Collaboration And TDD

Tuesday, January 8

Oregon State University

Version Control (VCS)

Why VCS?

Version Control (VCS)

Why VCS?

- Manage changes over time to source code
  - Less fear of making mistakes that can't be undone
  - Freedom to experiment with new features, or different implementations, without messing up what you have (Branching)

Version Control (VCS)

Why VCS?

- Manage collaboration
    - Developers can work independently on different parts of a project and merge the parts
    - When conflicting versions arise, have a framework to reconcile differences

Git

Distributed version control system (DVCS)

- This is in distinction to most previous VCS's, in which full version history is kept in one central location (where it can become unavailable or even *lost*).

- Working with Git, each developer has a **copy** of the project's entire history.

- Most operations can be done using local files.

Git

At the core of Git is a content-addressable filesystem

- Key/value storage: give Git your file; Git stores it away and gives you a unique hash that identifies it.
- Git stores a series of snapshots of an entire project at particular times when you choose to save them.
- This makes branching and merging cheap.
- Almost all actions only **add** data--this means they can be undone if needed!

Terminology

Index (stage): Record of the versions of files ready to be committed.

Commit ("revision"): a preserved version of all tracked files. When you commit, git saves all modifications that have been staged.

Branch: represents an independent line of development. What you do to one branch has no effect on any other branch.

Master: The default development branch. (But don't use it for development. Master should always contain only **working** code!)

Merge: Merging takes the changes from one branch, and applies them into another.

Head: Git's way of referring to the current snapshot.

Terminology

Fetch: Get the latest changes from an online repository without merging them in.

Pull: Bringing in changes from a remote repository to the local one.

Push: Sending your committed changes to a remote repository

Using git locally

git init: makes the current directory a git repository (current directory contents are not part of the repo until you add them)

git add: adds file to the index/staging area

git commit: commits the files in the index/staging area

Using git (not just locally...)

git clone <URL>: downloads a repository from the specified location and copies it to your local machine

git pull: downloads changes from the remote repo; applies them to local repo and working directory

git push: uploads changes to the remote repository and merges them in

Working with branches

git checkout <name>: selects which branch or commit you want to work with; updates files in the working directory to match this version

git checkout -b <branch>: creates a new branch and checks it out

git branch: lists the branches

git branch <name>: creates a new branch and does not check it out.

The `master` branch

It contains the "truth."

- Milestone tags
- Deployment branch

All code merge into the master needs to be **reviewed**

Short demo of git

GitHub

Hosting service for version control using Git

It's the other part that makes Git awesome

Has builtin project management and collaboration tools

GitHub issues

Issues allow you to **plan** and track your work.

Write a **clear explanation** for each issue. If it's a user story, write the whole story as the description.

Tag the issue with a **label**, and a milestone.

Assign the issue

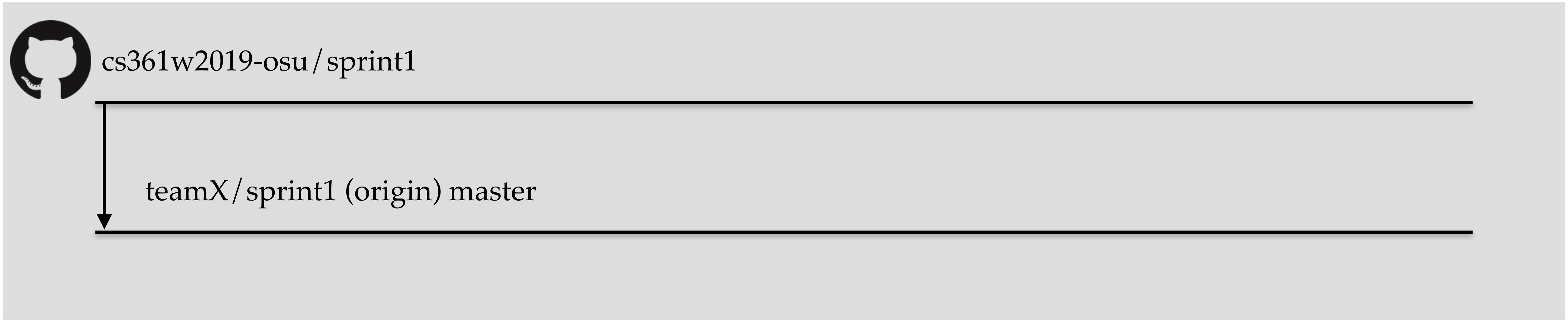
Close it by referencing the issue number (#) in the PR.

GitHub team workflow

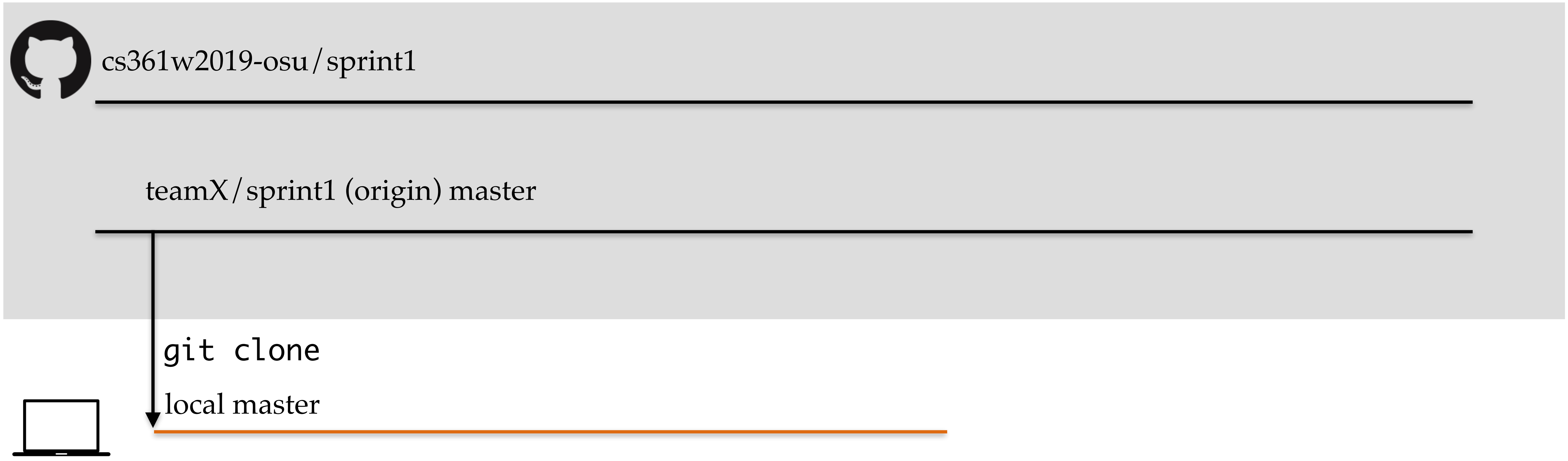
How do you use Git and GitHub effectively when working in teams?

