LEARN THE BASICS OF GIT AND GITHUB

6-9 Mar 2025, SCaLE 22x

PERRY RIVERA

Customer Success Architect @ Red Hat

lajuggler@fedoraproject.org

@juggler:matrix.org

riveraconsulting

Today We'll Cover...

- Who is Perry?
- A brief history of version control, Git and GitHub.
- What are they? Why are they used?
- How to set up Git?
- What is a commit? How do I stage code?
- What is a tag?
- What is a branch?
- How do I setup a new Github account?

Today We'll Cover...

- How do I pull/update code?
- What is the .git folder?
- Why cloning is important
- How to clone data down
- What is cherry picking?

Today We'll Cover...

- Feel free to use a laptop if you'd like to follow along. Familiarity with command-line is helpful but not required.
- Walkthrough: https://github.com/perryrivera/learn_the_basics_of_git_and_github/blob/main/git_basics_walkthrough.md
-

Disclaimer

Every effort has been made to offer current and accurate information to our users, but errors can occur. We assume no liability or responsibility for any errors or omissions in the content contained in this presentation. This information is provided “as is,” with no guarantees of completeness, accuracy or timeliness, and without warranties of any kind, express or implied. We do not warrant that this presentation, various services provided through this presentation, and any information, software, or other material downloaded from this presentation, will be uninterrupted, error-free, omission-free, or free of viruses or other harmful components. To the fullest extent permissible pursuant to applicable law, we disclaim all warranties, express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose.

About Perry

- Perry Rivera
- Fedora Ambassador since 2015
- Fun fact: My first Fedora version is Fedora Core 4
- Red Hat Customer Success Architect
  - DevOps team for demo.redhat.com (customers) partner.demo.redhat.com (partners)
  - We enable customers through our cutting-edge technologies like OpenShift virtualization and OpenShift AI
  - Our teams use Fedora as our daily driver

A Brief History of Version Control

- A long time ago...
- Software engineers would keep many versions/files around
  - coolprog1.c
  - coolprog2c
  - And so on..
- These poor souls, If they remembered, backed up these files, duplicating similar data

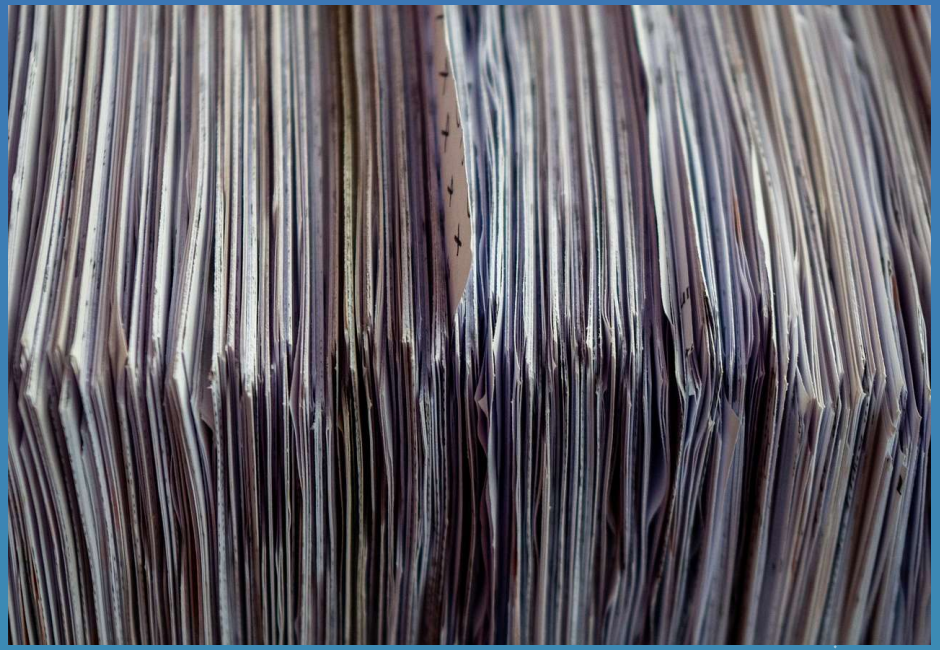
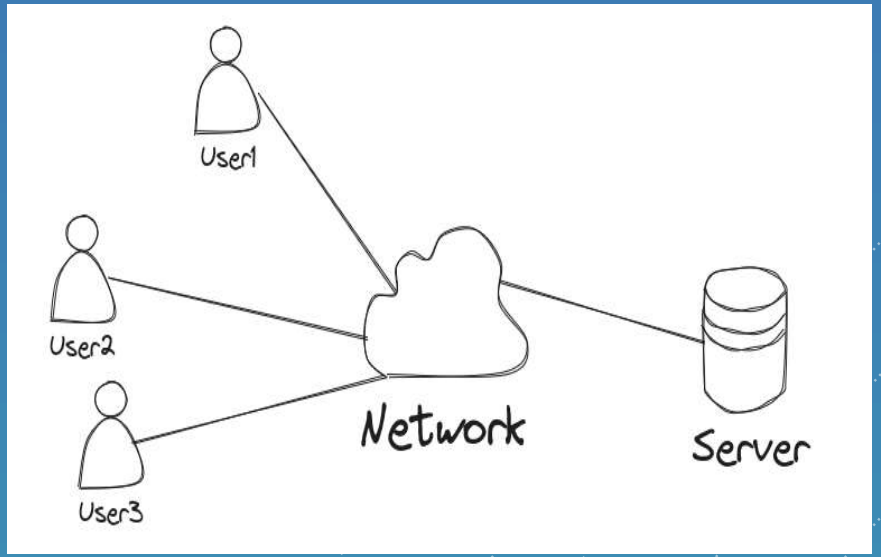


