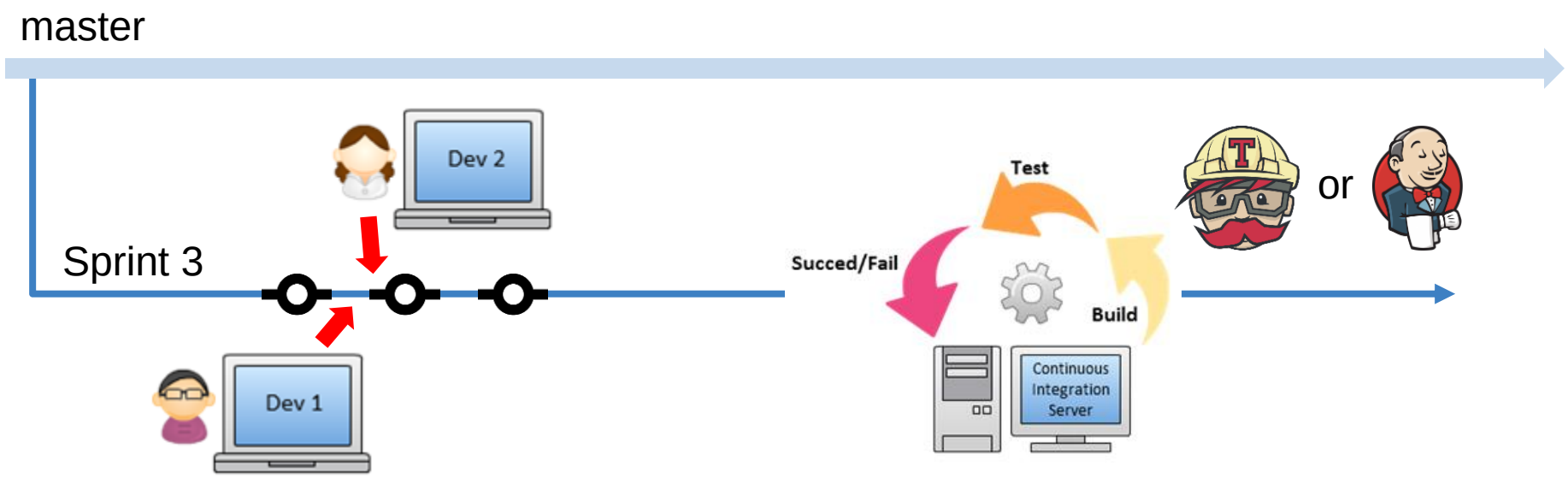
(Review Slide) Continuous Integration (CI)

- Entire cycle can be repeated several times a day…
- Ideally changes submitted to CI server should be small & independent (i.e., "disentangled")
  - Easy to identify/fix bugs

