

[320] Reproducibility 2: Version Control (git)

Yiyin Shen

Reproducibility

Big question: *will my program run on someone else's computer?*

Things to match:

1

Hardware

← a program must fit the CPU;
`python.exe` will do this, so
`program.py` won't have to

2

Operating System

← we'll use Ubuntu Linux on
virtual machines in the cloud

3

Dependencies

← today: versioning

Dependency Versions

program.py

```
import os, sys, json
import pandas

import pandas

print("Pandas Version:", pandas.__version__)

# code that uses pandas
```

this program "depends" on pandas



you can check a
module version



behavior depends on which release was installed

```
pip install pandas
```

or

```
pip install pandas==0.25.1
```

or

```
pip install pandas==0.24.0
```

or...

Versioning: motivation and basic concepts

Many tools auto-track history (e.g., Google Docs)

The screenshot displays a Google Docs document with a version history sidebar on the right. The document text includes several paragraphs about productivity and procrastination. Handwritten red annotations are present: 'what changed' with arrows pointing to the first two paragraphs, 'when it changed' pointing to the March 4, 6:35 AM version, and 'who changed it' pointing to the March 2, 7:45 AM version.

Document Content:

I am so grateful that I get to write for a living. I also really, really, don't want to start writing right now.

That's more- or- less my constant mindset. When I manage to get started ~~I can~~ I get a lot done, but I rarely find myself in the mindset where I *want* to get started ~~on something that I know will take a lot of time or effort~~. This leads to me falling back into the ~~dopamine-rich~~ dopamine-rich environment called "internet," where algorithmically designed distractions devour time until it's 5 o'clock and oh well I'll seize the day tomorrow.

You've been there. We've all been there. ~~There's a Thing you should be doing but for some reason just can't get started on. Maybe the Thing is setting up a website. Maybe the Thing is a coding project you've been putting off. Maybe the Thing is a book you've intended to write. Whatever the Thing is, you just can't get started. And it wouldn't happen if we could only get started. I can relate.~~

Which is why over time I've found ways to force the issue on myself. Here are a few tricks I, and a few of my co-workers, use to start doing a thing, even when we really, really don't want to do the ~~t~~Thing. ~~In other words, how to motivate yourself to start a task when you don't feel motivated.~~

~~## Use Your Calendar to Force You to Get Started~~ ~~Plan Your Day Around Doing The Thing~~

Every workday morning, after breakfast, I plan my day. I look at my to do list, my inbox, and my calendar, ~~and~~ then figure out how I'm going to use my unscheduled time in order to accomplish what needs accomplishing. I then allocate time for each task on my calendar.

This does two things. First: it forces me to see my time as a resource I have to allocate. Second, adding things to my calendar means notifications on my phone and computer throughout the day, reminding me of the intention I set for myself. It's amazing how that ~~reminder~~ little bit of accountability can keep me motivated. The calendar helps you make the most of the time you have available each day. From author Marc Levy, *[If Only It Were True]*(<https://www.amazon.com/Only-Were-True-Marc-Levy/dp/0743276841>):

Version History:

- Only show named versions (toggle)
- THIS MONTH
 - March 4, 9:10 PM (Melanie Pinola)
 - March 4, 6:35 AM (Justin Pot)
 - March 2, 7:45 AM (Melanie Pinola)
 - March 1, 3:07 PM (Melanie Pinola, Justin Pot)
 - March 1, 10:55 AM (Justin Pot)
- FEBRUARY
 - February 28, 3:35 PM (Justin Pot)
 - February 28, 12:54 PM (Justin Pot)
 - February 28, 11:53 AM** (Melanie Pinola, Justin Pot)

Version Control Systems (VCS)

Useful for many kinds of projects

- code, papers, websites, etc
- manages all files for same project (maybe thousands) in a repository

Explicit snapshots/checkpoints, called **commits**

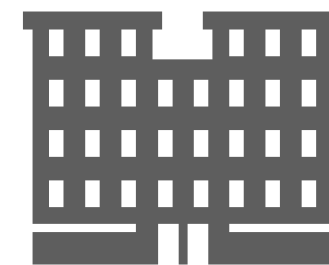
- users **manually** run commands to preserve good versions

Explicit **commit messages**

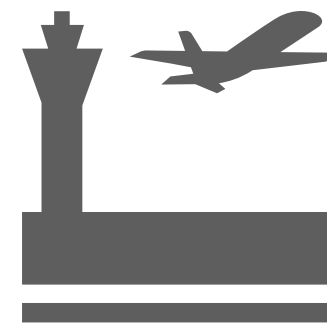
- who, what, when, **why**

Work can **branch** out and be **merged** back

- people can work offline
- can get feedback before merging
- humans need to resolve **conflicts** when versions being merged are too different