Image by wal_172619 from Pixabay

Maybe Not So

A Brief History of Version Control

- Eventually, tools like rcs (Revision Control System), cvs (Concurrent Versions System), and later, svn (Subversion), were developed
- Software engineers would write code locally on their client systems and push their changes up to a server
- A server admin would have to back up user data *shudder*



Maybe Not So

A Brief History of Version Control

- Linus Torvalds observed limitations in cvs/svn
  - Client/server architecture was a central and potential single point of failure without backups
  - CVS tracked updates file-by-file, while SVN tracks an entire commit as a new revision
  - CVS had some issues with same-time commits
  - CVS didn't support binary files well (full copies of binaries versus binary-differenced commits)
  - Server did the heavy lifting for the code base

Maybe Not So

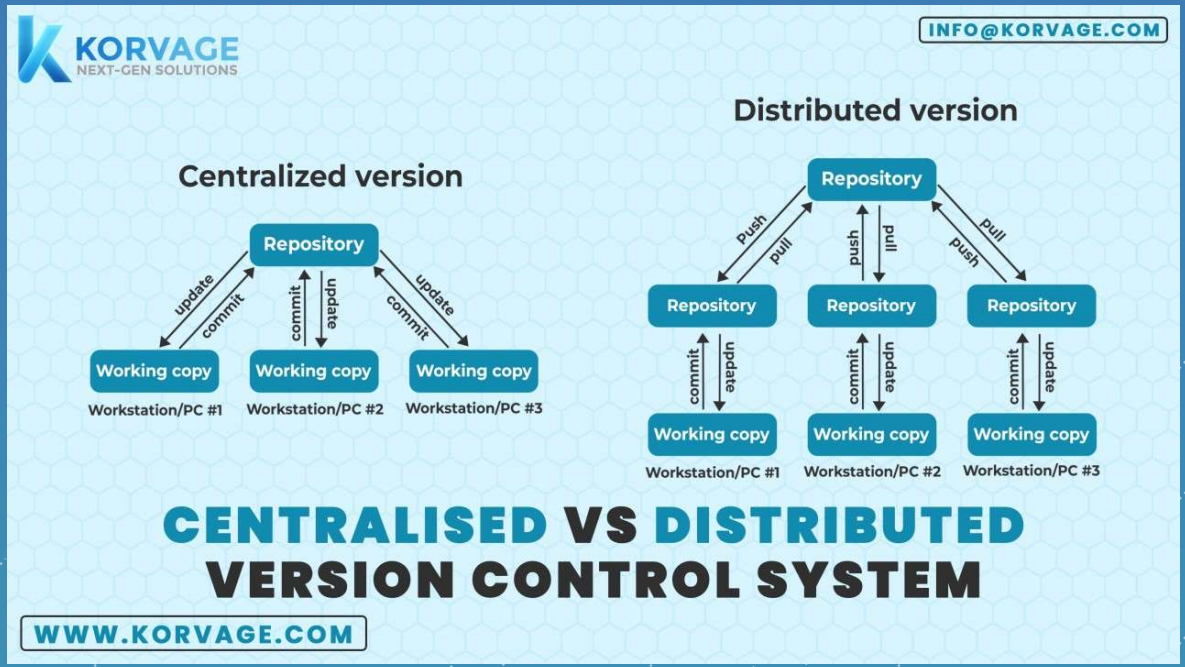
A Brief History of Version Control

- Linus Torvalds developed git as a result
  - Distributed architecture means everyone has a local copy of the entire repository, especially the history
  - Collaboration
  - Fast and asynchronous
  - Each directory has a complete history
  - This is a fast, efficient directory content manager.

Maybe Not So

A Brief History of Version Control

- Linus Torvalds developed git as a result
  - Distributed



Maybe Not So

A Brief History of Version Control

- How did it get its name?
  - git's historical commit message: "Initial revision of "git", the information manager from he**"
  - git - the stupid content tracker

https://funsizedatabytes.substack.com/p/git-on-with-it?utm_campaign=post&utm_medium=web

Maybe Not So

A Brief History of Version Control

- How did it get its name?
  - Per Wikipedia, "git" can mean anything, depending on your mood:
    - random three-letter pronounceable word not used by any common UNIX command.
    - stupid. contemptible and despicable. simple. Take your pick..
    - "global information tracker": you're in a good mood, and it actually works for you.

Installing git

- It comes pre-installed on some systems
- So, take it for a spin:
  - git version
  - If you get a version number as a response, you're all set
  - If not, refer to https://github.com/git-guides/install-git

How to set up Github Account

- https://github.com/
- Upper right-hand corner: Sign up
- Supply an e-mail, username, and password
- Other alternatives (not covered here):
  - Gitlab
  - Gitea
  - Bitlocker
  - And more...

Should I choose Free or Paid GitHub?

- There's an array of choices...it depends on your needs...
- Check out:
  - https://github.com/pricing
  - Take a look at the add-ons too and Compare Features section.
- If starting out, maybe start with the Free and go from there...

Generating a New Token

- Generate a token
  - https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-fine-grained-personal-access-token
    - Click your picture
    - Click Settings. On the left hand menu, scroll down and click Developer settings
    - Click Personal Access Tokens
    - Click Tokens (classic)
    - Click Generate New Token
    - Click Generate New Token (classic)