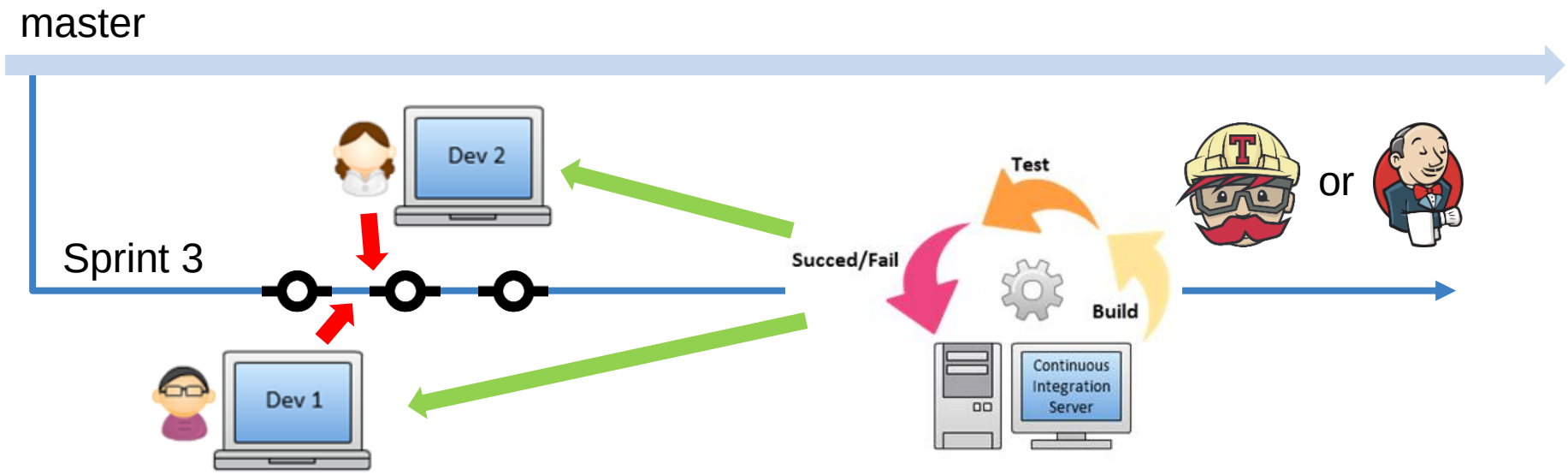


https://www.code-maze.com/what-is-continuous-integration/

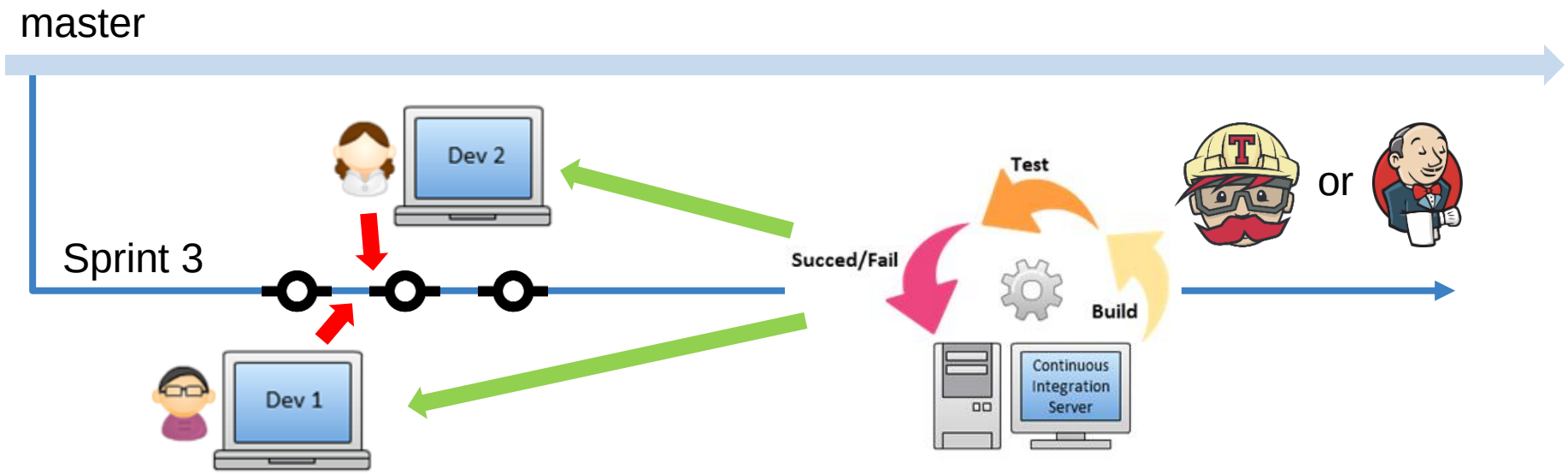
Programming in our Sprint-Branch Workflow



