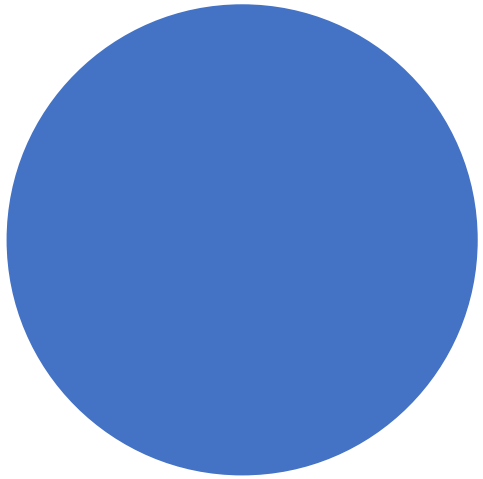
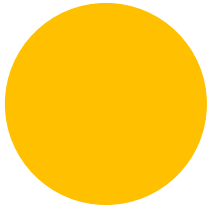
Napkin Drawing Advice: It's NOT an architecture diagram

Why would someone use what you are building?

What does the system do? Use verbs. "Find" is much better than "Search"

Who is the system for? Show how users interact with your system

Open-Source Software: Governance and Technologies

Examples using Git and GitHub

From Previous Lecture: Build on Foundations

Network Messaging, Messaging Patterns, Protocols

Algorithms

Design Patterns

Engineering Practices

Tools

Drill Down: Engineering Practices

Open-source practices: GitHub to Governance

Developing telescoping applications

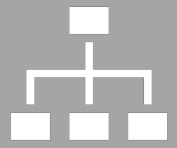
Deploying at scale: continuous integration and deployment

Operating at scale

Open-Source Practices

Shared Project Repositories Are More Important than Your Personal Repository

Show that you can contribute to other people's projects



Or show that you can create and manage a project that other people contribute to.

Your own branch of a code isn't impressive. Contributing back to the main branch of a project is what matters.

Name some open-source projects

Open Source Is Not Just Having Code on GitHub

- Open-source software depends on **communities** of developers
- Communities need **governance** (rules) for how they operate

Apache Software Foundation Governance Examples

- Project members form the Project Management Committee (PMC)
- Major project decisions are made through votes by the PMC
- Votes take place through emails: **a persistent audit trail**
- Most votes are on public mailing lists.
- Voting periods are typically several days

Some Do's of Open-Source Audit Trails

Do you work well with other developers?

Do you effectively and articulately make your case when discussing issues?

Do you make good pull requests and commits?

Do you handle criticisms well?

Do you effectively critique other developers?

What is Git?

Git is a collaborative, distributed version control system.

- Everyone has their own branch of the code in a local repository.
- Each branch has a unique ID
- You can work entirely separately and never give back....

If you want to work collaboratively, you must combine (merge) branches.

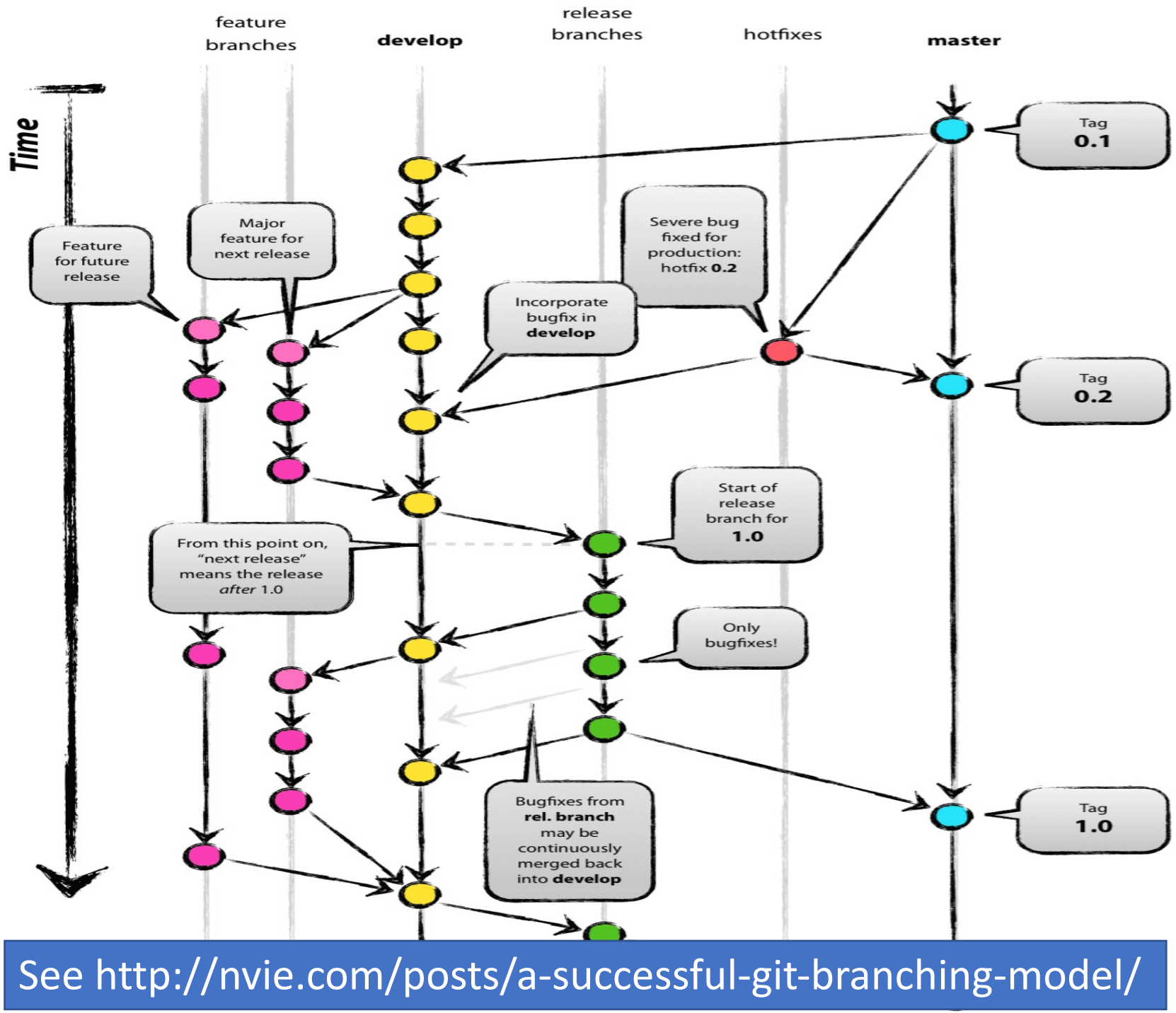
- GitHub provides tools for collaborative coding