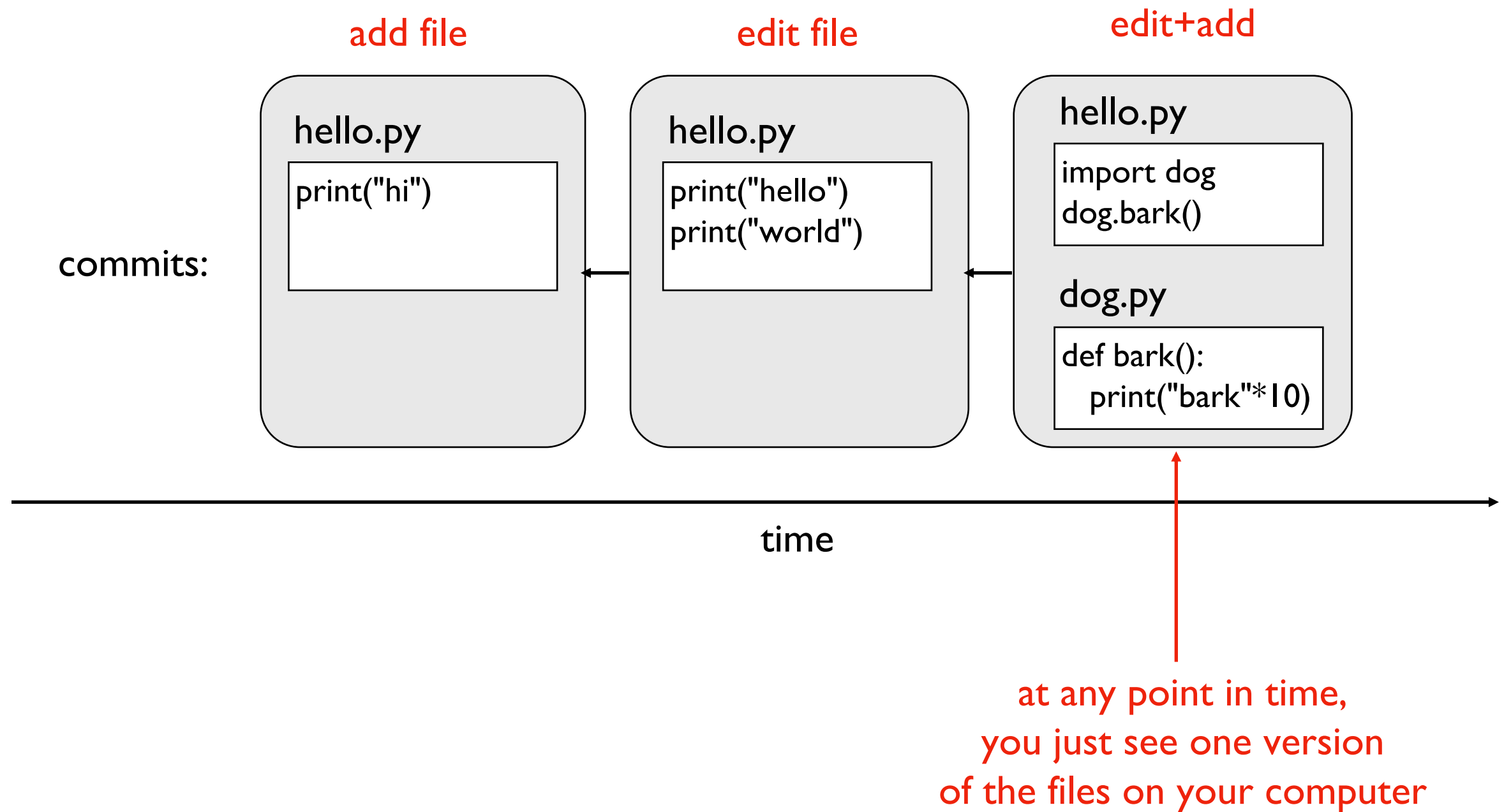
partner A working
on `hw.py` at school



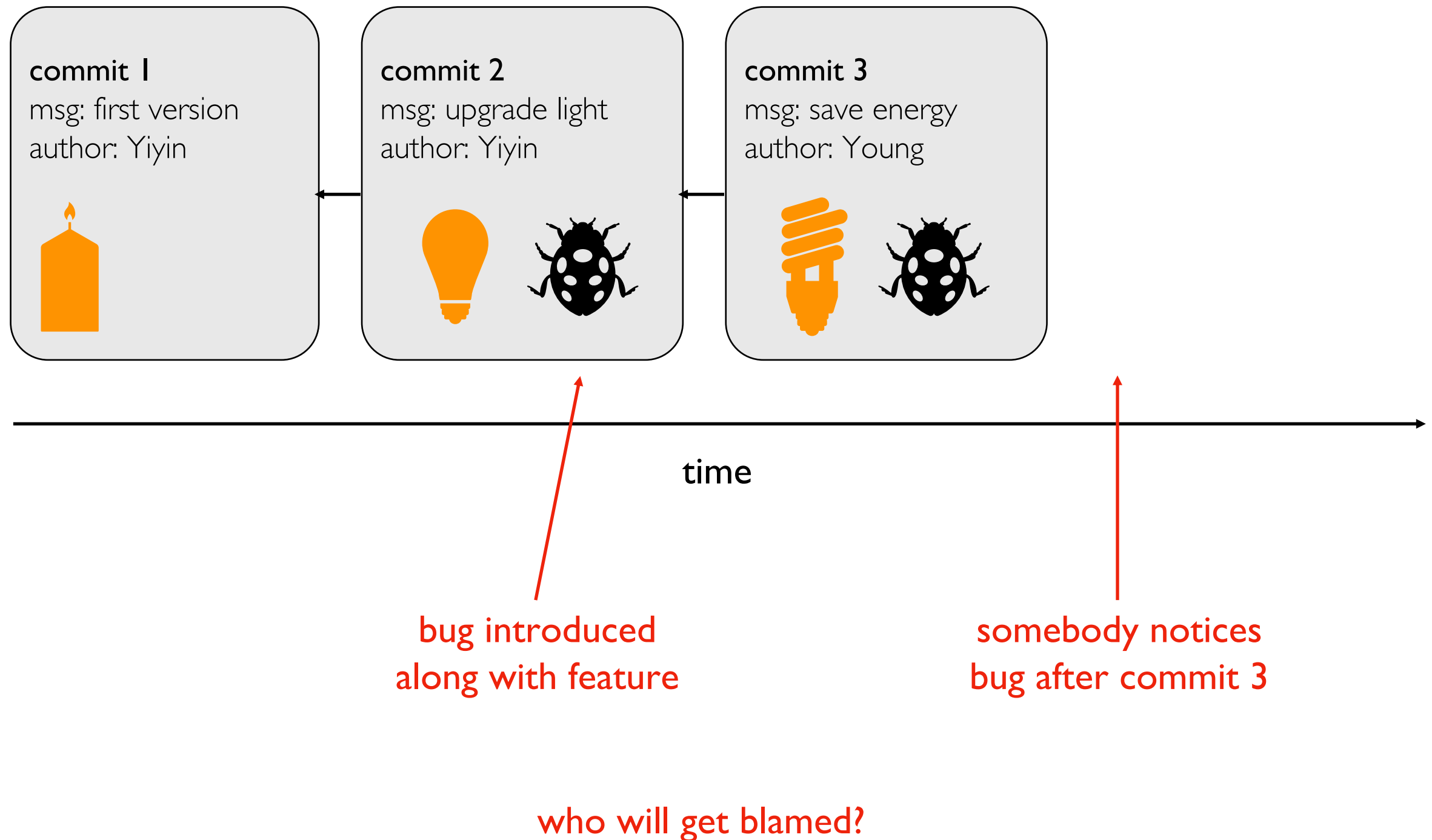
partner B also
working on `hw.py`,
without wifi

