Version Control with Git

- Before we start
  - Sign up at github.com

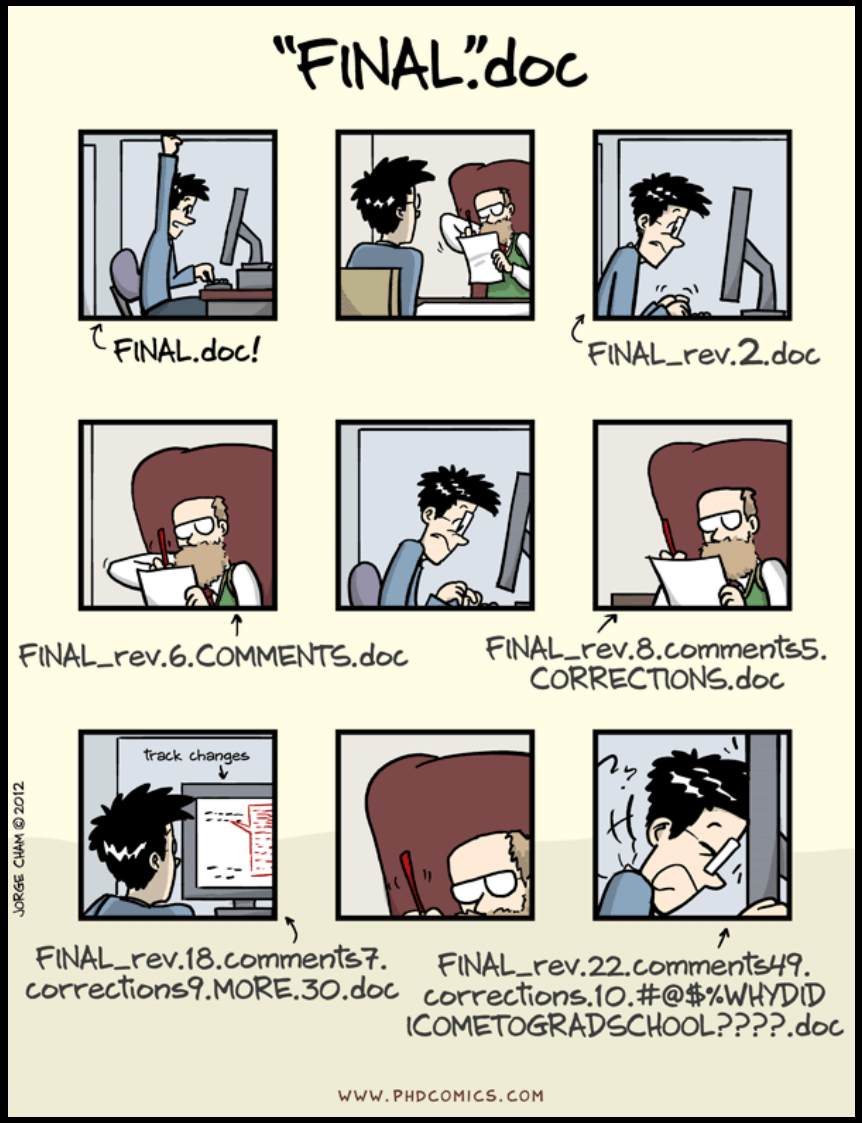
What is Version Control?

(AKA revision control, source control)

- Tracks changes to files
- Any file can be tracked
- Text (.txt, .csv, .py, .c, .r etc.) works best
    - These allow smart *diff* / *merge* etc.