Programming in our Sprint-Branch Workflow

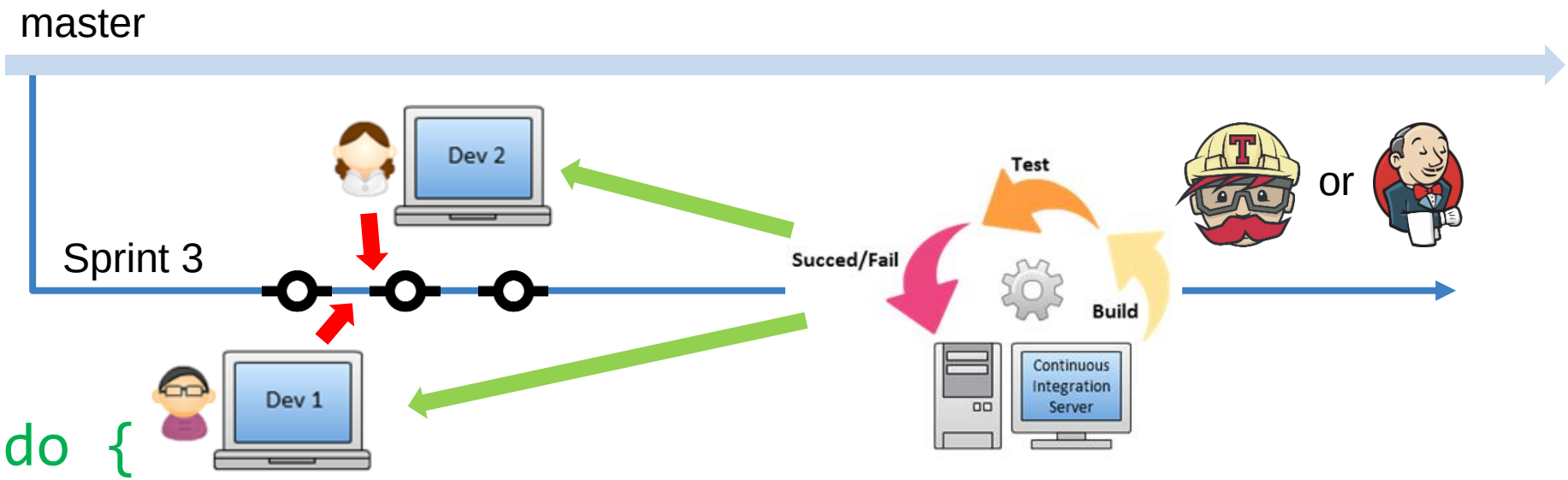


Programming in our Sprint-Branch Workflow



- Start by creating a *sprint branch* in the repository
- Each Developer
  - checks-out the *branch*
  - implements their Task in the *workspace* (AKA *sandbox*) on their own computer
  - create/execute unit-level tests
  - remove defects in their private *workspaces*
  - commits their completed Tasks to the *sprint branch*, making their changes available for testing with those committed by other Developers