what happens when the plane lands?

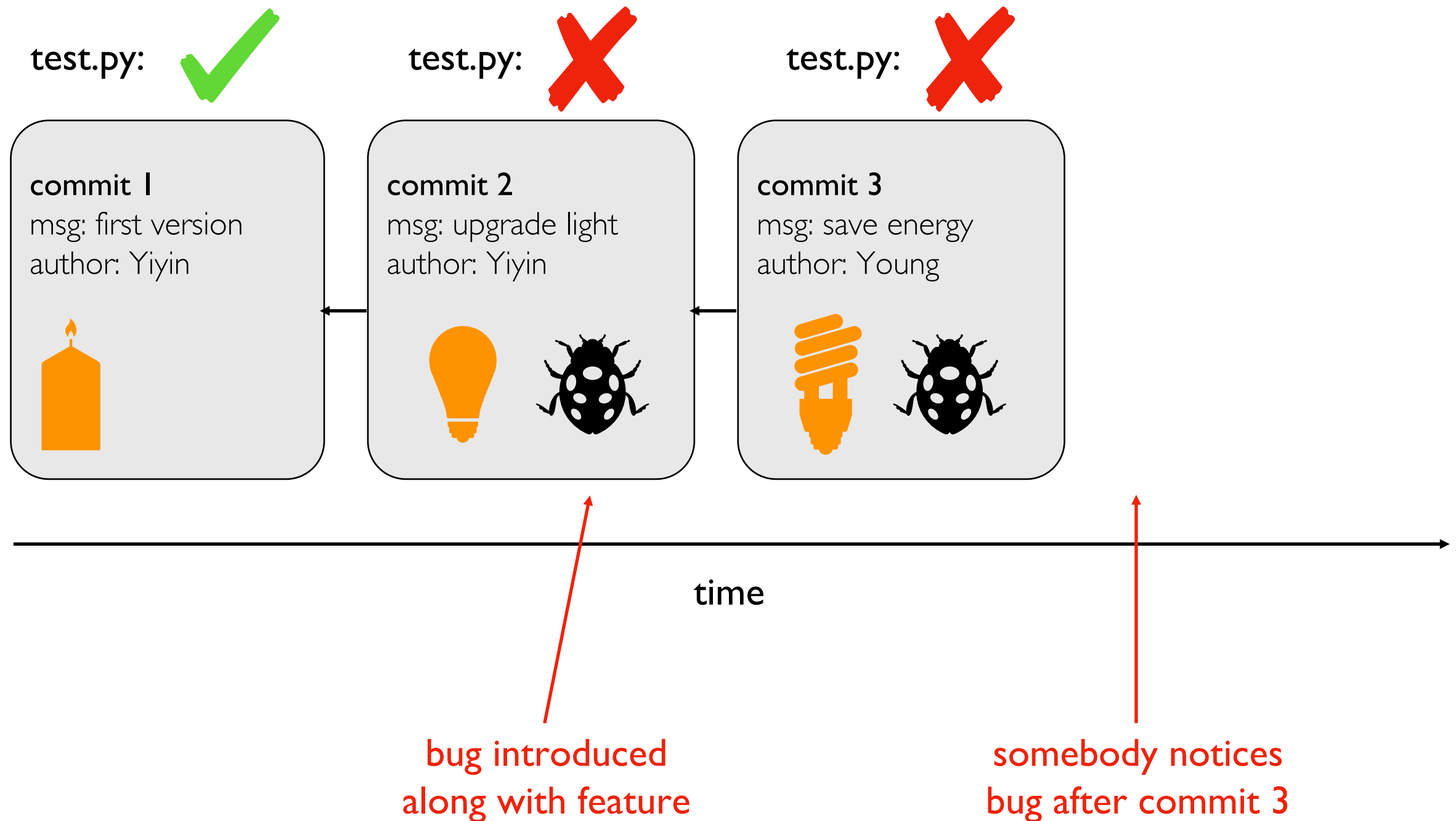
Example



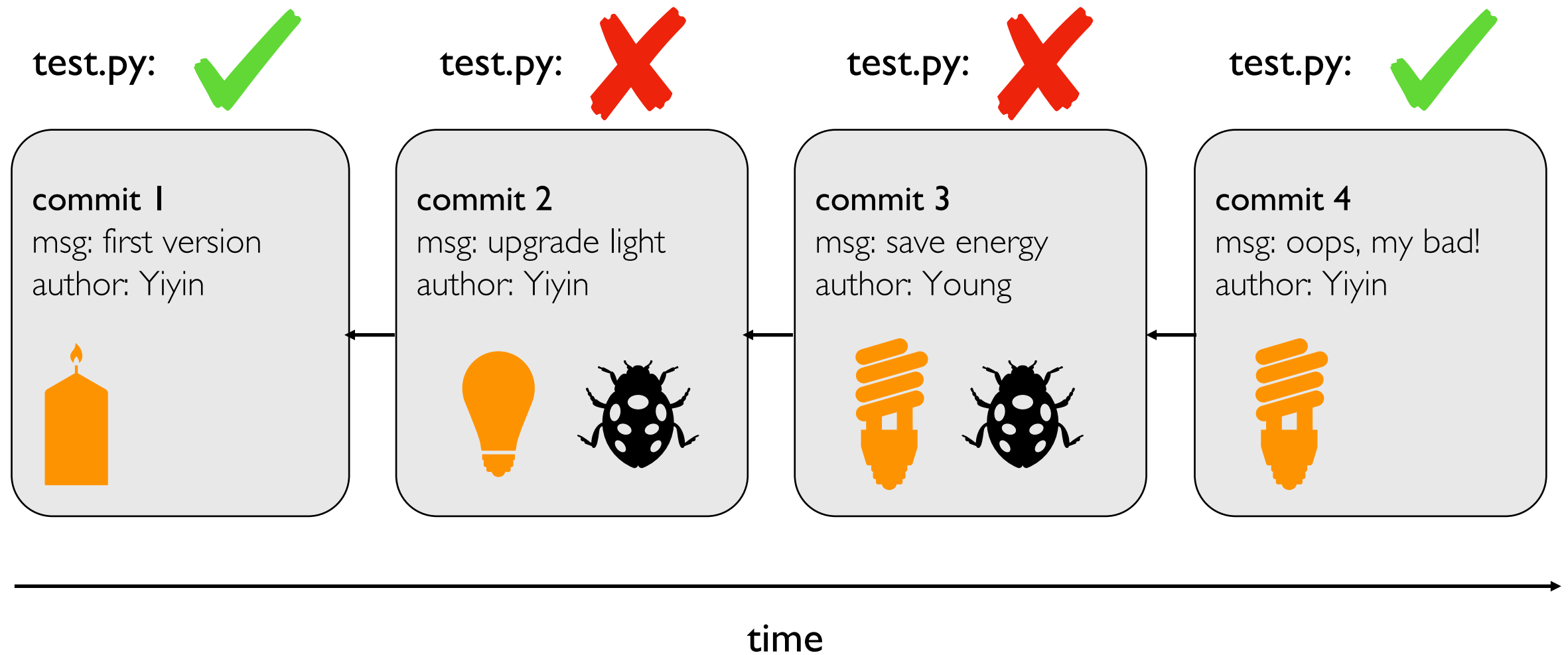
