

Using Git and GitHub for Your Project Assignments and Life

Tools for collaborative team coding on open source projects

Q: Isn't this a class about applied distributed systems?

A: This lecture is about how to lay the foundations for a good distributed system application.

Also, Git is a distributed
system

From Lecture 1: Build on Foundations



Network Messaging, Messaging Patterns,
Protocols



Algorithms



Design Patterns



Engineering Practices



Tools

The order is important

Engineering Processes and Practices



Open-source practices: GitHub to Governance



Developing telescoping applications



Deploying at scale: continuous integration and deployment



Operating at scale

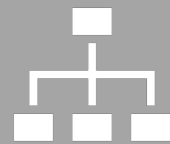
GitHub Is Your Living Resume

- I hire people
- I see lots of resumes.
- They are all the same.
- Demonstrate to me what you can do.

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.

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
- Community members need to follow the rules
- Community members need to act professionally

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
- Most votes are on public mailing lists.
- Voting periods are typically several days

Some Do's of Open Source



Do you work well with other developers?



Do you effectively and articulately make your case when there are disagreements?



Do you make good pull requests and commits?



Do you handle criticisms well?



Do you effectively critique other developers?



Learning Open Source
Practices Is an Important
Part of this Class

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 have to combine (merge) branches.

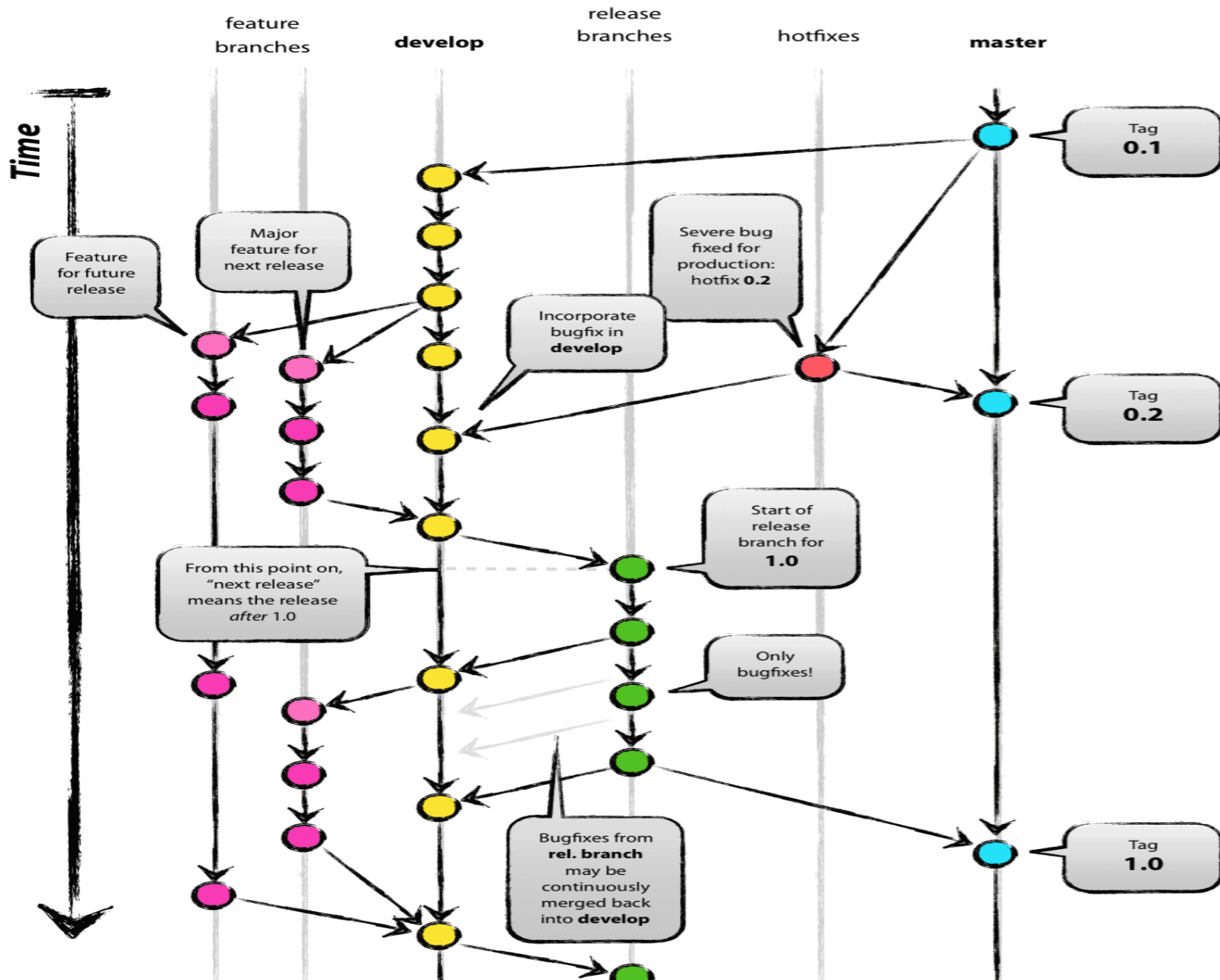
- GitHub provides tools for collaborative coding

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 (1/2)

Each project milestone should be a Release branch.