Integration in our Sprint-Branch Workflow

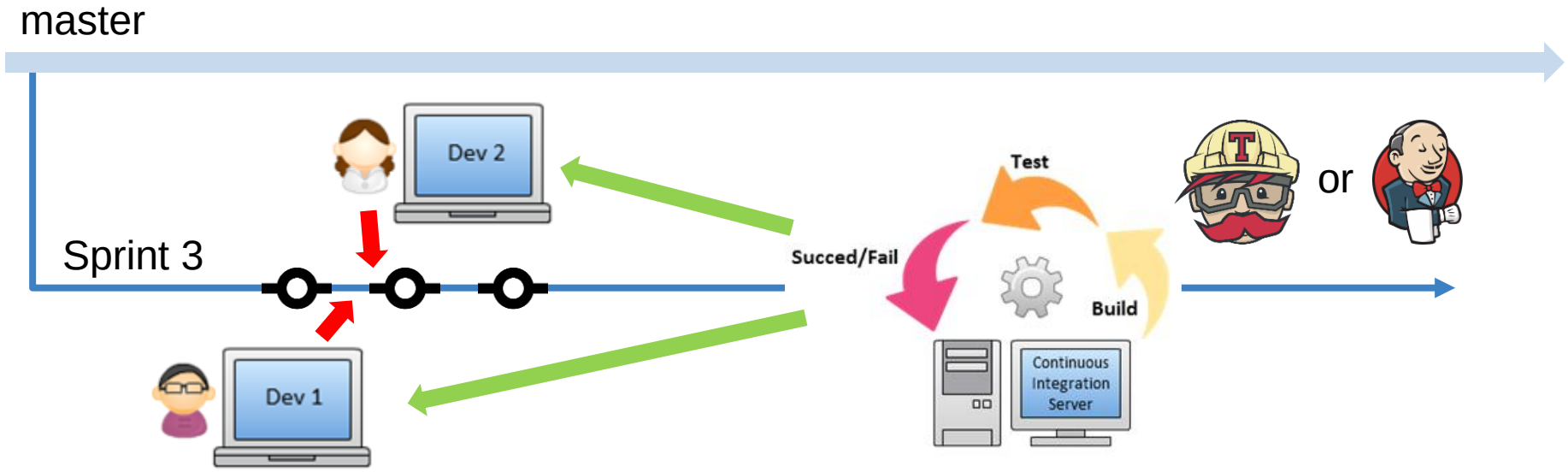


do {

- Automated Build System
  - checks out a fresh copy of all changes committed to the branch and
  - rebuilds the executables
  - executes the automated integration tests and
  - flags the build as “success” or “failure”
- Developers
  - repair integration defects and
  - commit their fixes to the branch

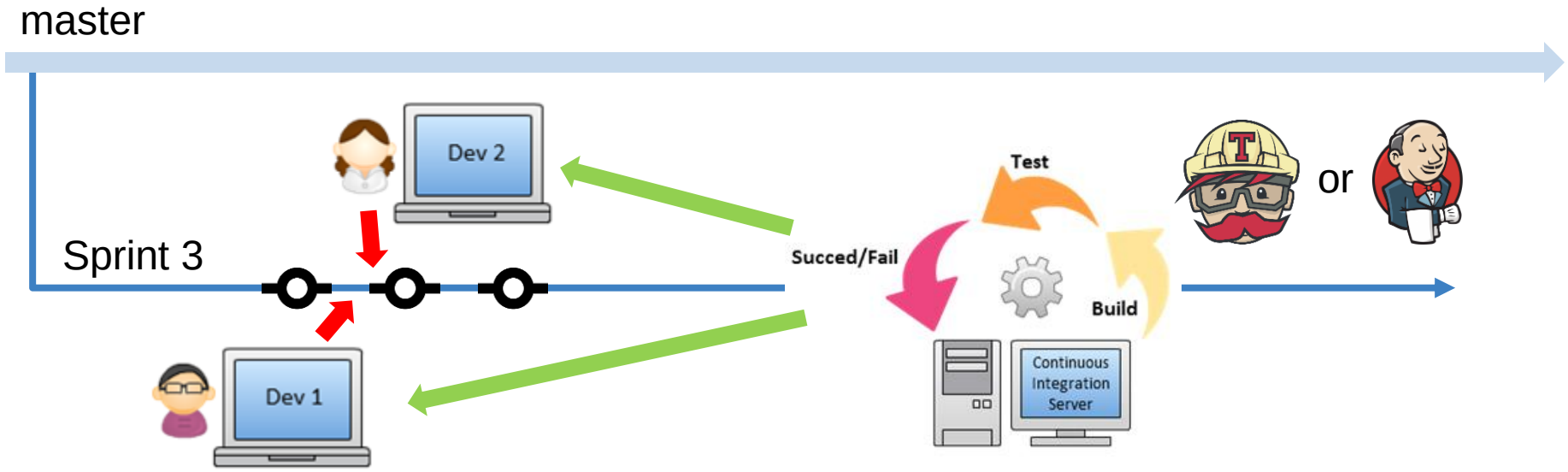
} until (allTasksInSprintAreCompleted)