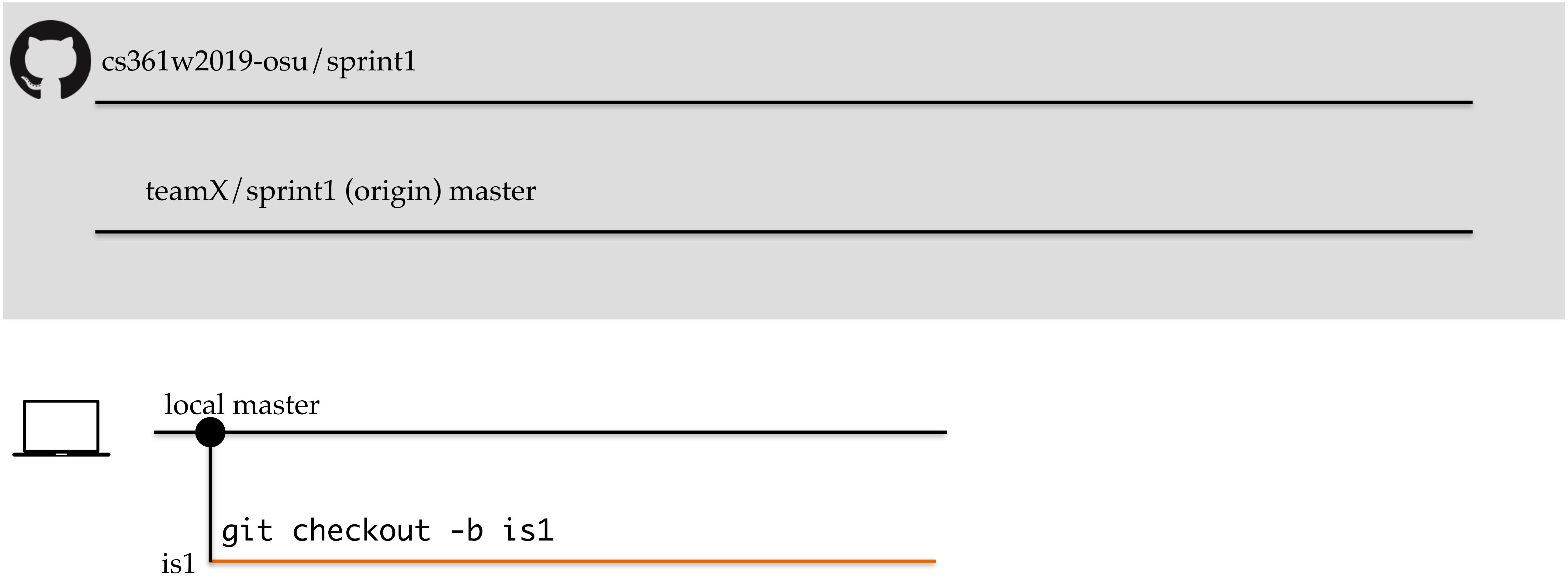
Oregon State University

1 cs361w2019-osu/sprint1



`teamX/sprint1` is also known as **"upstream"**

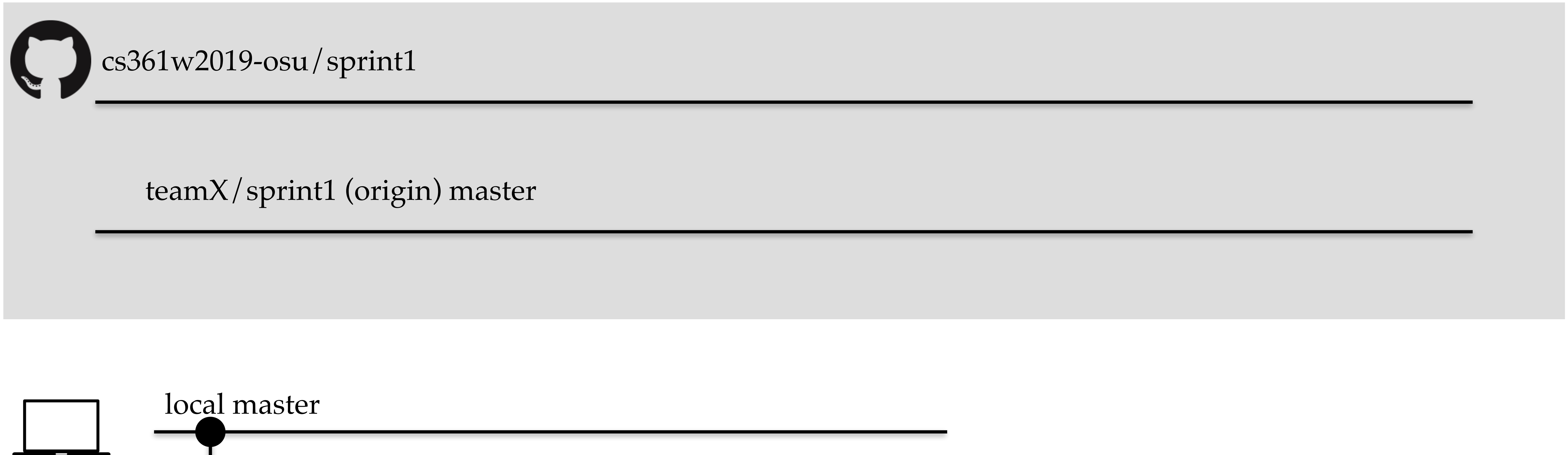

cs361w2019-osu/sprint1
teamX/sprint1 (origin) master
git clone
local master



```
git checkout -b is1
```

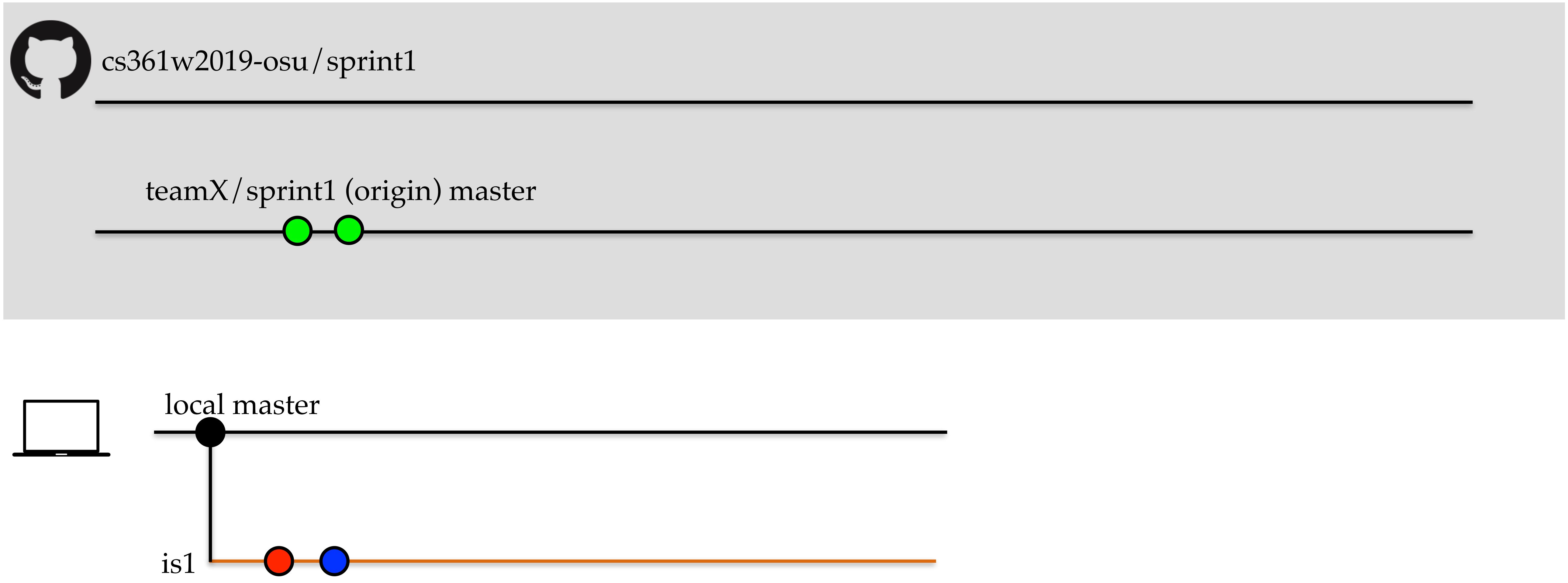
Naming branches using the issue number makes it easier to manage/identify them.

