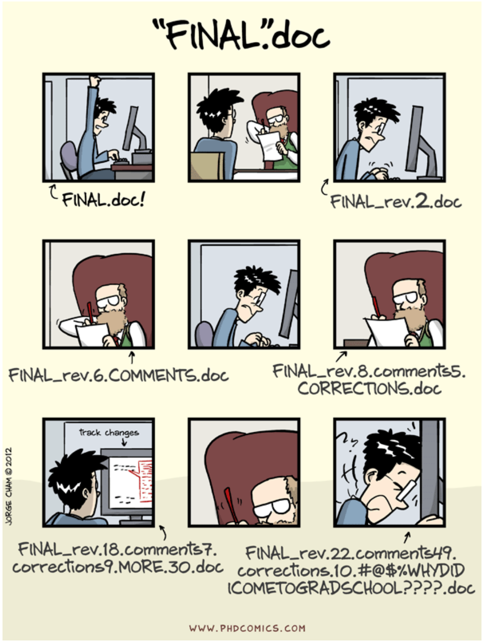
Software Version Control - GitHub

Agron5106 - Computational Skills for Biological Data Analysis

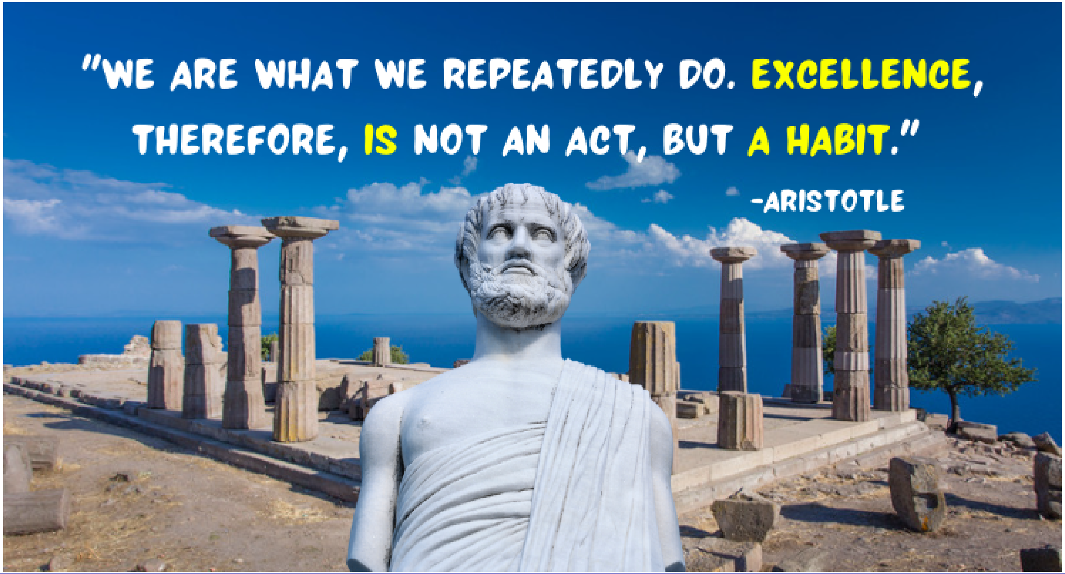
NTU

Department of Agronomy

Table of Contents


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Human vs Robot


"WE ARE WHAT WE REPEATEDLY DO. EXCELLENCE, THEREFORE, IS NOT AN ACT, BUT A HABIT."
-ARISTOTLE

Syntax

Commands

```
# In terminal/WSL/Bash/zsh

# This is the command you need to type (No $ sign).
$ cd ~
$ pwd
```

Outputs

```
# Expected output

# These are outputs you should see
/home/YOUR_USERNAME/
```

Table of Contents

Issues and Pull Requests

From Assignment 2

Please follow this interactive short course before you attempt this question.
https://github.com/skills/introduction-to-github
Alternatively, this is the document based tutorial:
https://github.com/education/github-starter-course
Demonstrate the following task in your repository
`agron5106-assignment2-YOUR-USERNAME`.

- Create an issue with meaningful title and message in your repository
- Create a new branch
- Make at least two commits on the new branch. Demonstrate what are you working on at this branch.
- Create a "Pull request" from the new branch to the `"main"` branch.
- Merge the Pull request into the `"main"` branch

Issues

Use GitHub Issues to track ideas, feedback, tasks, or bugs for work on GitHub.

- Integrated with GitHub
- Quickly create issues
- Track work
- Stay up to date
- Community management
- Efficient communication
- Comparing issues and discussions

Pull requests

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

Branch

Use a branch to isolate development work without affecting other branches in the repository. Each repository has one default branch `main`, and can have multiple other branches. You can merge a branch into another branch using a pull request.

Branch

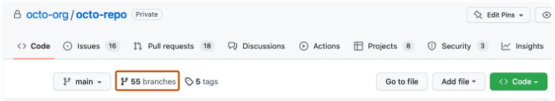
Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository.

Once you're satisfied with your work, you can open a pull request to merge the changes in the current branch (the head branch) into another branch (the base branch). For more information, see "About pull requests."