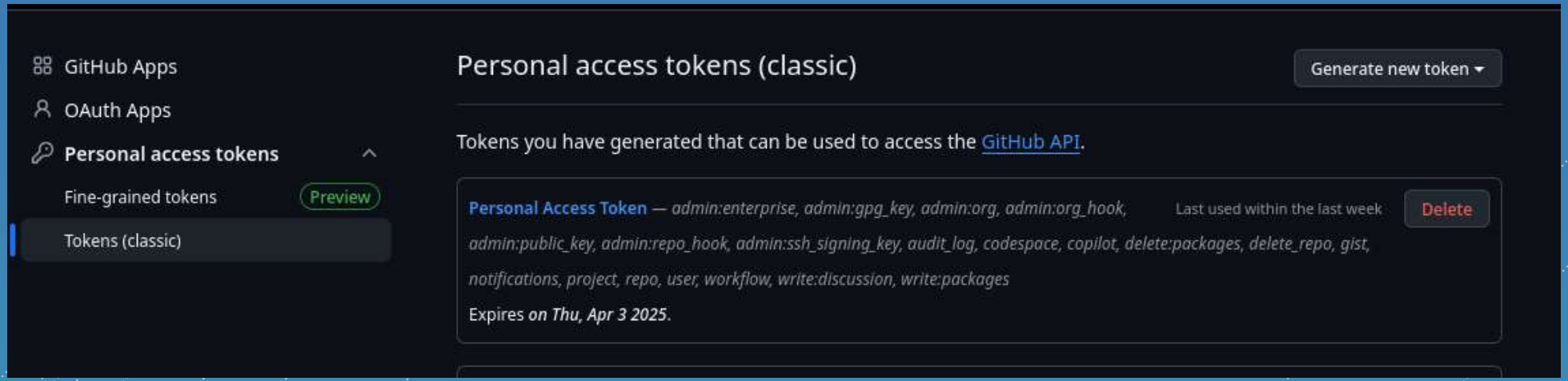
Generating a New Token

- Generate a token
  - (cont'd)
    - Note: Personal Access Token DD/MM/YYYY
    - Expiration: (leave at 30 days)
    - Access: If it's just you, choose toplevel checkboxes
    - Click Generate token
  - IMPORTANT: Make sure to copy and secure your token in a vault, such as KeePassXC

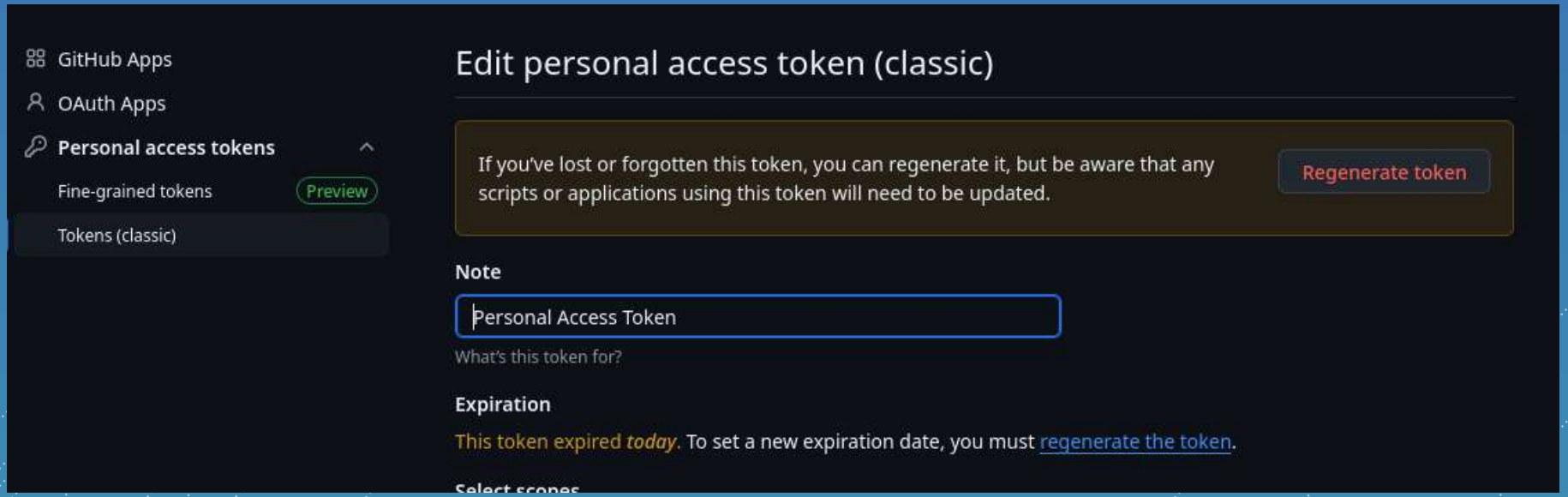
Re-generating a Token

- Re-generate a token
  - Click your picture
  - Click Settings. On the left hand menu, scroll down and click Developer settings
  - Click Personal Access Tokens
  - Click Tokens (classic)
  - Click on the token you created about a month ago...
  - Click Re-generate Token.