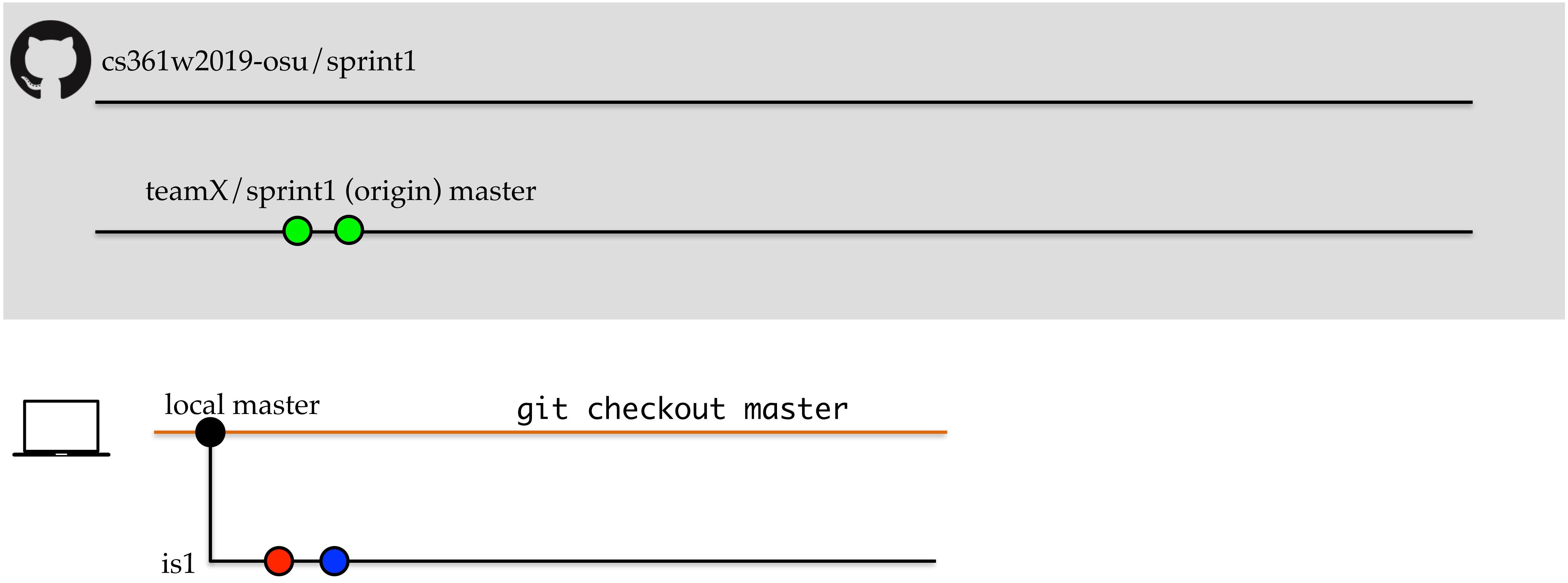
Oregon State University



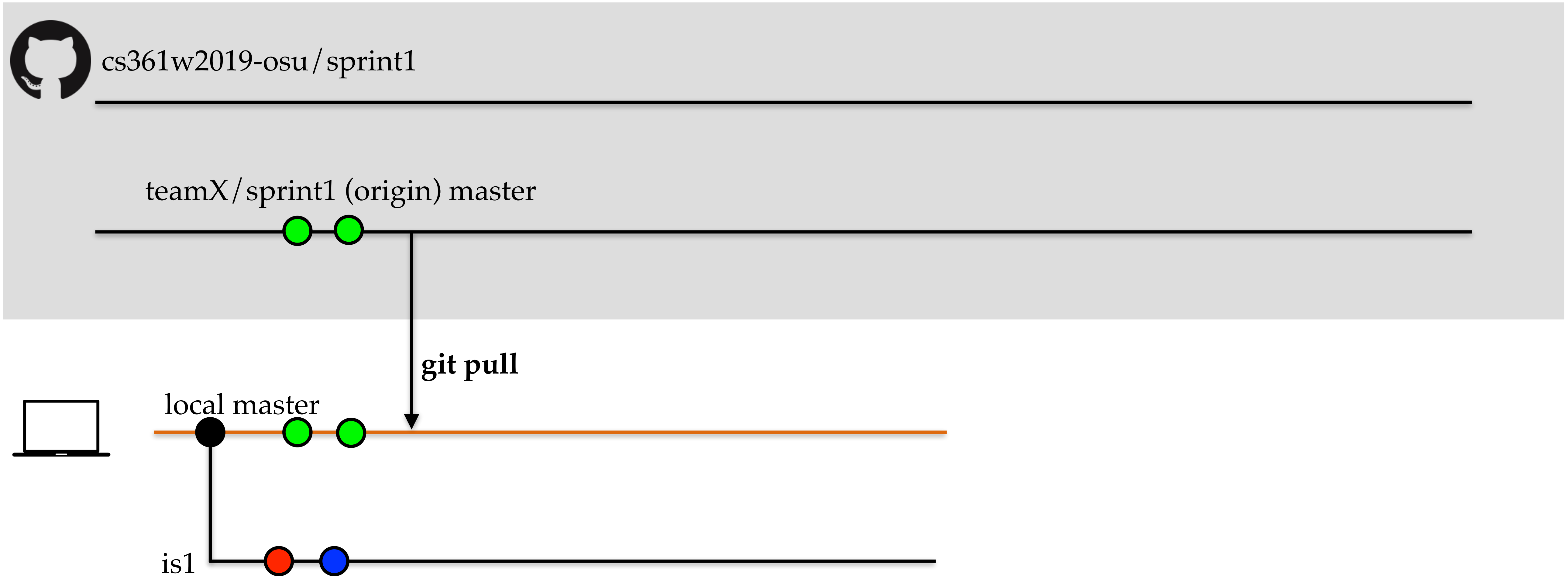
```
git add
git commit
```

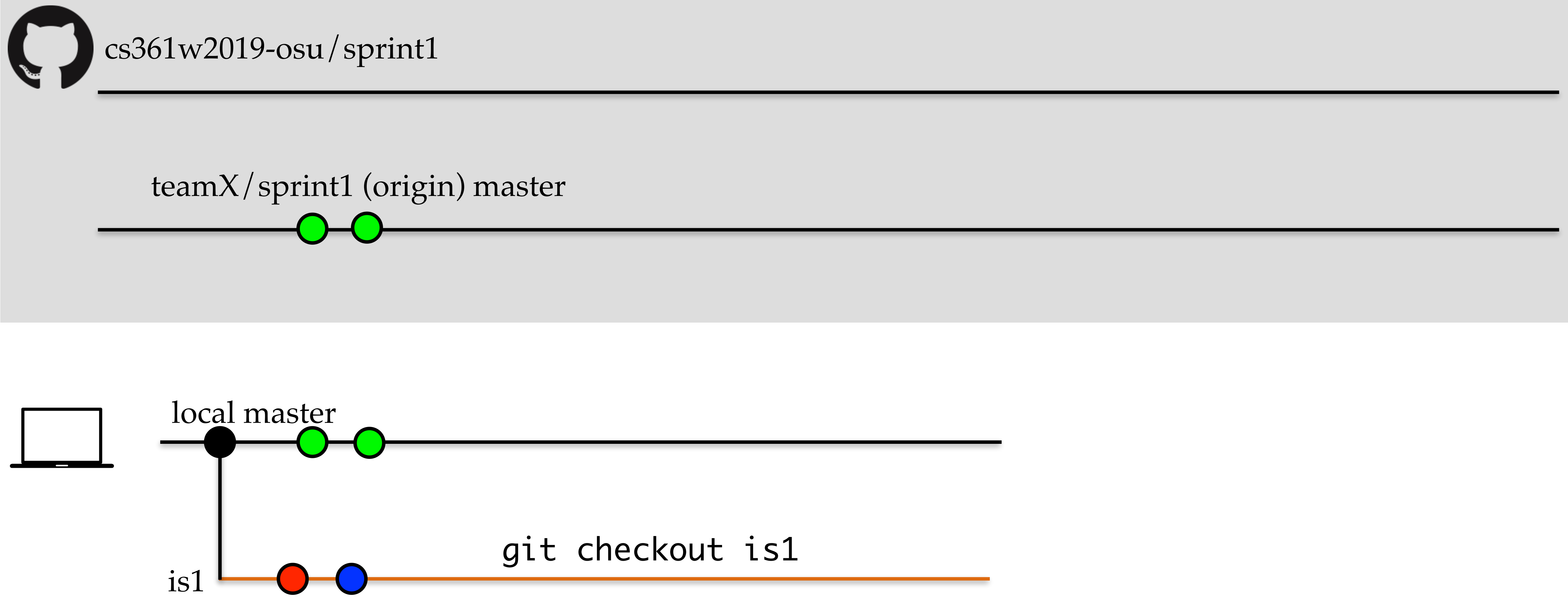
Commit early and often

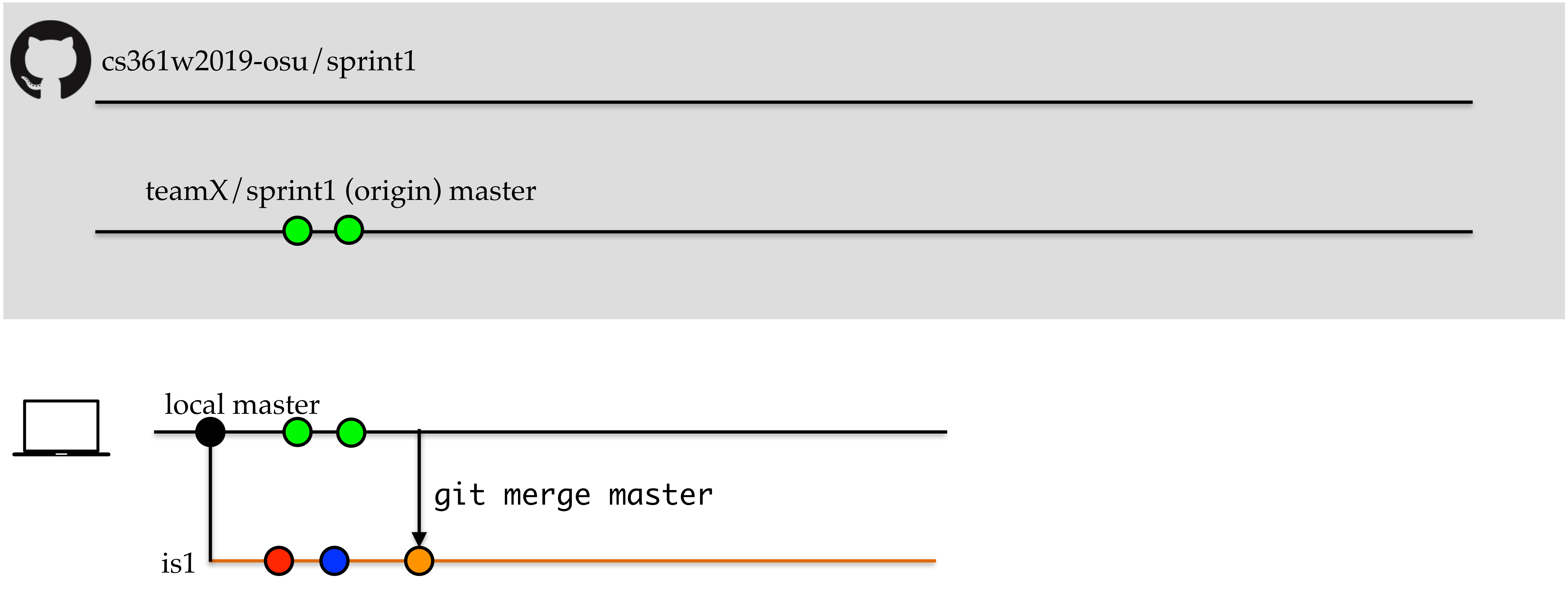

cs361w2019-osu/sprint1