Teams need a strategy for how to merge their branches.

- **Communities** choose and follow rules (conventions) on how they use tools: **Governance**

Some Useful Conventions and Principles

Foundations for the Apache Way



See http://nvie.com/posts/a-successful-git-branching-model/

1000 Words about Previous Picture

Branch	Description
Master	All other branches trace back to here. Final releases are here. Must always build and pass all tests.
Develop	Code for next version of Master. Integration Branch. Everyone's code goes back here. Must always build and pass all tests.
Feature	Working branches with code not ready for integration. May have 1 or more developers. Goes away when merged back into Develop.
Release	Code that is preparing to go back to the Master. Only bug fixes.
Hotfix	Code that fixes a bug discovered in Master that must be fixed immediately. Merged back to both Master and Develop branches.

Only the Master and Develop branches live forever!

Applying This for Your Assignments (1/3)

Note Git is not well set-up for microservices

- Ideally you need multiple repositories (each project has its own GitHub organization)
- Since we don’t have this, you need to choose a convention and stick with it
- And document it on your project’s Wiki.

Applying This for Your Assignments (2/3)

Use team-wide Develop branches for each project microservice

- This is tied to the CI/CD system that you use

Each team member can have their own “Feature” branches.

- Make pull requests to merge with the Develop branch.
- **Another team member handles the merge**
- **Communicate any issues using GitHub Issues.**

Apply This to Your Assignments (3/3)

Use named Release branches for submitting assignments.

- ”Project Milestone 1”, “Project Milestone 2”
- This is tied to the CI/CD system that you use.
- This is what you submit for grading

Keep Develop and Release branches separate!

Create new branch for Project Milestone N+1

A Side Note on the Apache Way

The above procedure is just one possible convention.

- It works well with continuous integration while allowing feature development.
- But it could be a lot of overhead for a small team.

Apache projects decide their own branch and merge strategy.

- Decisions are made on the developer discussion list.
- Public votes if necessary
- Conclusions are public

What if this convention doesn't work for the project?

- That gets discussed on the mailing list as well, in public and on the record.
- Public votes are cast if necessary.

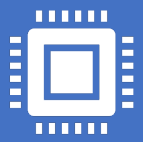
GitHub

What Is GitHub?

A public repository for open-source projects that are managed with Git.

Tools for helping you manage your code and your community.

And more https://guides.github.com/



GitHub also integrates with JIRA and other online tools

Connect a git commit to a JIRA issue.

Using GitHub Issues

See https://guides.github.com/features/issues/ for a full guide.

- See ”Milestones, Labels, and Assignees”

Use this feature to discuss your project.

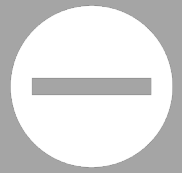
Use code commit comments to tie commits to issues.

- Include the issue number (#xxx) in your commit message.
- See also https://help.github.com/articles/closing-issues-via-commit-messages/

GitHub issues provide an audit trail for your work.

Using GitHub Issues for Assignments

All work must be described using issues.



Each commit is associated with one issue.

But...

I did all this work on my branch, and it didn’t get merged, and someone else on the team did what I was supposed to do, and...

But...

I know it doesn’t look like I did much, but really, I was contributing to all the discussions, and I helped write the code, but all the commits came from my teammates, and...

You Must Have an Audit Trail

Your assignments must work for the graders

Each team member needs to be able to point to their specific contributions

Pull Requests

Notifies others of changes to a common branch, initiate reviews

If you want to contribute to a code branch that you don't have write access to, use a **pull request**.

In Apache projects, submitting pull requests (or patches) is the way to establish yourself with the project community.

Submit enough accepted patches or pull requests and you will be voted into the project.

Using Pull Requests for Assignments

Each team member has her/his own “feature” branch

Use Pull Requests to merge with Develop

Feature branches must be merged back to the Develop branch by another team member.

Use GitHub’s Code Review tool to review

All communications about merging take place using GitHub Issues

Code Release Process

Choose a Release Manager.

Discuss using Issues

The Release Manager creates the Project Milestone N branches from Develop branches.

Everyone votes on the release.

+1 for working, -1 for not working

Fix bugs directly in the release branch

Release Manager manages these pull requests

Release Manager also merges the Release branches back to the Develop branches

Using GitHub Wikis for Documentation

Good code documents itself, but…

https://guides.github.com/features/wikis/

Each project milestone must have a Wiki entry that includes all instructions on how to build and test the assignment.

Your grader will only look at the wiki when setting up and testing your submissions

Some “Apache Way” Lessons

- Community over code.
- Discuss issues publically in an archived, citable manner.
- Volunteer by assigning yourself to public issues.
- Cite the issue(s) associated with each commit.
- Review pull requests for code bases you can’t write to
- Call votes on important decisions