Re-generating a Token



Re-generating a New Token



Set the Stage...

- In your home directory, define a spot to set up repositories, e.g. repo
- mkdir repo && cd $_ [be sure to use double ampersand!]

Image by faraindahouse from Pixabay

How to Create a Repository

- In a browser, again using your username: https://github.com/perryrivera?tab=repositories
- In the upper-right hand side, click the New button
  - Repository name: my-first-repo
  - Description: my-first-repo
  - Select Private
  - Click Create repository

How to add a remote repository to your local repository

- In a terminal:
  - cd ~/repo
  - Create project directories under this, e.g. gitstudy, coolproject, etc.
  - cd into each directory

How to add a remote repository to your local repository

- In a terminal:
  - git init
    - Creates hidden .git metadata that the git command uses
  - Add a remote repository to your local repository. Use your git username
    - git remote add origin https://github.com/perryrivera/my-first-repo.git
    - NOTE: If you get error fatal: “not a git repository”, did you do a git init yet???

How to add a remote repository to your local repository (Example)

- mkdir my-first-repo
- cd my-first-repo
- git remote add origin <url>
- git clone <url>
- git status
- git push

How to Set Up git

- git clone <url>
  - Use clone to get a copy of a remote repository
  - How do I figure out <url>
    - Go to the place where you'd like to clone:
      - e.g. https://github.com/perryrivera/my-test-repo
      - Click the green Code button:

      - Click the copy button
        to the right of the https url:

How to Set Up git

- git clone <url> [continued]
  - How do I figure out <url>
    - Paste this into your command, e.g.:
      - git remote add origin https://github.com/janeuser/my-test-repo.git
      - git clone https://github.com/janeuser/my-test-repo.git
  - https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository

How to Set Up Global Config

- While still in ~/repo/my-first-repo
  - git config --global user.email "janeuser@gmail.com"
  - git config --global user.name "Jane User"
  - git config --list

How to Stage and Add Files

- echo blah > this_is_my_first_textfile
- ls
  - this_is_my_first_textfile
- git add <file>
  - Or, for multiple untracked files in current and subdirectories, use git add .
- git commit <file>
  - Enter commit message, e.g. Initial Commit

How to Tag A File

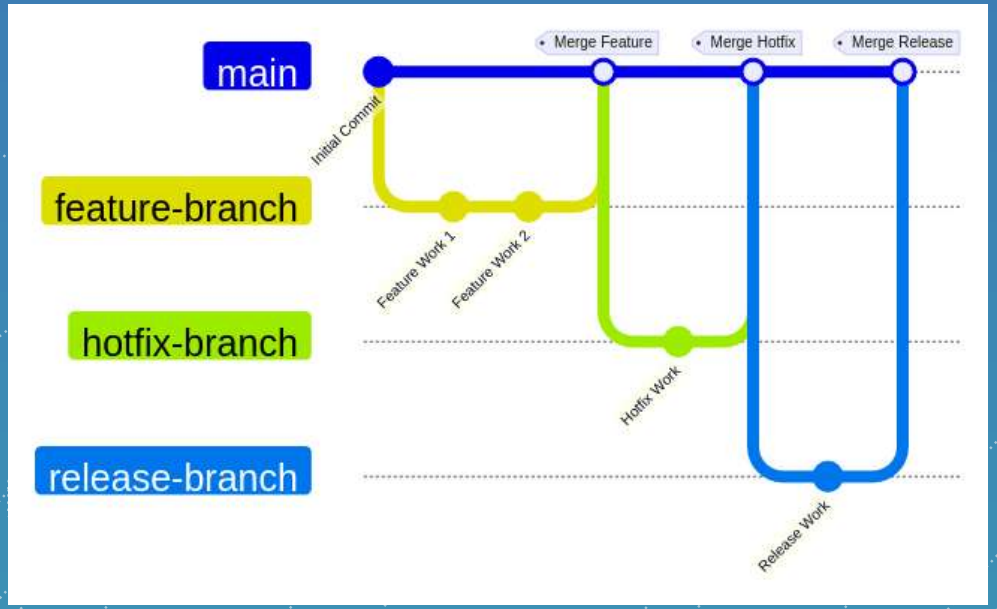
- Use it to tag the last commit
- Simple Tag: git tag
  - git tag release4 [annotation is last commit message]
- Annotative Tag
  - git -a tag v5.4.2 “Silver Edition Release v5.4.2”
- git log --one-line
- git tag -n