teamX/sprint1 (origin) master
local master
is1



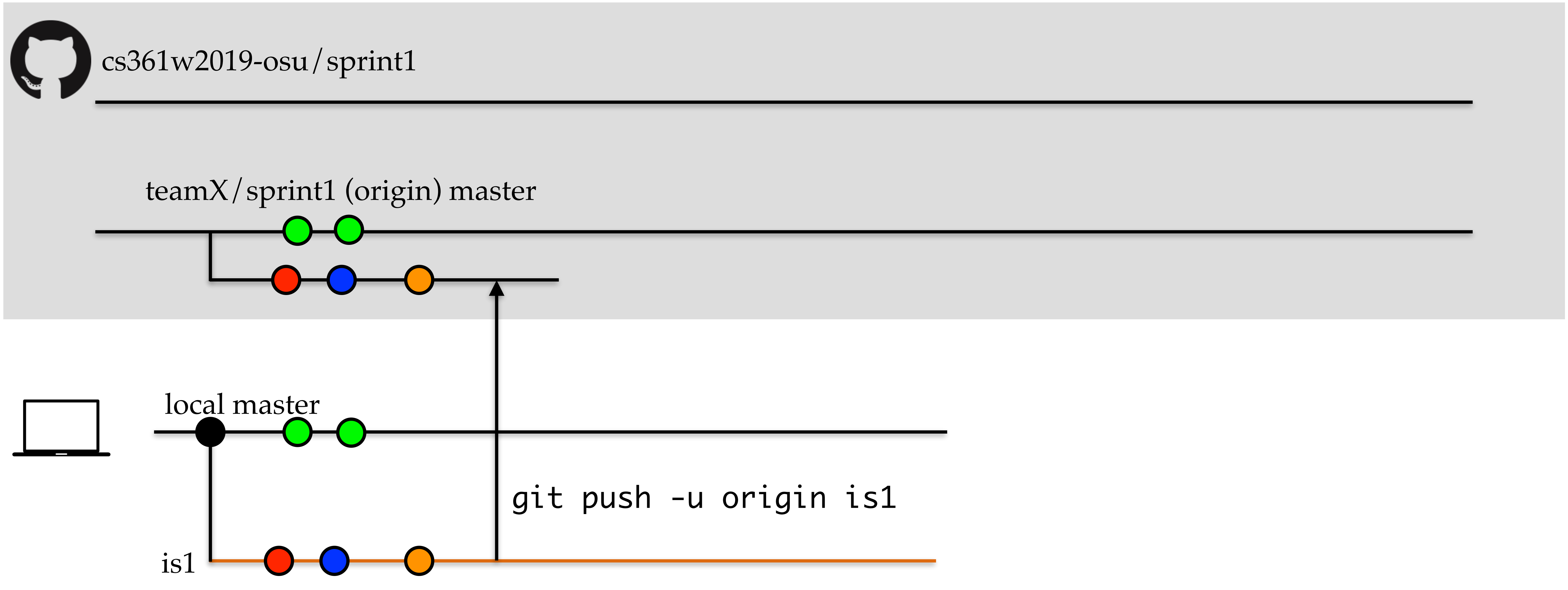
```
git checkout master
```


cs361w2019-osu/sprint1
teamX/sprint1 (origin) master
git pull
local master
is1

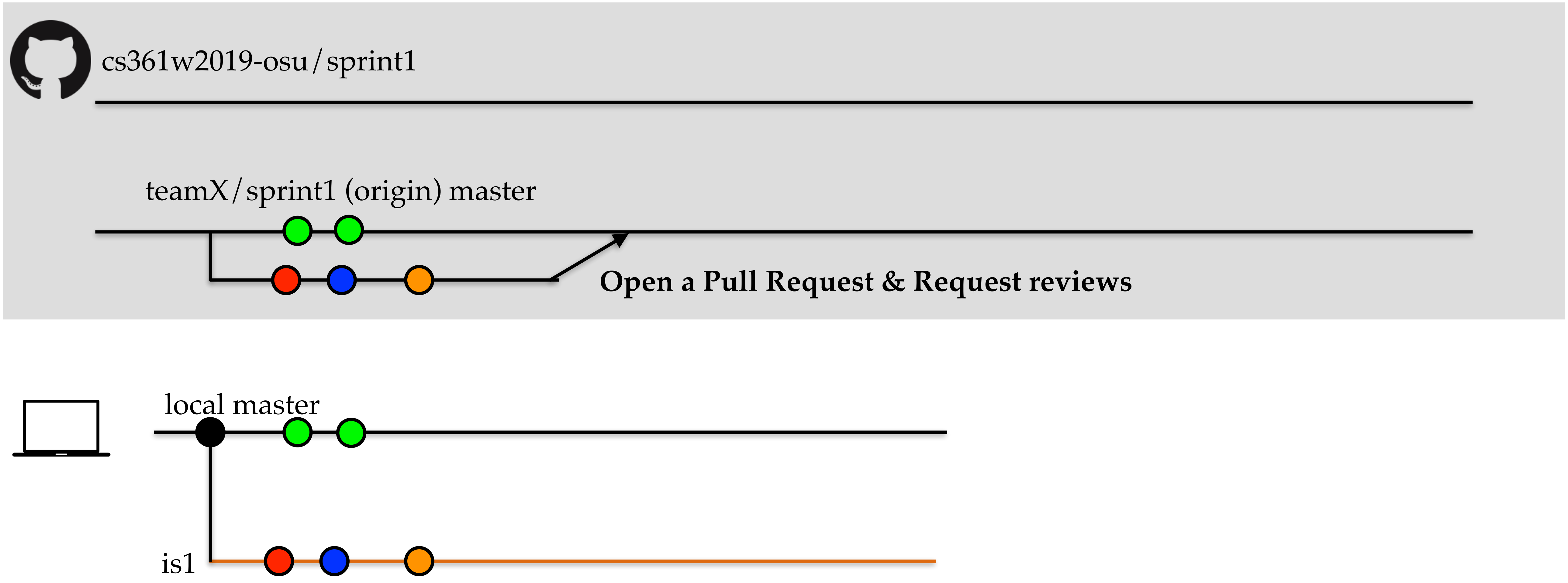

cs361w2019-osu/sprint1