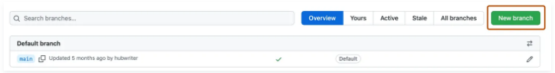
GitHub Doc - creating a branch

Creating a branch via the branches overview

1. On GitHub.com, navigate to the main page of the repository.
2. Above the list of files, click **Branches**.



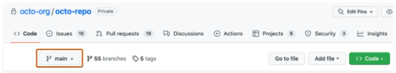
3. Click **New branch**.



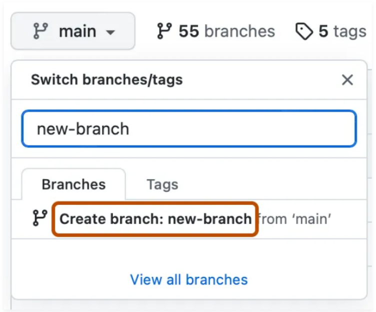
4. Under "Branch name", type a name for the branch.
5. Under "Branch source", choose a source for your branch.
   - If your repository is a fork, select the repository dropdown menu and click your fork or the upstream repository.
   - Select the branch dropdown menu and click a branch.
6. Click **Create branch**.

Creating a branch using the branch dropdown

1. On GitHub.com, navigate to the main page of the repository.
2. Select the branch selector dropdown menu.



3. Optionally, if you want to create the new branch from a branch other than the default branch of the repository, click another branch, then select the branch selector dropdown menu again.
4. In the "Find or create a branch..." text field, type a unique name for your new branch, then click **Create branch**.



Create new branch on GitHub

```
# Run `git clone` if needed.
# Current status, and branch
$ git status
$ git branch -a

# Check remote (GitHub) infromation
$ git remote update
$ git branch -a
```

```
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
  remotes/origin/fix_calculation
```

Create new branch on GitHub

```
# To switch to another branch
$ git checkout fix_calculation
$ git status
$ git branch -a
```

```
* fix_calculation
  main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
  remotes/origin/fix_calculation
```

Now, you are working on a different branch `fix_calculation` .

Create new branch on GitHub

```
# Get update version from GitHub
$ git pull

# Upload your update to GitHub
$ git push

# switch to another branch
# git checkout BRANCH_NAME
$ git checkout main
$ git branch -a
```

Create new branch locally

```
# Let's go back to the main branch first
$ git checkout main
$ git branch -a

# To create a new branch locally
# SYNTAX: git checkout -b NEW_BRANCH_NAME
$ git checkout -b feature_update_calculation
$ git branch -a
```

```
* feature_update_calculation
main
remotes/origin/HEAD -> origin/main
remotes/origin/main
```

Create new branch locally

```
# To upload this new local branch to GitHub
$ git push # THIS WILL NOT WORK!!
$ git branch -a
```

```
# You will NOT see remotes/origin/feature_update_...
* feature_update_calculation
main
remotes/origin/HEAD -> origin/main
remotes/origin/main
```

Why? We need to tell git where you are going to `git push` to. You can have multiple remote repositories (multiple GitHub accounts, multiple git remote servers).

Create new branch locally

```
# To upload this new local branch to GitHub
# SYNTAX: git push REMOTE_NAME LOCAL_BRANCH_NAME
$ git push origin feature_update_calculation
$ git branch -a
```

```
* feature_update_calculation
main
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/feature_update_calculation
```

What is `origin`?

git remote and origin

```
# Display the URL/address of your GitHub repository
$ git remote -v
```

```
origin  https://github.com/CompAgronUser/farm_project.git (fetch)
origin  https://github.com/CompAgronUser/farm_project.git (push)
```

`origin` is just a label/nickname of your current GitHub repository. By default, we call it `origin`. But it doesn't have to.

git log and git branch

```
# Show branches
$ git branch -a
$ git branch -avv



# Display log
$ git log --oneline

# Display both log and branches
$ git log --oneline --graph

# Summary stats
$ git log --oneline --graph --stat
```

Pull requests (again)

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

Table of Contents

Workflow

With Git, you can use a variety of branching strategies and workflows. Having a structured workflow for collaboration in complex projects is crucial for several reasons:

- Code organisation
- Version control
- Code quality
- Traceability and accountability
- Easier onboarding
- Time and resource management
- CI/CD: Continuous integration and Continuous deployment

Workflow - Notes

- Code organisation: Keep the codebase organised, prevent overlapping work, and ensure focused efforts towards a common goal.
- Version control: Allow simultaneous work on different features without conflicts, maintaining code stability.
- Code quality: A code review and approval process helps maintain high code quality and adherence to coding standards.
- Traceability and accountability: Enable tracking of changes and their authors, simplifying issue identification and responsibility assignment.
- Easier onboarding: Help new team members quickly grasp the development process, and start contributing effectively.
- Time and resource management: Enable better planning, resource allocation, and meeting deadlines, ensuring an efficient development process.
- CI/CD: Incorporate automated testing and deployment processes, streamlining the release cycle and delivering high-quality software consistently.