Acceptance Testing in our Sprint-Branch Workflow



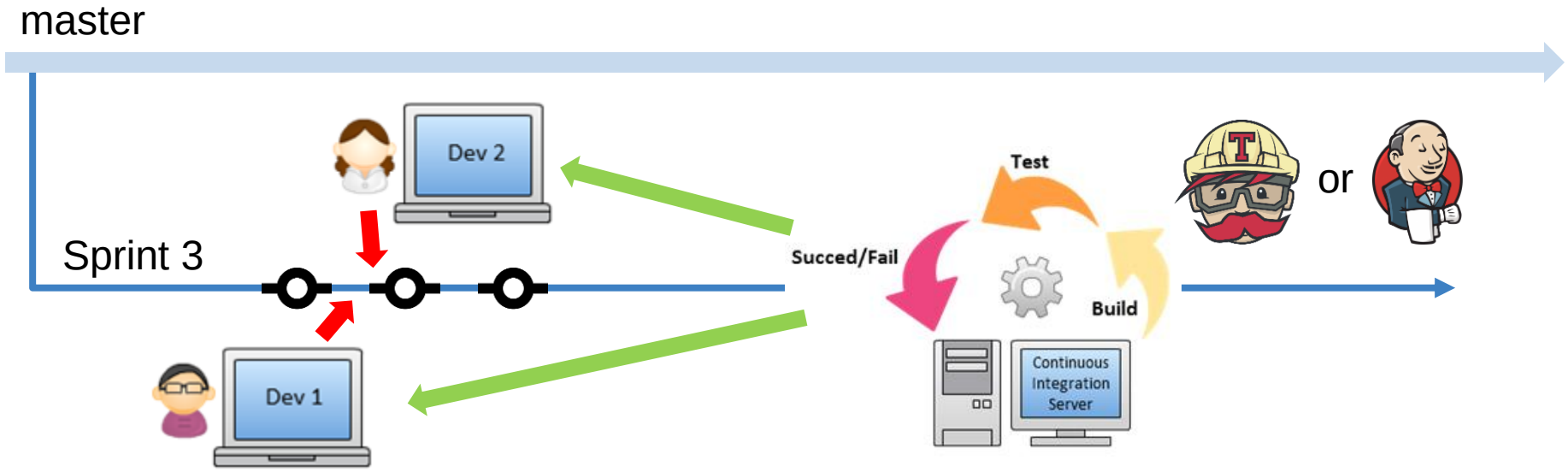
- Question: What about manual Acceptance Testing? When should manual Acceptance Testing be performed?

Acceptance Testing in our Sprint-Branch Workflow



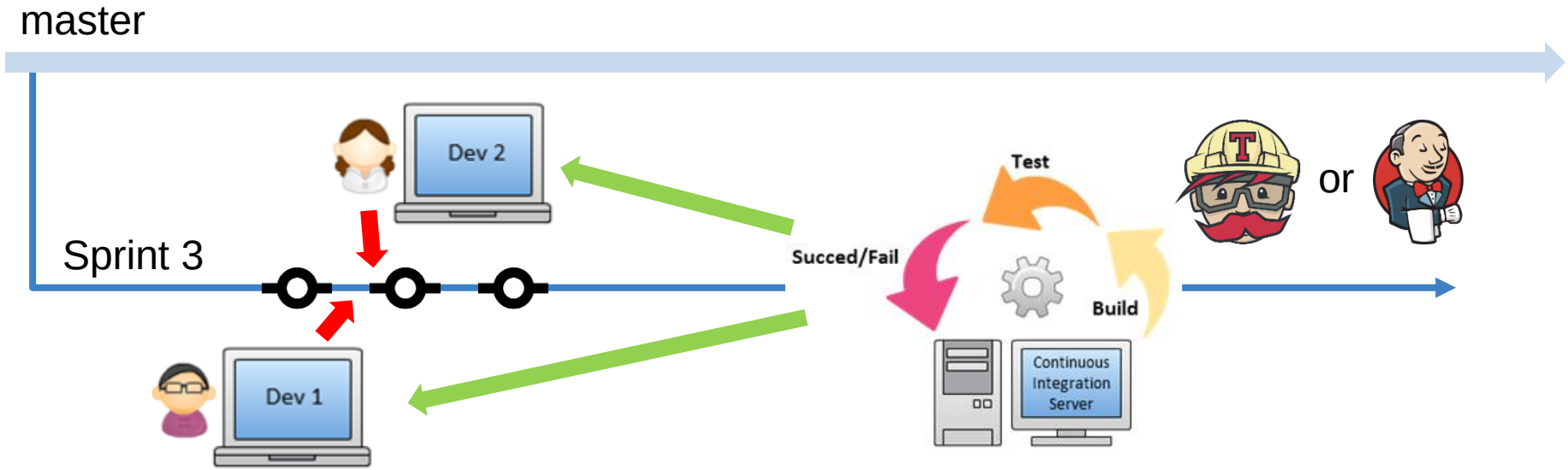
- Manual Acceptance Testing begins after all Tasks have been integrated on the *branch*