teamX/sprint1 (origin) master
local master
is1
git checkout is1

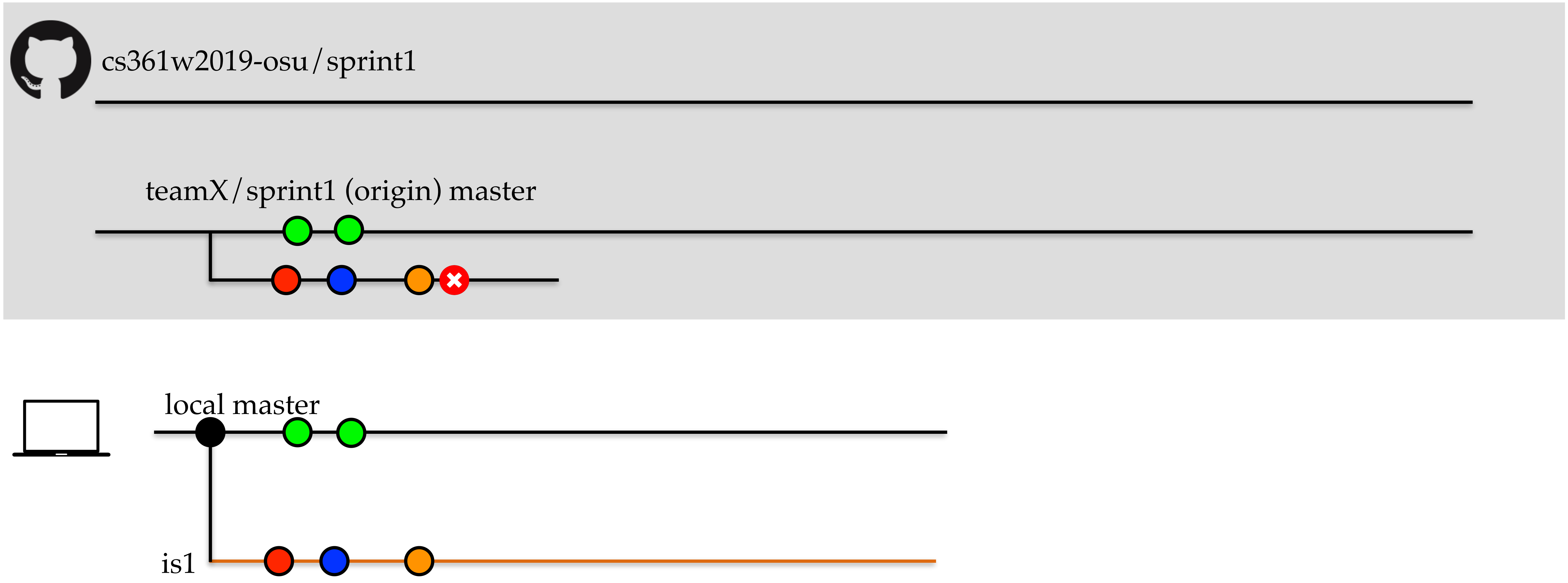


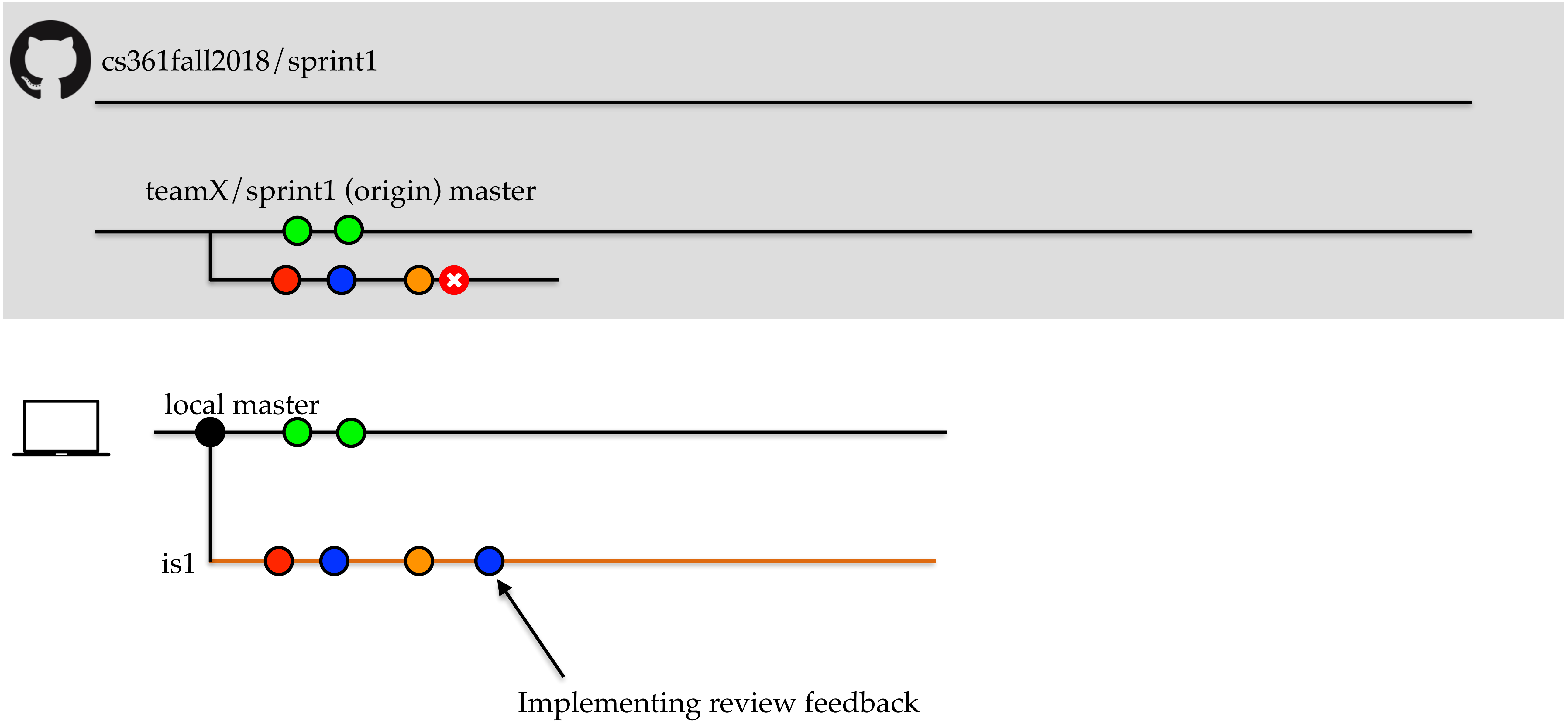
An alternative is to use **`git rebase`**. This will produce the same result, but the history will be cleaner.