Common workflow

- Centralised workflow - Everything is on a single branch.
- GitHub flow - Lightweight workflow recommended by GitHub.
- Git flow
- GitLab flow
- Trunk-based Development

GitHub flow

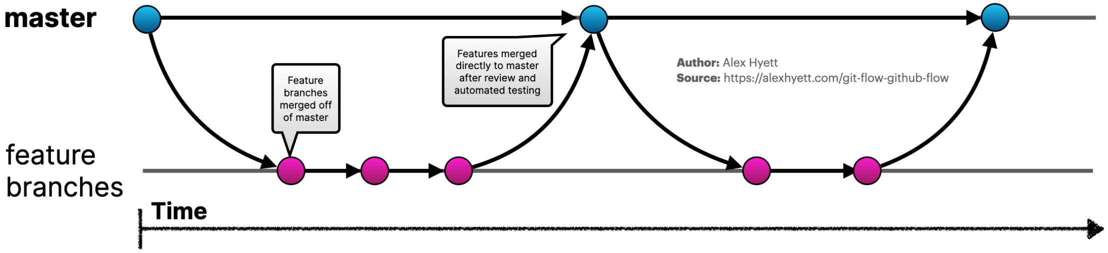
GitHub flow is a lightweight, branch-based workflow. The GitHub flow is useful for everyone, not just developers. For example, here at GitHub, we use GitHub flow for our site policy, documentation, and roadmap.

GitHub Doc - QuickStart - GitHub Flow

GitHub flow

- Create a branch - with a short, descriptive branch name
- Make changes
- Create a pull request
- Address review comments
- Merge your pull request
- Delete your branch (archive)

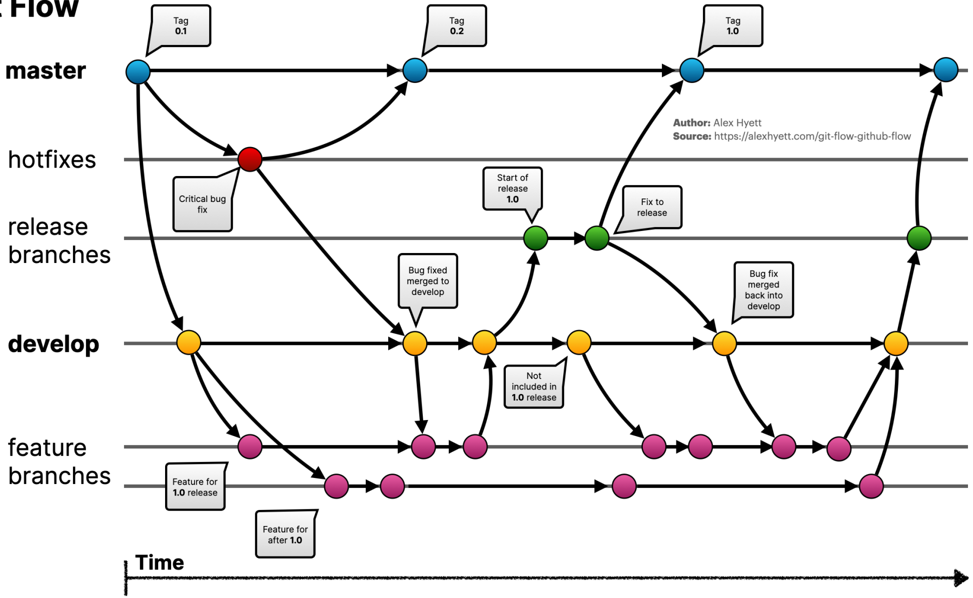
GitHub Flow


master
feature branches
Feature branches merged off of master
Features merged directly to master after review and automated testing
Author: Alex Hyett
Source: https://alexhyett.com/git-flow-github-flow
Time

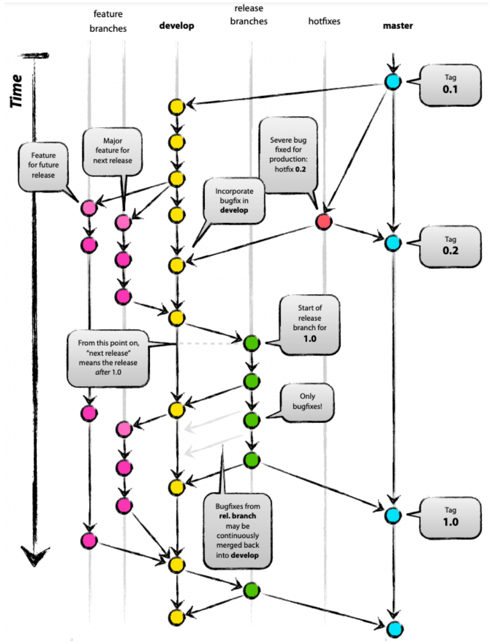
Git flow

Git flow was one of the first proposals to use Git branches, and it has received a lot of attention. It suggests a `main` branch and a separate `develop` branch, with supporting branches for `features`, `releases`, and `hotfixes`.