Use case 1: troubleshooting discovered bug



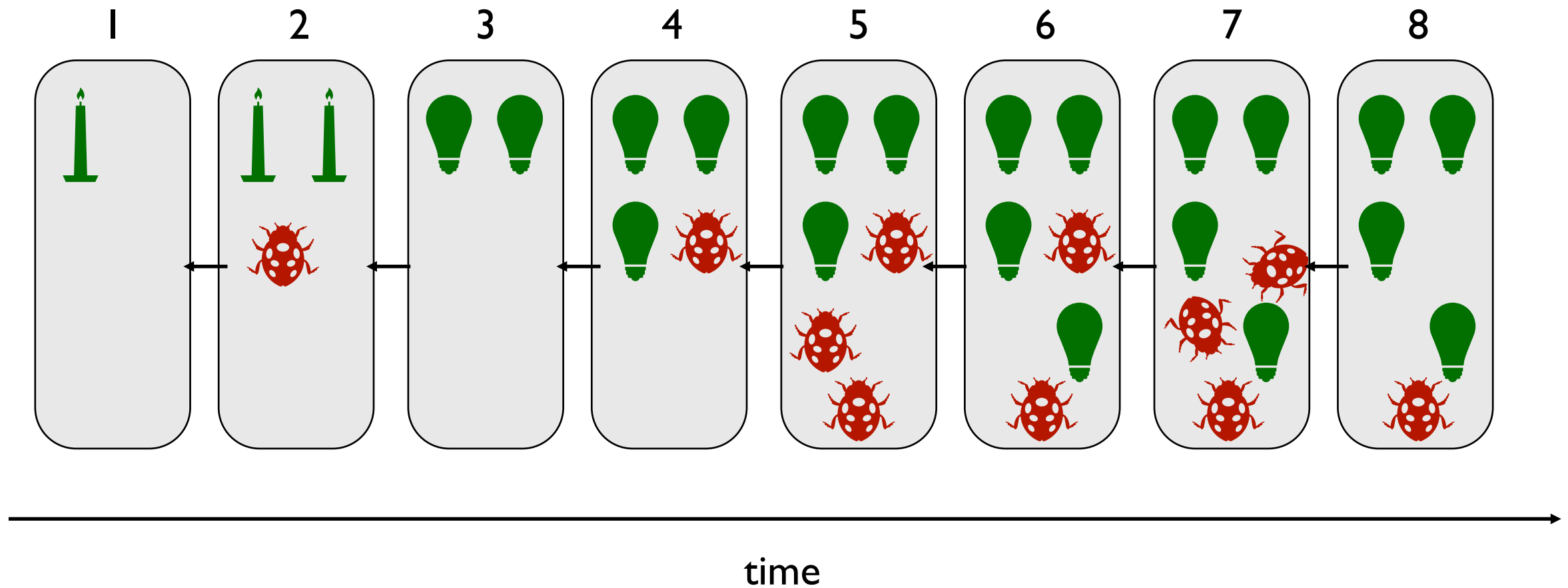
Use case 1: troubleshooting discovered bug