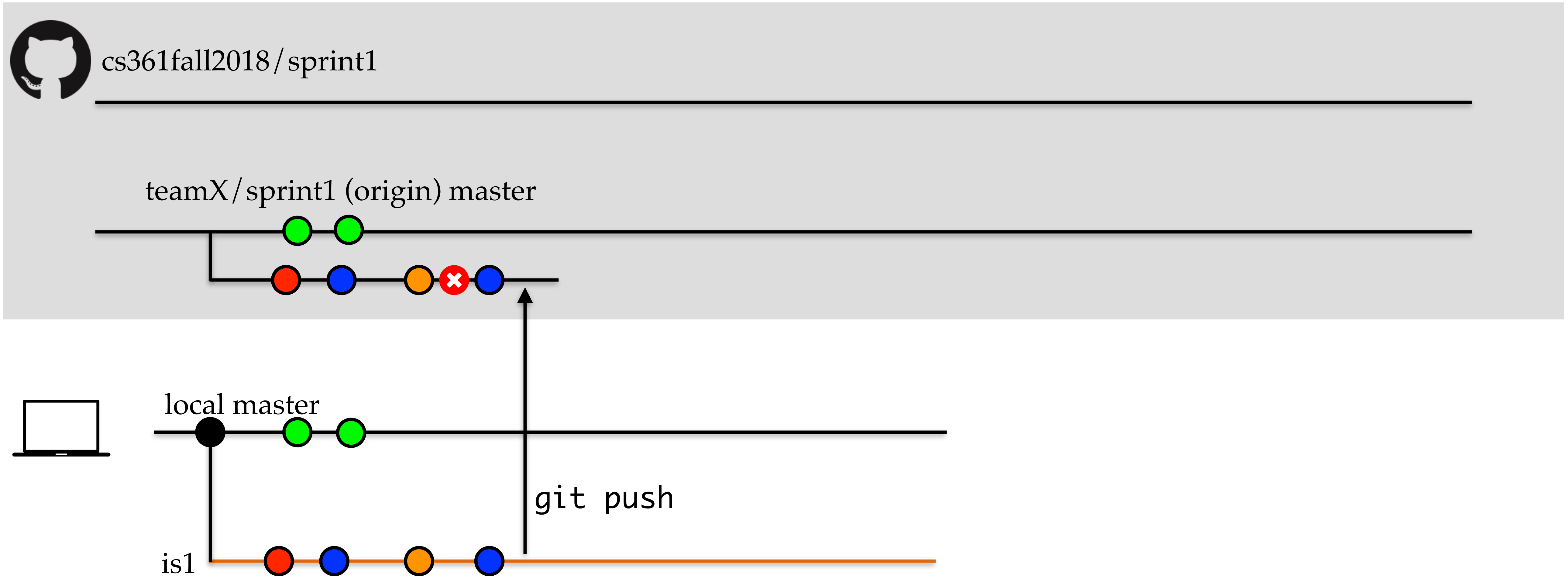


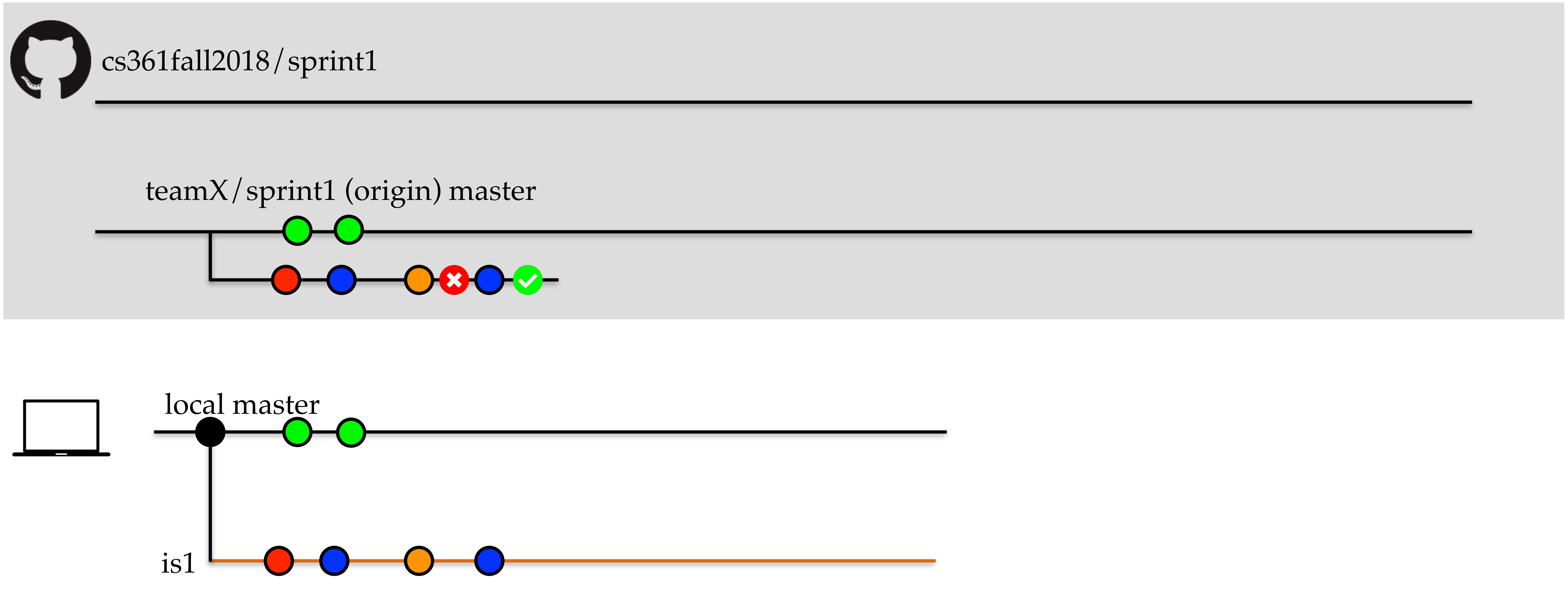
```
git push -u origin is1
```

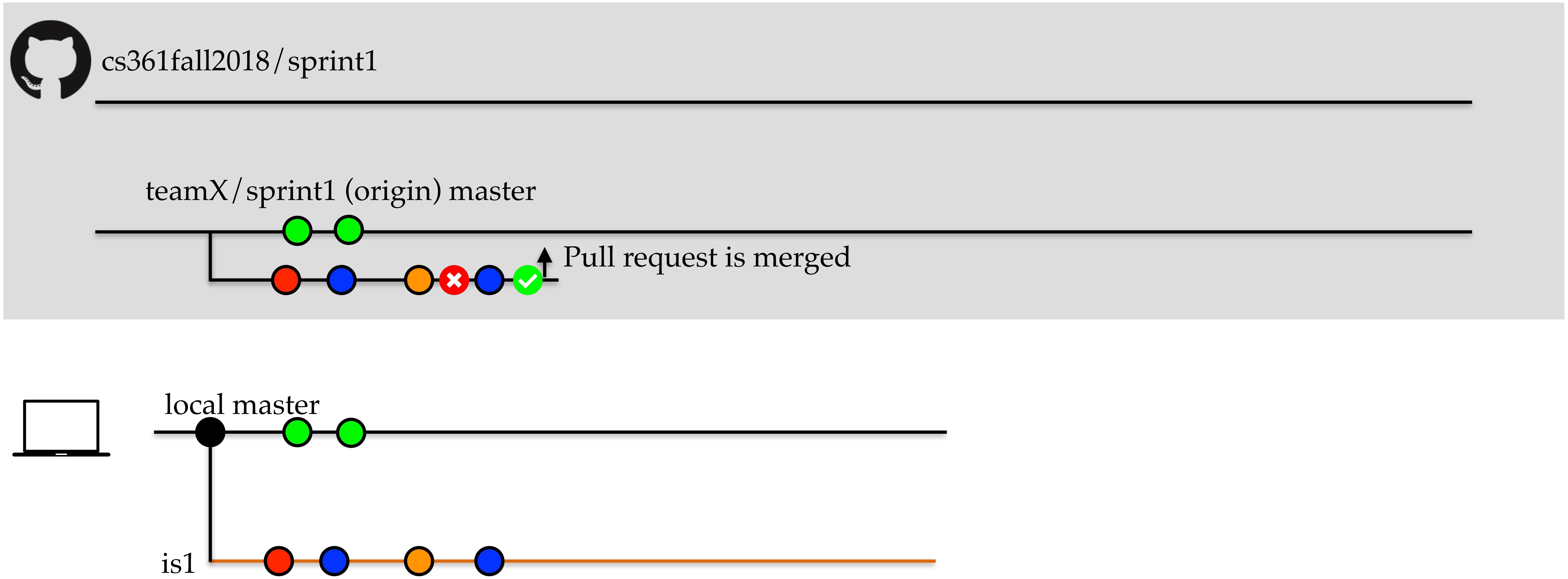

cs361w2019-osu/sprint1
teamX/sprint1 (origin) master
Open a Pull Request & Request reviews
local master
is1

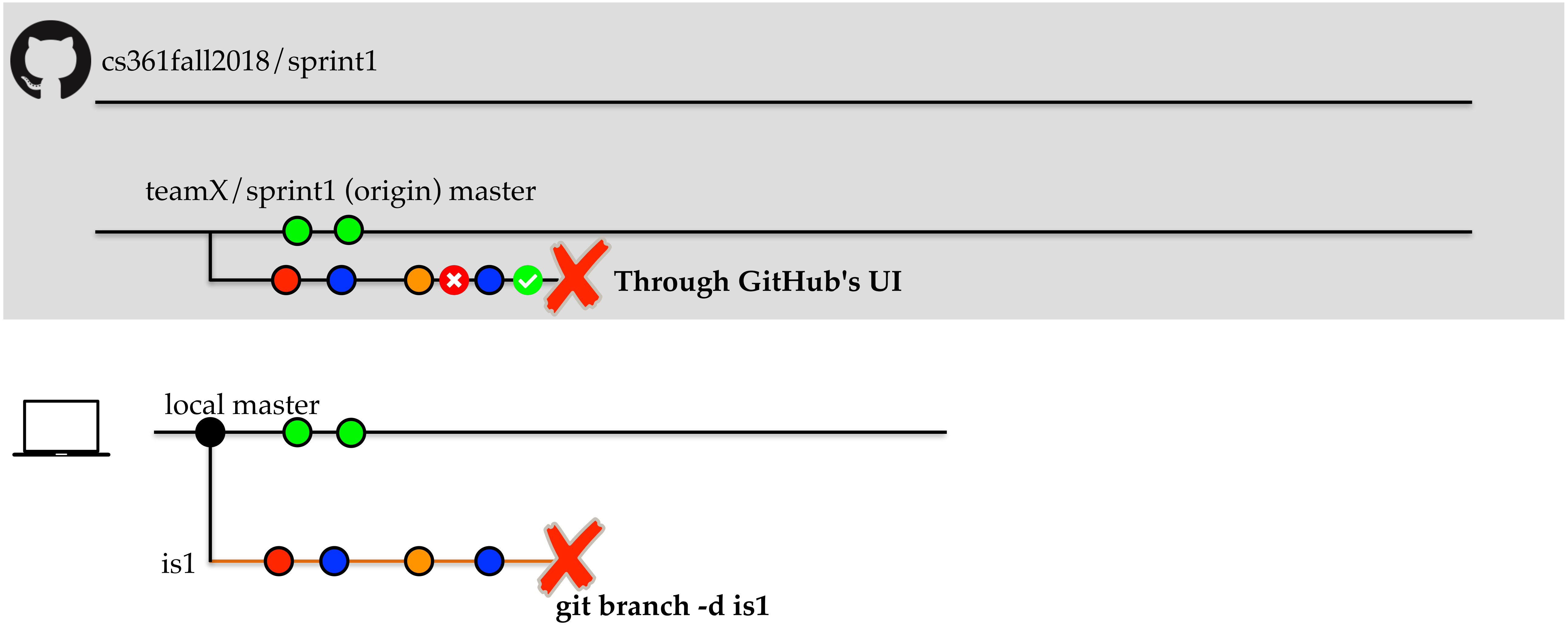

cs361w2019-osu/sprint1
teamX/sprint1 (origin) master
local master
is1


cs361fall2018/sprint1
teamX/sprint1 (origin) master
local master
is1
Implementing review feedback


cs361fall2018/sprint1