Git Flow


Tag 0.1
Tag 0.2
Tag 1.0
master
hotfixes
Critical bug fix
release branches
Start of release 1.0
Fix to release
Author: Alex Hyett
Source: https://alexhyett.com/git-flow-github-flow
Bug fixed merged to develop
Bug fix merged back into develop
develop
Not included in 1.0 release
feature branches
Feature for 1.0 release
Feature for after 1.0
Time

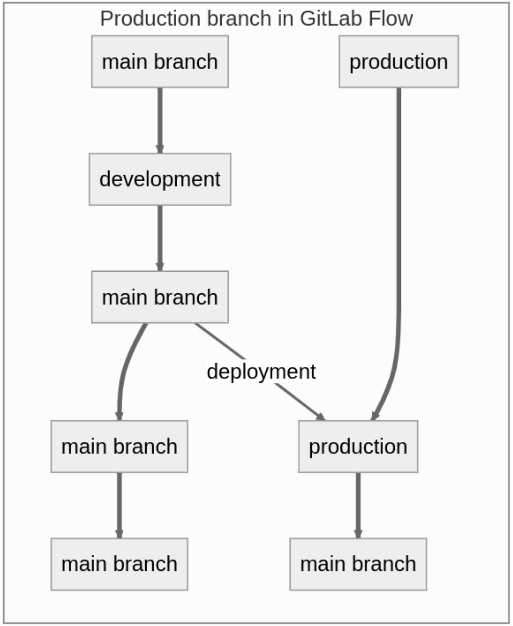
Original git flow


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Gitlab flow

GitHub flow assumes you can deploy to production every time you merge a feature branch. While this is possible in some cases, such as SaaS applications, there are some cases where this is not possible, such as:

- You don't control the timing of a release. For example, an iOS application that is released when it passes App Store validation.
- You have deployment windows - for example, workdays from 10 AM to 4 PM when the operations team is at full capacity - but you also merge code at other times.


Production branch in GitLab Flow
main branch
production
development
main branch
deployment
main branch
production
main branch
main branch

Trunk-based development

Trunk-based development is a version control management practice where developers merge small, frequent updates to a core “trunk” or main branch.

Trunk-based development

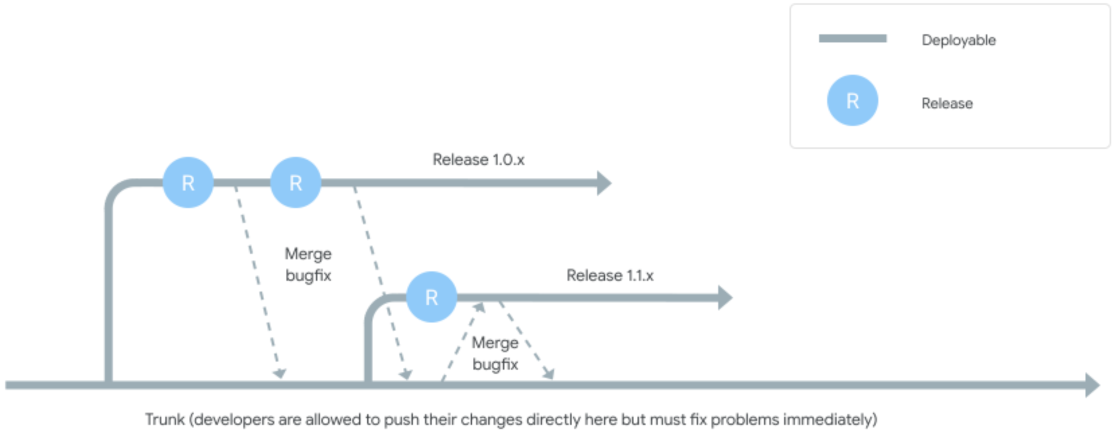

Deployable
R
Release
Release 1.0.x
R
R
Merge
bugfix
Release 1.1.x
R
Merge
bugfix
Trunk (developers are allowed to push their changes directly here but must fix problems immediately)

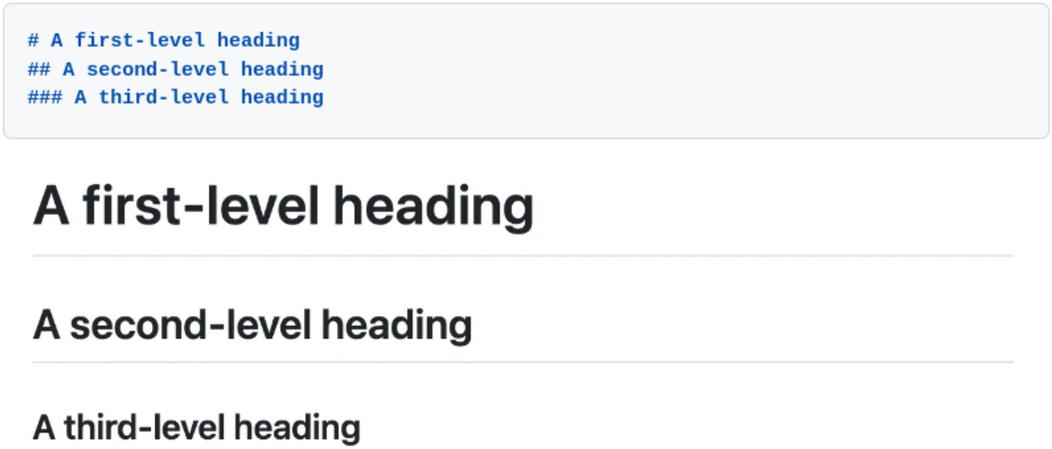
Table of Contents

Markdown

Markdown is an easy-to-read, easy-to-write language for formatting plain text.

- Basic writing and formatting syntax
- Communicate using Markdown
- Not limited to GitHub.
- Not limited to plain text.