Use case 1: troubleshooting discovered bug

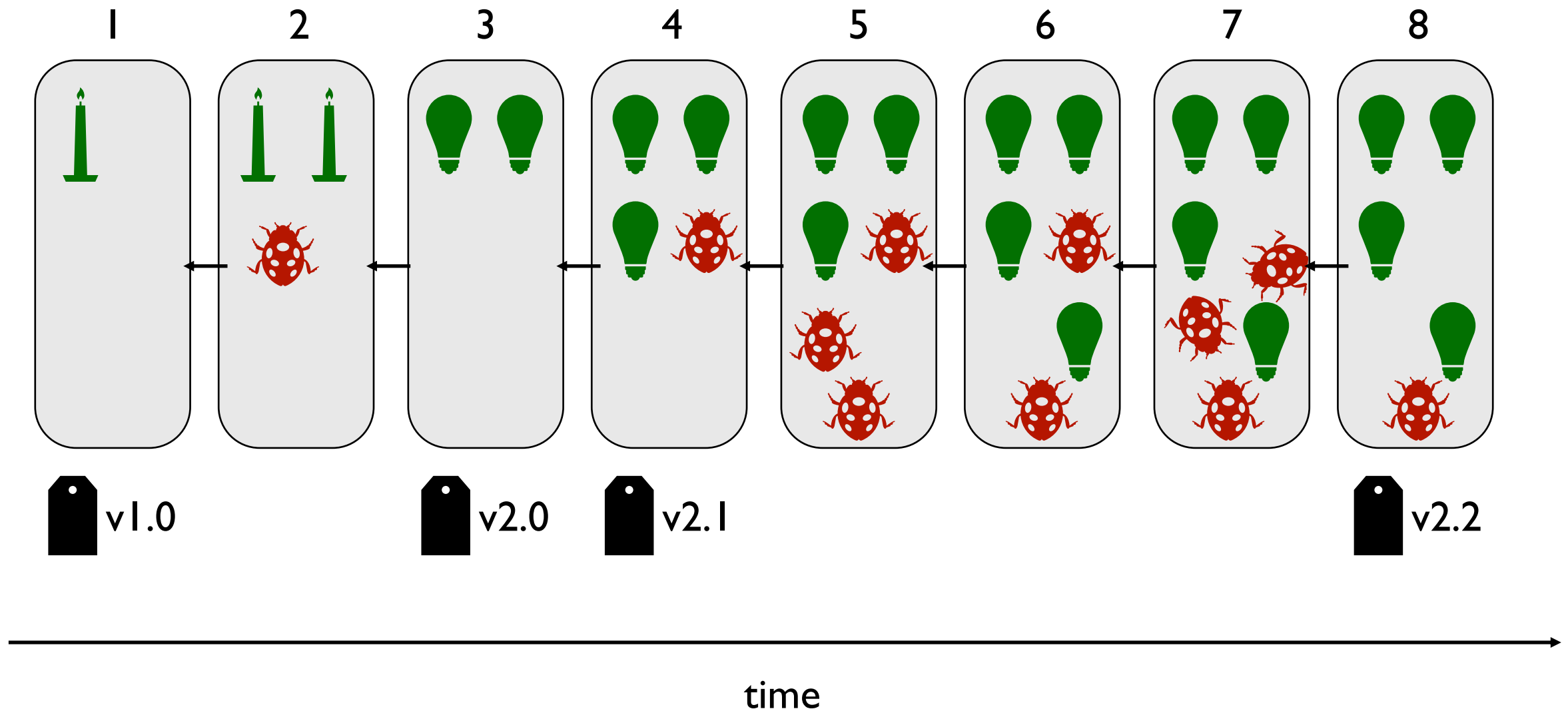


Use case 2: versioned releases



which version would you use?