teamX/sprint1 (origin) master
local master
git push
is1


cs361fall2018/sprint1
teamX/sprint1 (origin) master
local master
is1


cs361fall2018/sprint1
teamX/sprint1 (origin) master
Pull request is merged
local master
is1



Oregon State University

Proper collaboration will be part of all project sprints

Test Driven Development

What is TDD?

A software development cycle that is based on 3 simple rules:

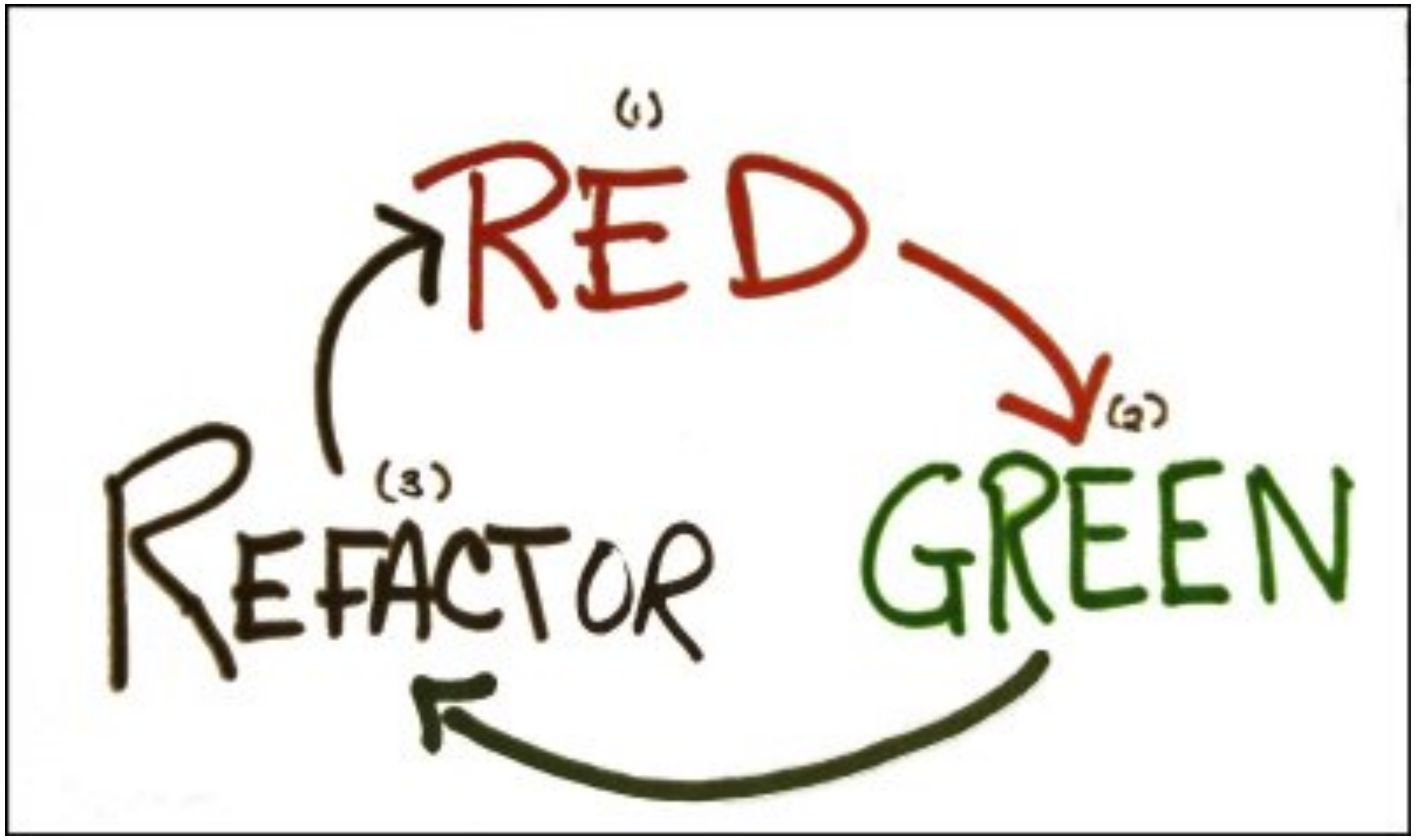
1. You must write a failing test before you write any production code