Markdown - Heading



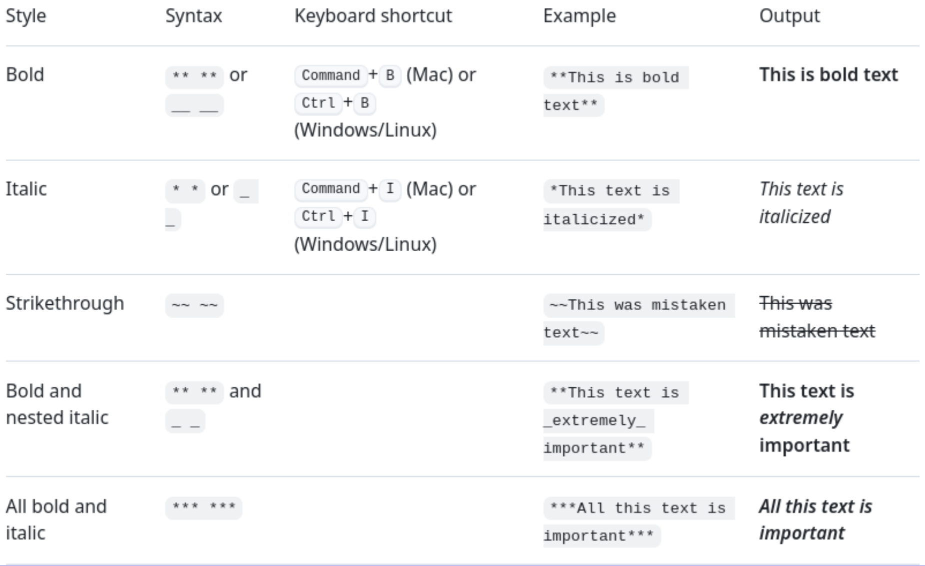
```
# A first-level heading
## A second-level heading
### A third-level heading
```

A first-level heading

A second-level heading

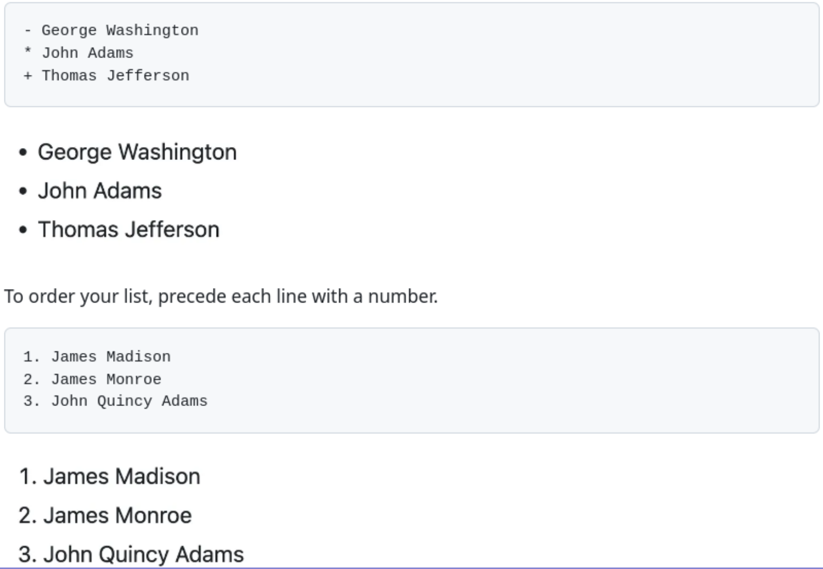
A third-level heading

Markdown - Style



Style	Syntax	Keyboard shortcut	Example	Output
Bold	`** **` or `__ __`	`Command` + `B` (Mac) or `Ctrl` + `B` (Windows/Linux)	`**This is bold text**`	**This is bold text**
Italic	`* *` or `_ _`	`Command` + `I` (Mac) or `Ctrl` + `I` (Windows/Linux)	`*This text is italicized*`	*This text is italicized*
Strikethrough	`~~ ~~`		`~~This was mistaken text~~`	~~This was mistaken text~~
Bold and nested italic	`** **` and `_ _`		`**This text is _extremely_ important**`	**This text is *extremely* important**
All bold and italic	`*** ***`		`***All this text is important***`	***All this text is important***

Markdown - List



```
- George Washington
* John Adams
+ Thomas Jefferson
```

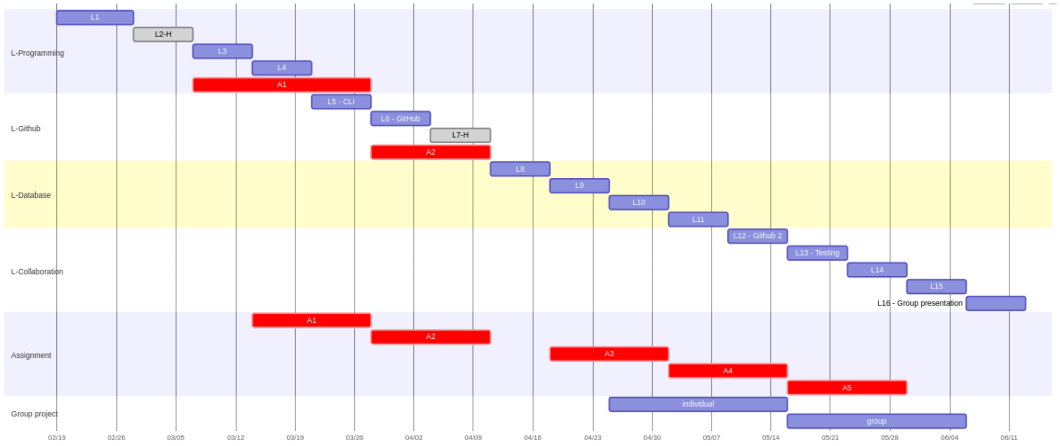
- George Washington
- John Adams
- Thomas Jefferson