Pushing the file to origin’s repo

- git push --set-upstream origin master
  - Enter username
  - Enter password, which is a copy of your Personal Access Token. This is good for 30days (or whatever duration you set previously..).

Why do we want to branch? How to create a branch?

- Use it to branch off the default/main branch for new development / feature
-



Why do we want to branch? How to create a branch?

- Use it to branch off the default/main branch for new development /feature
- git branch firstBranch
- git checkout firstBranch
  - touch bugfix1
  - touch bugfix2
  - git add bugfix1
  - git add bugfix2

Why do we want to branch? How to create a branch?

- git checkout firstBranch (continued)
  - git stash
  - ls
  - git stash apply
  - ls
  - git commit
  - git push --set-upstream origin firstBranch
- git checkout main
- ls

Recap

We covered:

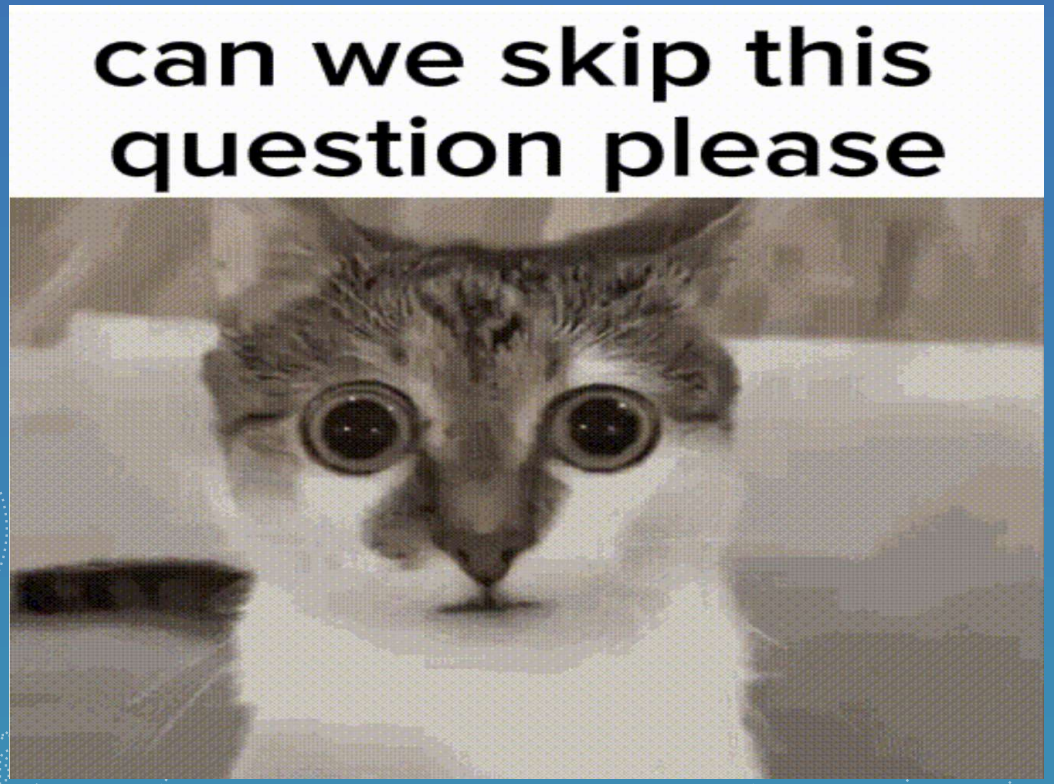
- A whirlwind view of version control, Git and GitHub and why they're useful
- The importance of the .git folder
- GitHub setup
- Pulling and updating code