Acceptance Testing in our Sprint-Branch Workflow



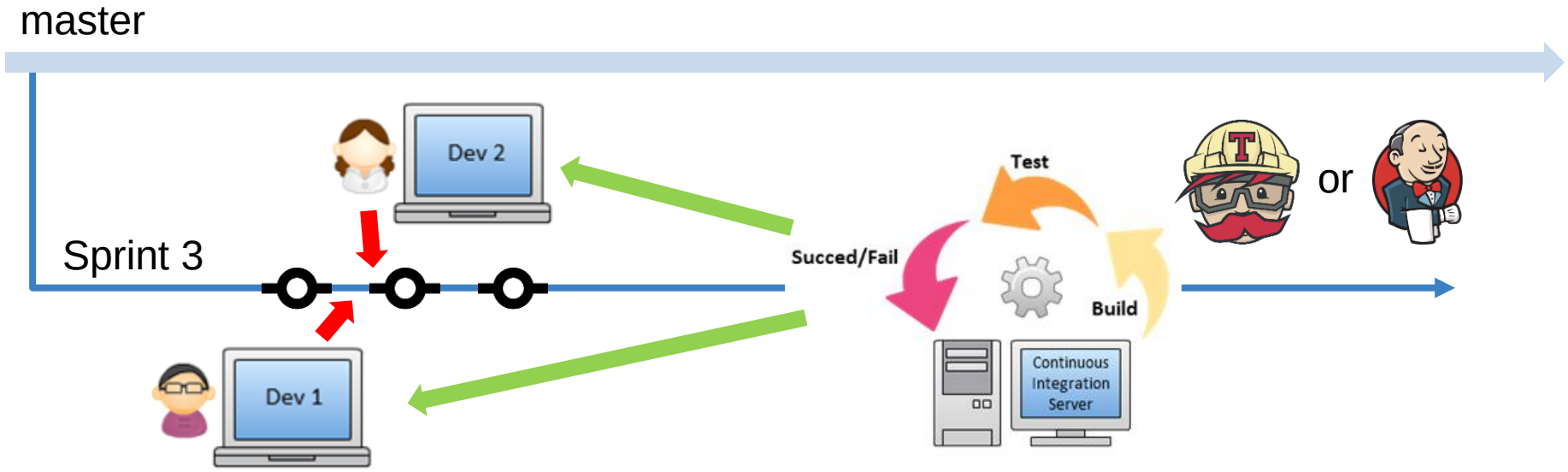
- Manual Acceptance Testing begins after all Tasks have been integrated on the *branch*
- Developers
  - repair defects in the private workspaces on their own computers,
  - get the unit-level tests passing, and
  - commit their changes to the *branch*