To order your list, precede each line with a number.

```
1. James Madison
2. James Monroe
3. John Quincy Adams
```

1. James Madison
2. James Monroe
3. John Quincy Adams

Markdown with Mermaid



~~~
```mermaid
gantt
    dateFormat MM-DD
    axisFormat %m/%d
    TodayMarker off
    tickInterval 1week

section L-Programming
    L1  : L1, 02-19, 02-28
    L2-H  : done, L2, after L1, 7d
    L3  :L3, after L2, 7d
    L4  :L4, after L3, 7d
    A1  : crit, 03-07, 03-28
section L-Github
    L5 - CLI        : after L4, 7d
    L6 - GitHub     : 7d
    L7-H            : done, 7d
    A2              : crit, 03-28, 04-11
~~~

Markdown with Mermaid

Mermaid

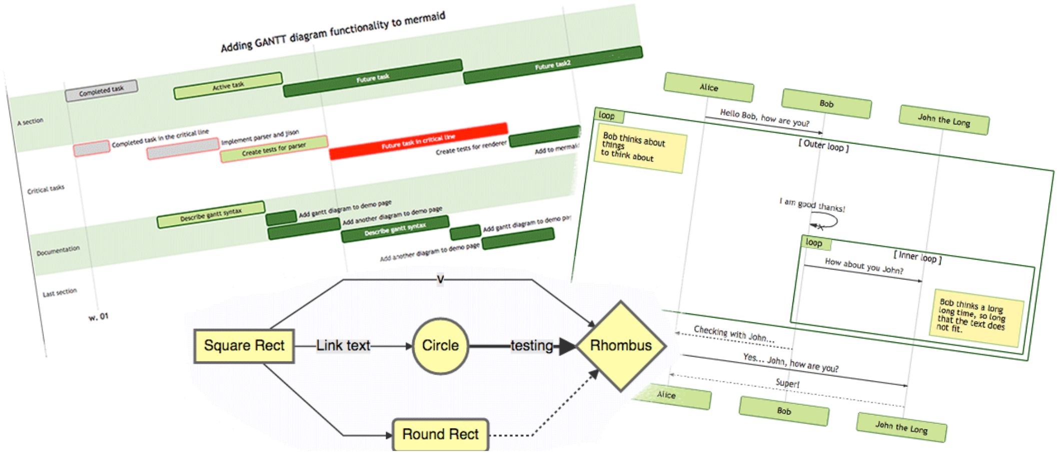
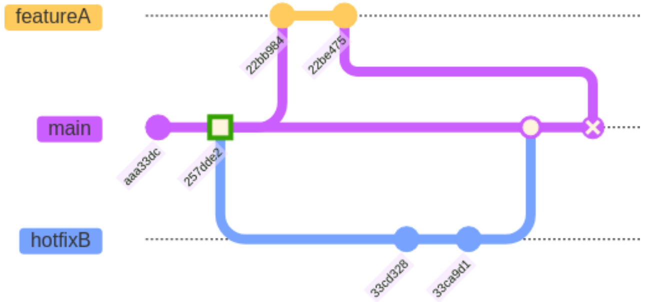


Table of Contents

Merge conflict

If there are multiple and different changes at the same line of code, sometimes git cannot figure out which is the one you want.



featureA

```
ans <- t * 5 + 32
```

main

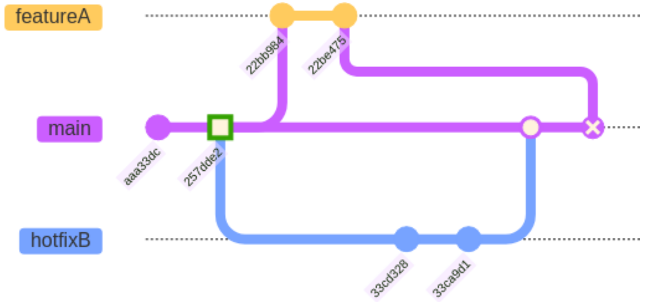
```
ans <- t * 5
```

hotfixB