Recap

We also covered:

- Staging and committing code
- Cloning code
- Branching basics

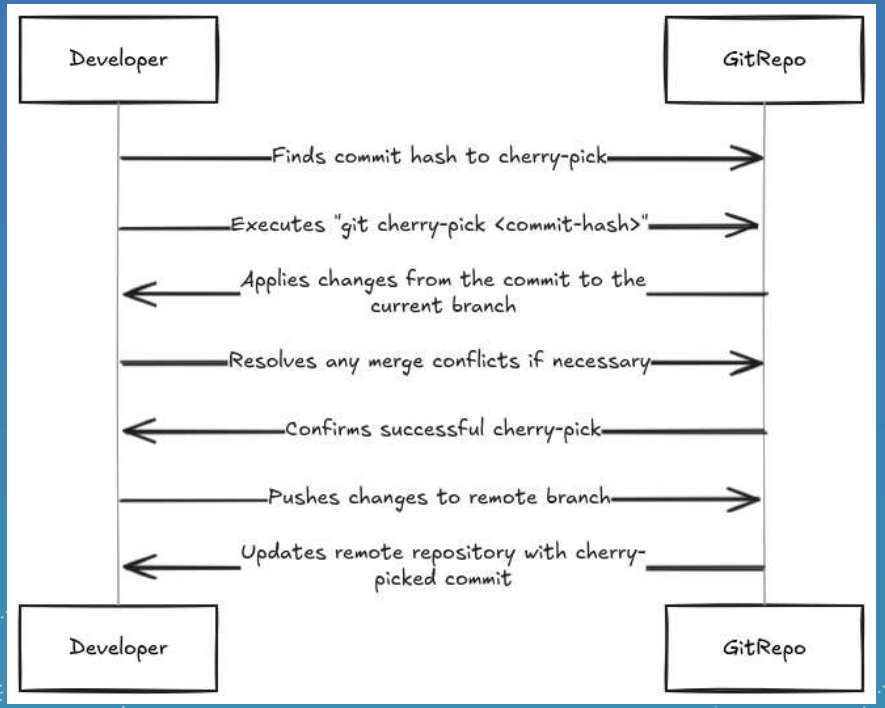
ANY QUESTIONS?



THANK YOU

APPENDIX

How to Cherry-pick Files



How to Cherry-pick Files

- Find the commit you want
- git log <name of committed file>
  - Copy the commit id
- git checkout featureBranch
- git pull

How to Cherry-pick Files

- git checkout main
- git pull
- git cherry-pick <commitID>
- git push origin/featureBranch

Resources

- Step by Step Cherry-Pick
  - https://www.youtube.com/watch?v=HQU8Y38KgdA
  - https://stackoverflow.com/questions/62788703/git-how-to-cherrypick-commit-from-one-branch-and-create-pull-request-for-anothe
  - https://www.atlassian.com/git/tutorials/cherry-pick
- Git for Ages 4 and Up: https://youtu.be/1ffBJ4sVUb4?si=veY4Om7NKlBSp1vt

Resources

- Atlassian's Git Cheatsheet
  https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- Using GitHub with git / GitHub
  https://code.visualstudio.com/docs/sourcecontrol/overview

  Be sure to check out the Next Steps links at the bottom, too

LEARN THE BASICS OF GIT AND GITHUB

6-9 Mar 2025, SCaLE 22x

PERRY RIVERA

Customer Success Architect @ Red Hat

lajuggler@fedoraproject.org

@juggler:matrix.org

riveraconsulting