Use case 2: versioned releases

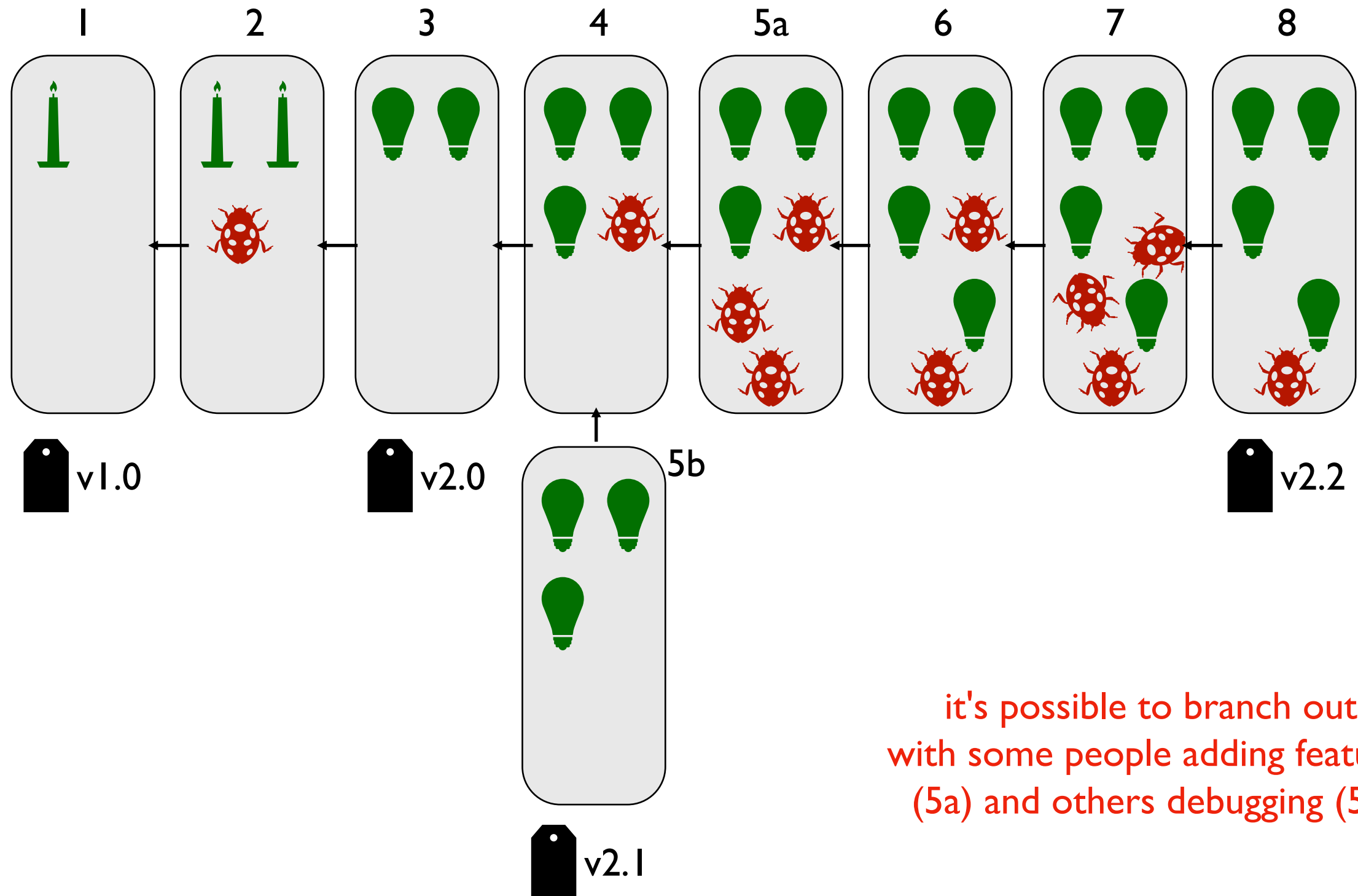


tag "good" commits to create releases

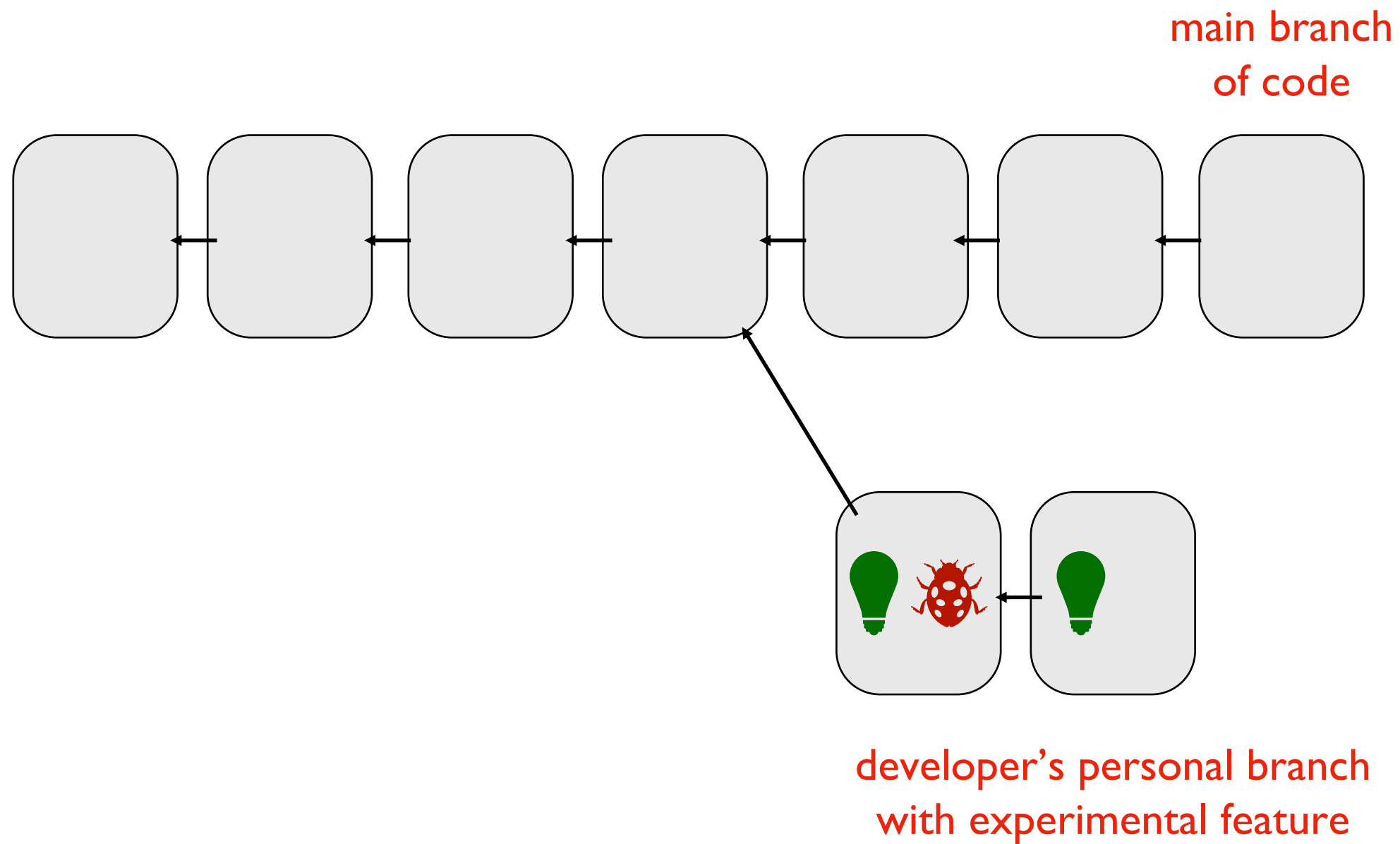
<https://pypi.org/project/pandas/#history>