```
ans <- t * 5 - 64
```

Merge conflict

If there are multiple and different changes at the same line of code, sometimes git cannot figure out which is the one you want.



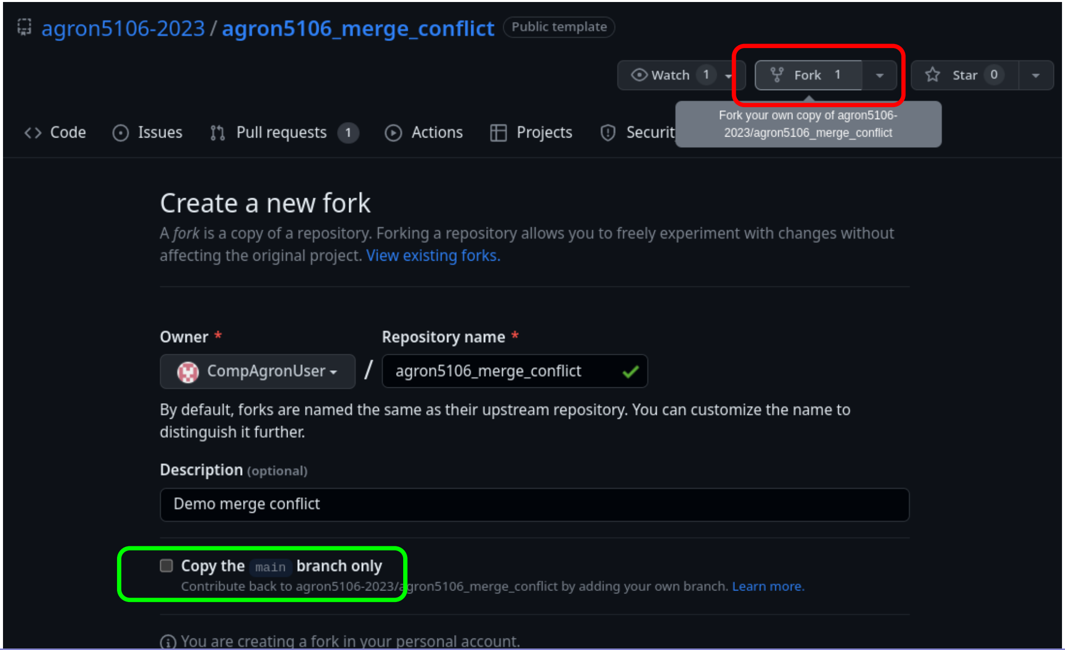
`featureA`
ans <- t * 5 + 32

`main`
ans <- t * 5

`hotfixB`
ans <- 32 + t * 5

Merge conflict workshop

- Start with a demo GitHub repository. https://github.com/agron5106-2023/agron5106_merge_conflict
- `Fork` it - make a personal copy - with all branches. Unfortunately, GitHub classroom doesn't show any history, so we can't force a merge conflict to happen.
- Create a Pull request, make sure it's within the same repository, this should result a conflict.
- Resolve the conflict on GitHub.



agron5106-2023 / agron5106_merge_conflict Public template

Watch 1 Fork 1 Star 0

Fork your own copy of agron5106-2023/agron5106_merge_conflict

Code Issues Pull requests 1 Actions Projects Securit

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.

Owner * CompAgronUser / Repository name * agron5106_merge_conflict

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

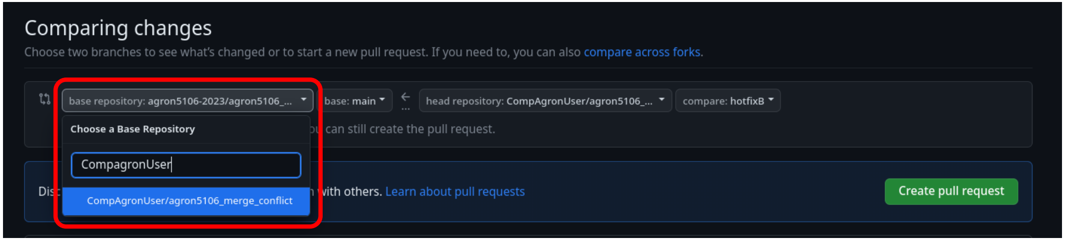
Demo merge conflict

☐ **Copy the `main` branch only**

Contribute back to agron5106-2023/agron5106_merge_conflict by adding your own branch. Learn more.

You are creating a fork in your personal account.

Create a Pull request.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: agron5106-2023/agron5106_... base: main ← head repository: CompAgronUser/agron5106_... compare: hotfixB

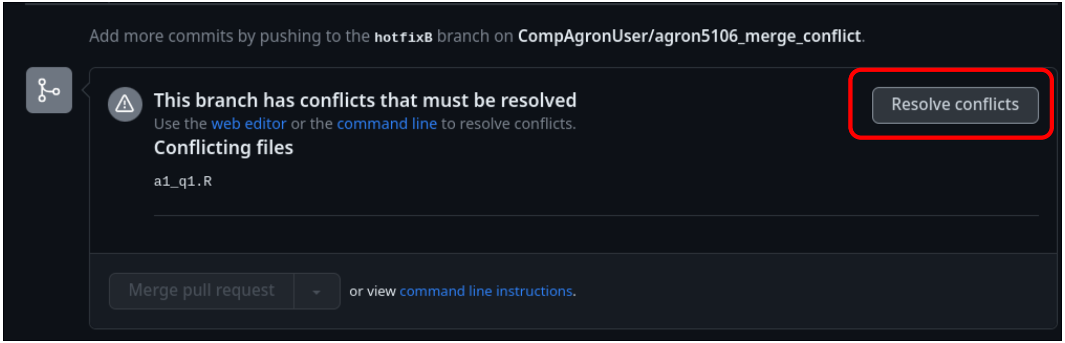
Choose a Base Repository

CompagronUser

CompAgronUser/agron5106_merge_conflict

…u can still create the pull request.

Dis… …n with others. Learn about pull requests



Add more commits by pushing to the `hotfixB` branch on **CompAgronUser/agron5106_merge_conflict**.

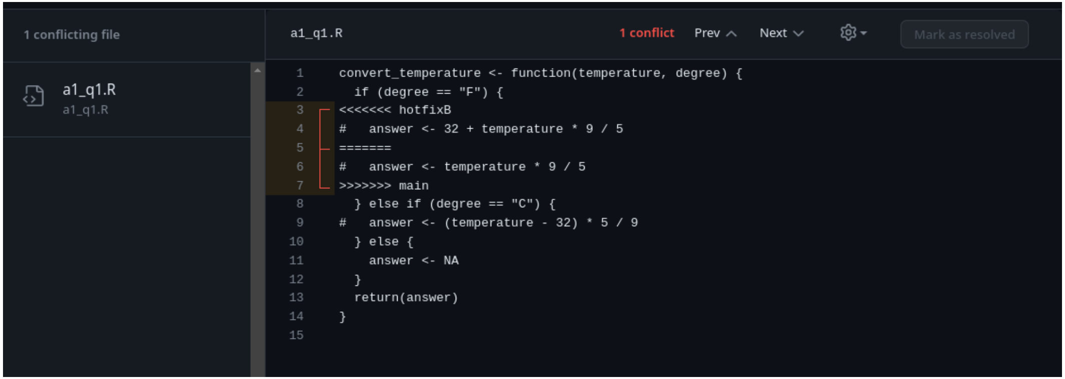
This branch has conflicts that must be resolved

Use the web editor or the command line to resolve conflicts.

Conflicting files

`a1_q1.R`

or view command line instructions.