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

Each team has team-wide Develop branches for each component

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

Each team member can have their own branch.

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

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.

Apply This to Your Assignments (2/2)

One team member serves as release manager

- Choose using GitHub issues (document the selection process)
- Rotate for each milestone
- Release manager creates the Project Milestone N release branches from Develop branches for each project component

After grading, merge the “Project Milestone N” branches back to Develop.

Create new branch(es) 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 code that is managed with Git.

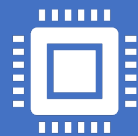


Tools for helping you manage your code and your community.



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.

Code commit comments tie commits to issues.

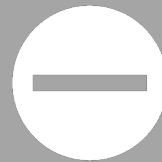
- Include the issue number (#xxx) in your commit message.
- See also <https://help.github.com/articles/closing-issues-via-commit-messages/>

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.



Use Issues to make pull requests to merge your branch with Develop.

Identify team members using @mentions if you need help.

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 WhatsApp 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 Review

Showing 2 changed files with 15 additions and 3 deletions.

5 GeoGatewayServer.js Show notes View

```
@@ -571,8 +571,9 @@ app.get('/uavsar_query/', function(req, res){
571 // has_wms query, this is the temporary solution, shall be removed later
572 app.get('/has_wms/', function(req, res) {
573 // var geoServerUrl = "http://gf8.ucs.indiana.edu:8080/geoserver/InSAR/wms?";
574 - var geoServerUrl = wmsUrl;

575 - if (req.server == 'coloring') {geoServerUrl = wmscolorUrl;}

576 var wmsParams = [
577     "version=1.1.1",
578     "request=DescribeLayer",
```

571 // has_wms query, this is the temporary solution, shall be removed later
572 app.get('/has_wms/', function(req, res) {
573 // var geoServerUrl = "http://gf8.ucs.indiana.edu:8080/geoserver/InSAR/wms?";
574 + var geoServerUrl;

575 + if (req.query.server == 'coloring') {geoServerUrl = wmscolorUrl;}
576 + else { geoServerUrl = wmsUrl;}
577 var wmsParams = [
578 "version=1.1.1",
579 "request=DescribeLayer",

Write Preview Styling with Markdown is supported

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Cancel Comment

13 html/js/tools.js View

```
@@ -936,8 +936,19 @@ function selectDataset(row, uid, dataname, heading, radardirection) {
936     if (typeof highresoverlay !== 'undefined') {
937         mapA.overlayMapTypes.setAt(0, null);
938     }
939 -
940 + //load color mapping one if checked
941 + if($('#color-mapping-checkbox').prop('checked')) {
942 +     var has_coloring;
943 +     has_coloring = checkwmslayer(uid,"coloring");
944 +     if (has_coloring) {
```

Code Release Process



Choose a Release Manager.

Discuss using Issues



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



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 branch with Develop

Using GitHub Wikis for Documentation



Good code documents itself, but...



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



Use these to describe your project.

Minimally, anything the instructors need to know to check your milestones.



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



Your grader will only look at the wiki.

Graders will not spend time trying to understand your setup.

Make GitHub Announcement (Optional)

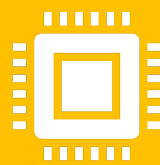


You can use GitHub to
announce a release
and make it citable

<https://guides.github.com/activities/citable-code/>



This gives you a Document Object
Identifier (DOI)



Useful for citing code: “This result
was produced by this specific version
of our code”

Some “Apache Way” Lessons

- Community over code.
- Discuss issues publically in an archived, citable manner.
- Assign yourself to issues.
 - Volunteer
- Cite the issue(s) associated with each commit.
- Review pull requests for code bases you can’t write to
 - Patches -> Apache
- Call votes on important decisions
 - Team policies with git branches, code review, issue organization, agile policies
 - Software releases
 - Granting write access to important branches.
- Make and announce your source code releases.
- And be prepared for what happens next
 - Documentation, build systems, bug handling, code licensing, code attributions, ...