<https://github.com/pandas-dev/pandas/releases>

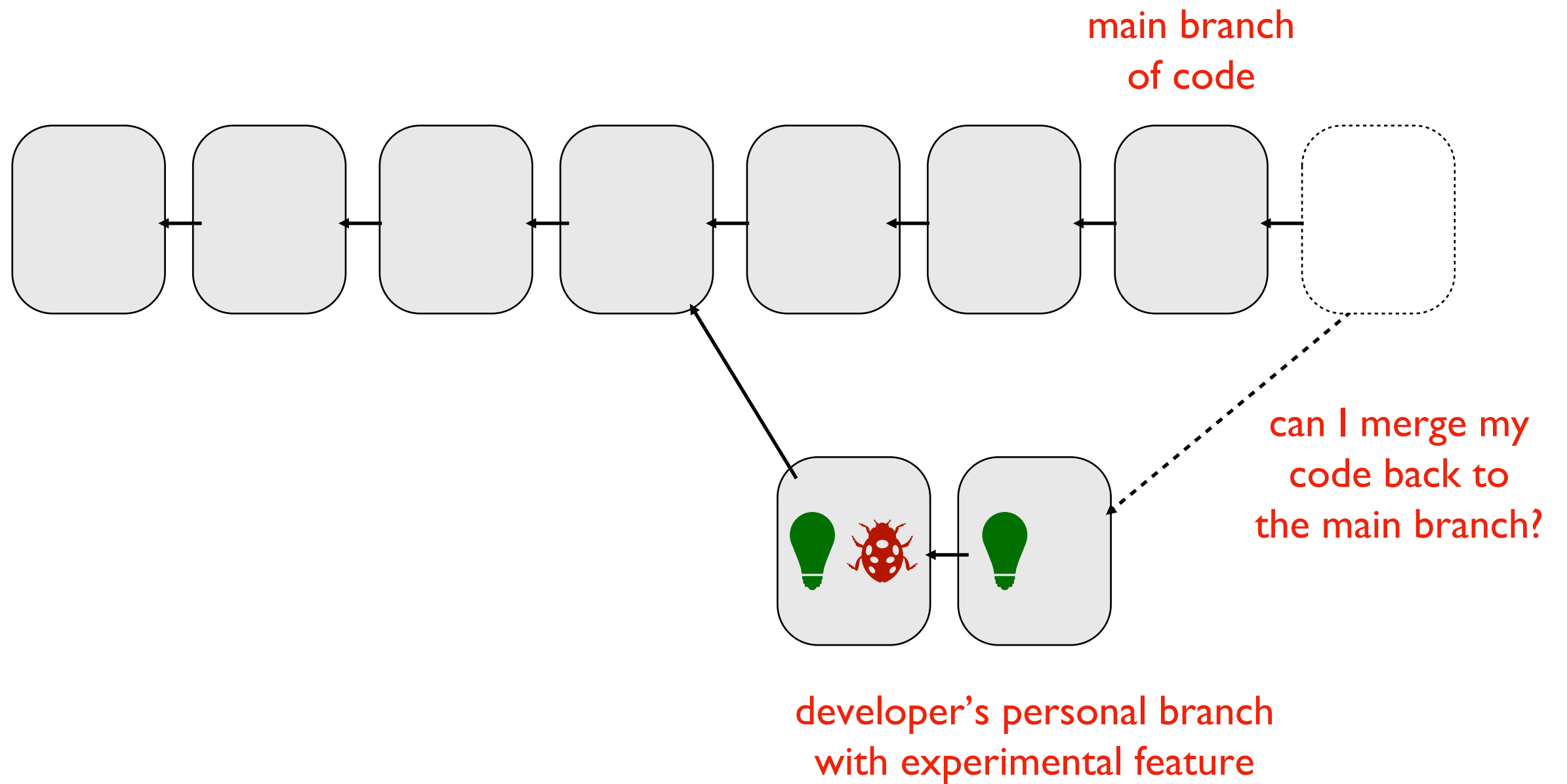
Use case 2: versioned releases



Use case 3: feedback



Use case 3: feedback



git

Version Control System Tools

tools

svn

git

Mercurial

TeamFoundation

git providers

GitLab

BitBucket

GitHub: signup for a free account for next weeks lab

- **do** choose a name that won't embarrass you on a resume
- **do not** post course work