```
convert_temperature <- function(temperature, degree) {
  if (degree == "F") {
<<<<<<< hotfixB
#   answer <- 32 + temperature * 9 / 5
=======
#   answer <- temperature * 9 / 5
>>>>>>> main
  } else if (degree == "C") {
#   answer <- (temperature - 32) * 5 / 9
  } else {
    answer <- NA
  }
  return(answer)
}
```

Some tips to reduce merge conflict

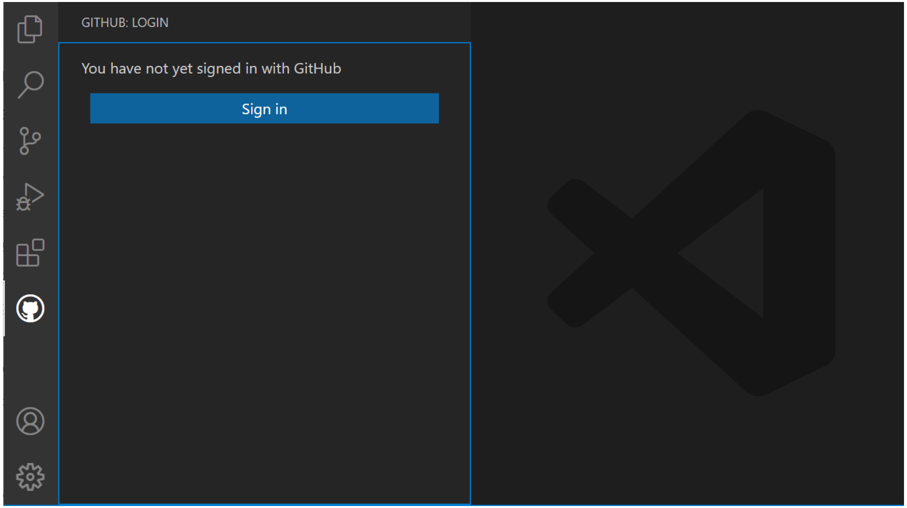
Merge conflict can be extremely difficult to deal with, especially when there are a lot of conflicts.
Hard to avoid them completely, but small conflicts with a few lines are relatively easier to address.

- Have a project plan.
- Commit frequently: A lot of small commits.
- Create pull request frequently: Short-lived branches.
- Sync frequently: `git pull`, `git push`.

Table of Contents

VS Code

Working with GitHub in VS Code

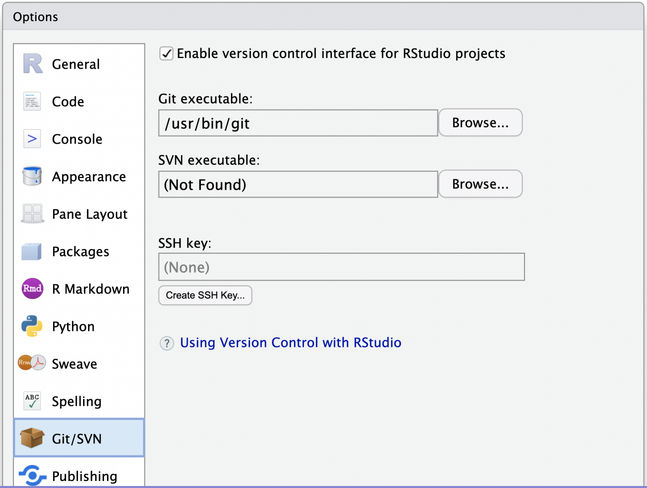


You have not yet signed in with GitHub

RStudio/Posit

RStudio - Version Control

Managing – Part 2 (Github and RStudio)


Options

[✓] Enable version control interface for RStudio projects

Git executable: /usr/bin/git

SVN executable: (Not Found)

SSH key: (None)

Using Version Control with RStudio