Alpha/Beta Testing in our Sprint-Branch Workflow



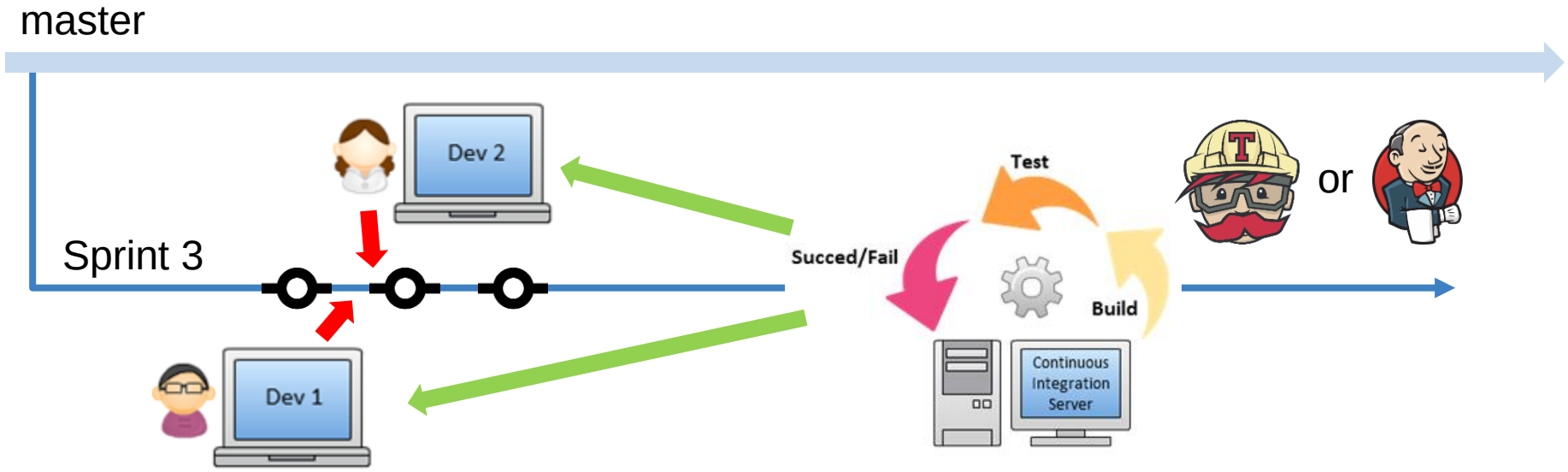
- After all Tasks are integrated and all Acceptance Tests pass, the Build System's result is called a *Release Candidate*
- A *Release Candidate* is ready for limited-production testing
- The Team performs Alpha Testing
- Sprint Review occurs about this time
- Customers perform Beta Testing
- Question: What happens with bugs discovered in Alpha and Beta testing?

Alpha/Beta Testing in our Sprint-Branch Workflow



- Trivial defects might be repaired in this sprint
- Complex defects recorded as Bugs and added to the Product Backlog to be scheduled for a future sprint

Product Release in our Sprint-Branch Workflow



- When the sprint is finished, the branch contains the fully integrated and tested changes from its Tasks
- Merge the *sprint branch* to the *master branch*
- Future sprints will each have their own branch

Summary of our Sprint-Branch Workflow

- Each sprint has its own *sprint branch*
- Programming and Unit-Level Testing occur in a Developer's private *workspace* on their own computer (where most defects are found and repaired)
- All code on a *sprint branch* passes its Unit-Level Tests and is ready for Integration Testing
- The Automated Build System builds the *sprint branch*
- Integration occurs on a *sprint branch*
- Acceptance Testing occurs on a *sprint branch*
- Release Candidate(s) are built on a *sprint branch*

Advantages of our Sprint-Branch Workflow

Advantages of our Sprint-Branch Workflow

- Scalable: Supports multiple teams, each with their own sprint branch
- Even if a sprint totally bombs, the Team still has a fully-tested, ready-to-release version of the product on the *master branch*
- Continuous integration of the *sprint branch*
- A Developer can fix a broken build by reverting a commit

Advantages of our Sprint-Branch Workflow

- Many if not most defects are found and repaired in a Developer's private workspace:
  - Pair Programming
  - Test-Driven Development
  - Unit-Level Testing
  - Static Analysis

- An urgent repair to released code can be performed on a *hot fix branch* concurrently with sprint development

Limitations of our Sprint-Branch Workflow

Limitations of our Sprint-Branch Workflow

- Code Reviews activities are not emphasized
- The Build System will begin integration testing as soon as a Developer commits their unreviewed changes
- This approach floods Integration Tests with structural defects that could have been removed before integration if the Team was reviewing proposed changes before they were committed to the sprint branch
- Potential risk of merge conflicts with master branch at the end of the sprint

Story-Branch Variation of Our Sprint-Branch Workflow