Linus Torvalds
developed git to manage
Linux as a
BitKeeper replacement

Git Demos

Lecture notes/readings/past exams repository:

<https://github.com/yiyins2/CS320-FA23-lecture-notes>

Projects/labs repository:

<https://github.com/yiyins2/CS320-FA23>

Activities:

- Connect to VM through SSH
- Clone a GitHub repo to the VM
- View repo history
- Switch between commits (versions)
- Write new commits

Git Demos

Connect to VM:

- Mac: terminal; Windows: powershell
- `ssh username@computer`: connect to a VM via SSH

Shortcuts:

- `^D` exit connection
- `^C` terminate the current command
- `^R` search history

Linux command line:

- `pwd` display current working directory
- `cd folder` go down a directory
- `cd ..` go up a directory
- `ls` list all files in the directory
- `cat` display the files

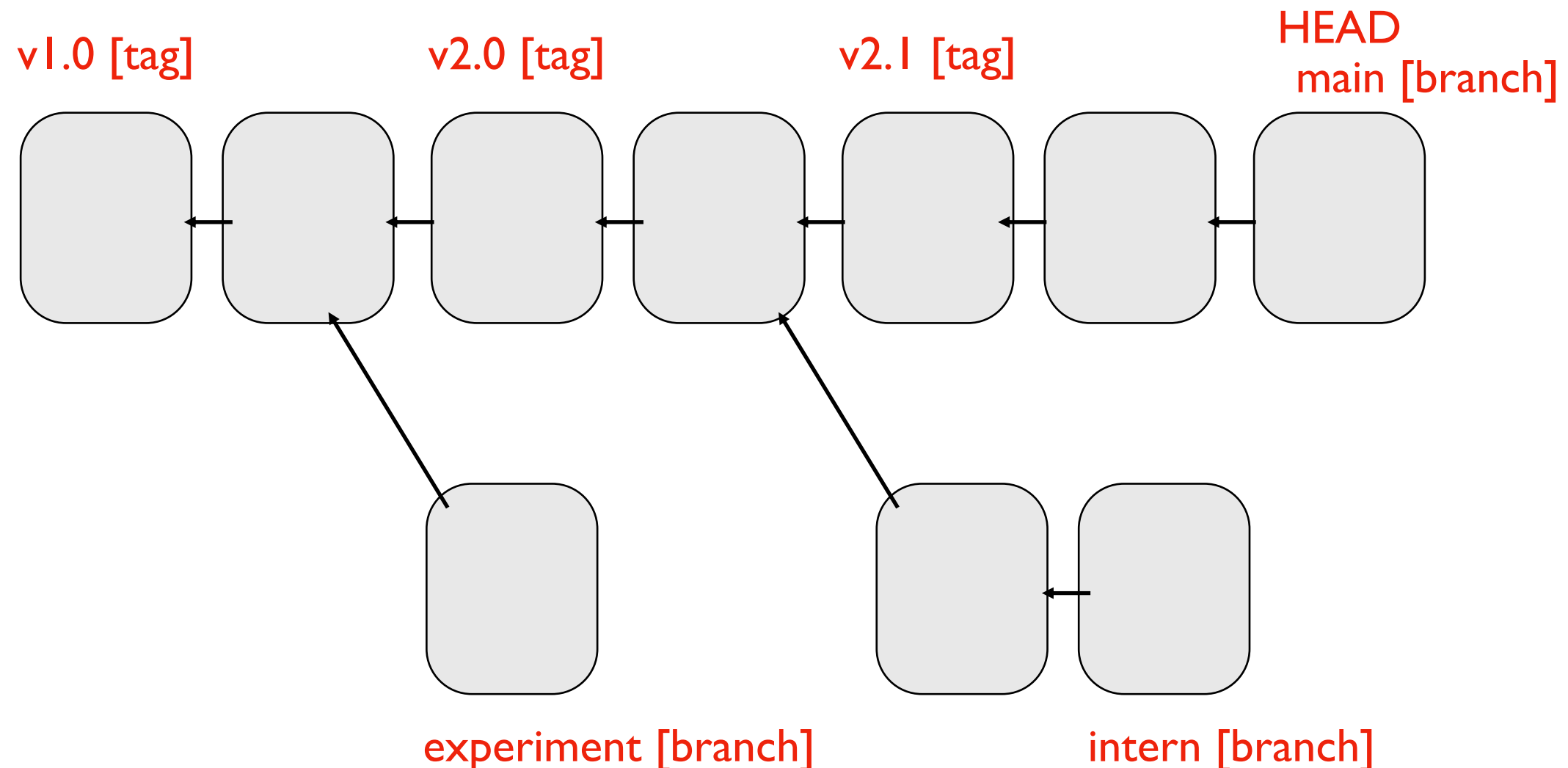
Git Demos

Git Commands:

- git clone: retrieve an entire repository from a hosted location via URL
- git log: show all commits in the current branch's history
- git status: show modified files in working directory, staged for your next commit
- git pull: fetch and merge any commits from the tracking remote branch
- git add: add a file as it looks now to your next commit (stage)
- git commit: commit your staged content as a new commit snapshot
- git push: transmit local branch commits to the remote repository branch
- git branch: list your branches. a * will appear next to the currently active branch
- git checkout: switch to another branch and check it out into your working directory

HEAD, Branches, and Tags

Remembering commit numbers is a pain! Various kinds of labels can serve as easy-to-remember aliases



HEAD: wherever you currently are (only one of these)

tag: label tied to a specific commit number

branch: label tied to end of chain (moves upon new commits)