2. You must not write more of a test than is sufficient to fail, or fail to compile

3. You must not write more production code than is sufficient to make the currently failing test pass

https://blog.cleancoder.com/uncle-bob/2014/12/17/TheCyclesOfTDD.html

TDD cycle



Why TDD?

The act of writing a unit test is more an act of design than of verification.

It is also more an act of documentation than of verification.

The act of writing a unit test closes a remarkable number of feedback loops, the least of which is the one pertaining to verification of function.

-Robert C. Martin (Uncle Bob) in Agile Software Development, Principles, Patterns, and Practices

Advantages of TDD

Clear place to start

Less code thrown away

Less hassle with I/O

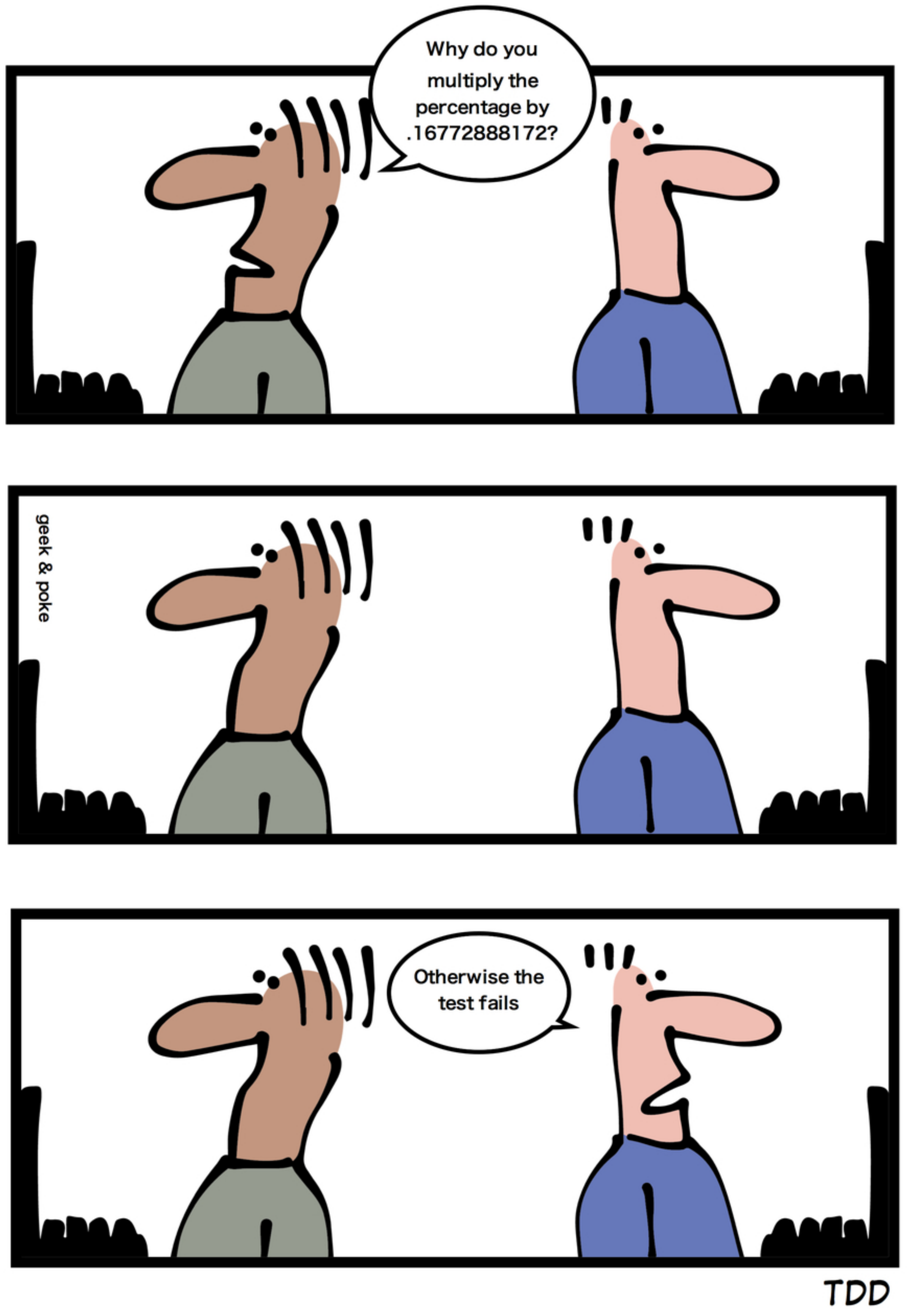
Less fear

Downsides of TDD

Reliant on tests running quickly

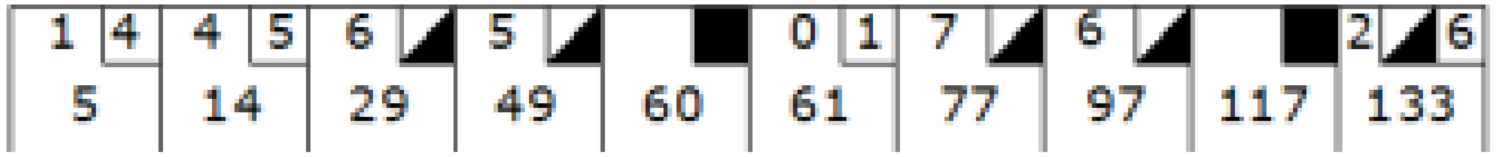
No overarching design

Tests require extra maintenance effort


Why do you multiply the percentage by .16772888172?
geek & poke
Otherwise the test fails
TDD

TDD Demo

Scoring Bowling



1 4	4 5	6 ◢	5 ◢	■	0 1	7 ◢	6 ◢	■	2 ◢ 6
5	14	29	49	60	61	77	97	117	133

The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3 above, the score is 10 (the total number knocked down) plus a bonus of 5 (the number of pins knocked down on the next roll).

A strike is when the player knocks down all 10 pins on his first try. The bonus for that frame is the value of the next two balls rolled.
In the tenth frame a player who rolls a spare or strike is allowed to roll the extra balls to complete the frame. However no more than three balls can be rolled in tenth frame.