Why Use Version Control? #1

- A more efficient backup
- Reproducibility



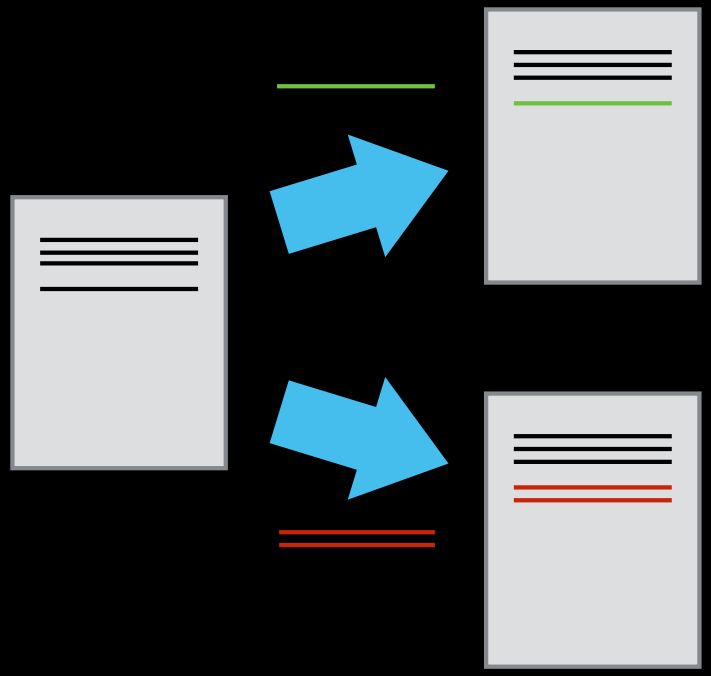
Why Use Version Control? #2

- Teamwork

Version Control Tracks Changes

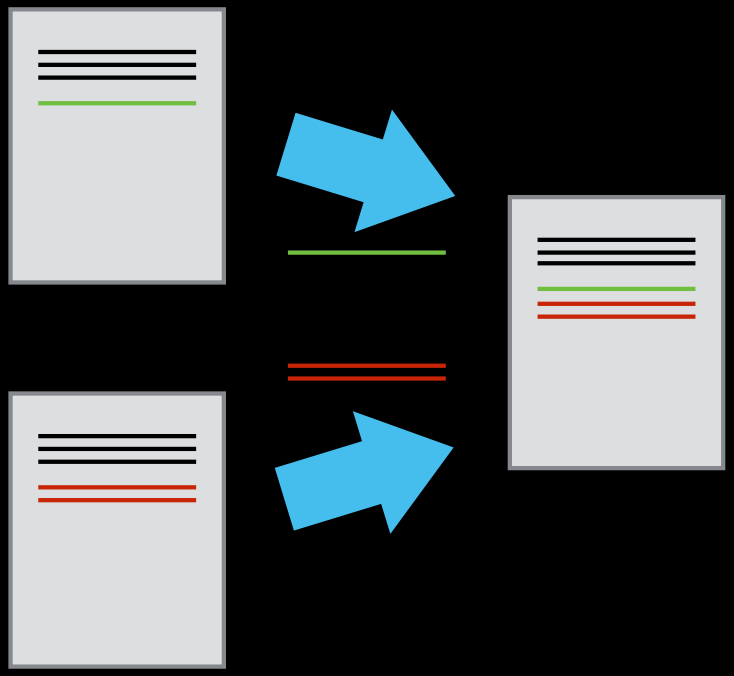
- Changes are tracked sequentially

Version Control Tracks Changes



- Different versions can be saved

Version Control Tracks Changes



- Multiple versions can be merged

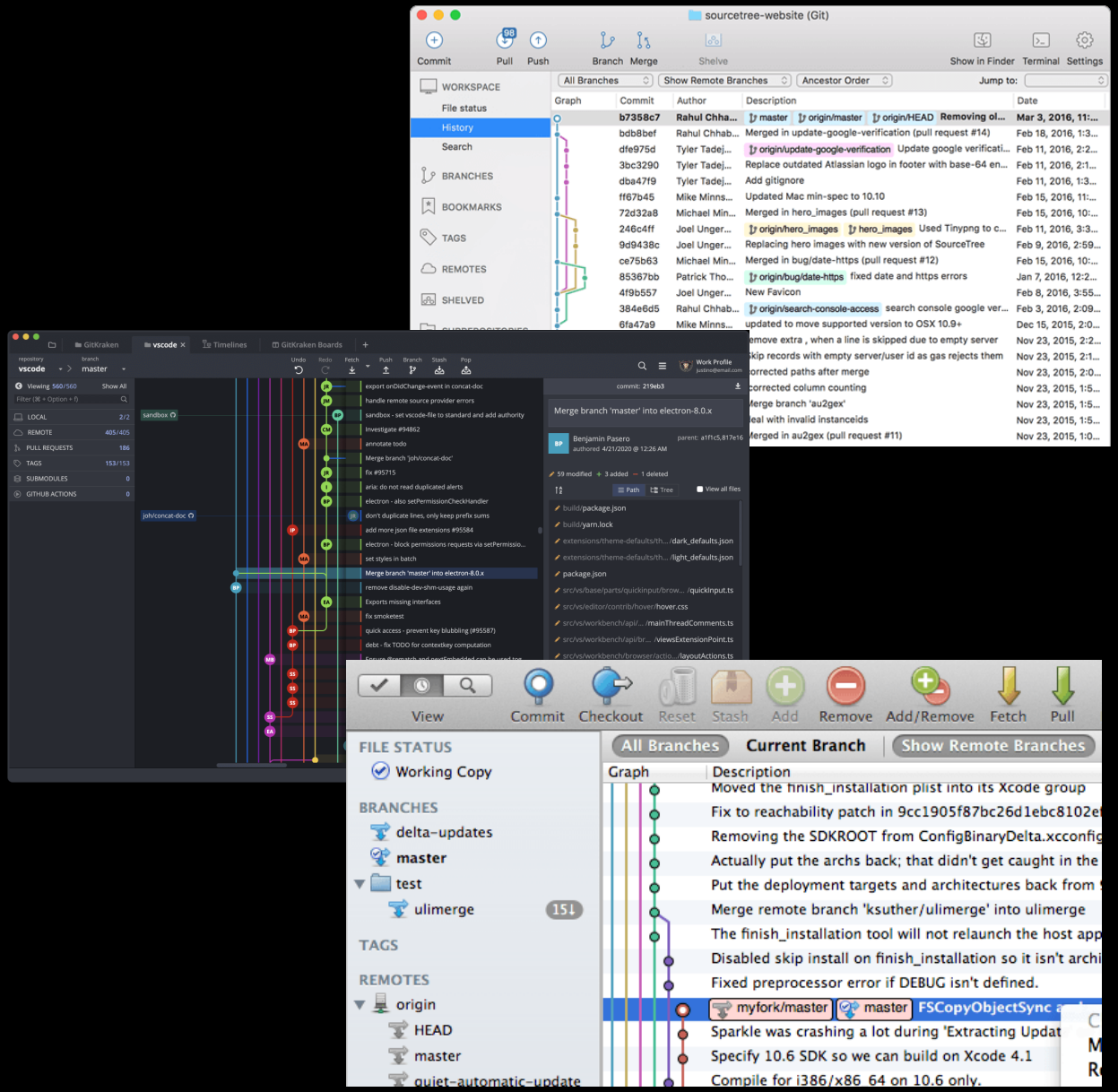
Version Control Alternatives

- Subversion (svn) - Centralised
- Mercurial (hg) - Distributed
- Git (git) – Distributed
  - Most widely used in academia!

- N.B. GitHub != git
  - Alternatives like GitLab exist

Graphical Version Control

- SourceTree
- Git Kraken
- Git GUI



Local Configuration

- git config

Getting Demo Files

- git clone https://github.com/Southampton-RSG/swc-ramp-git

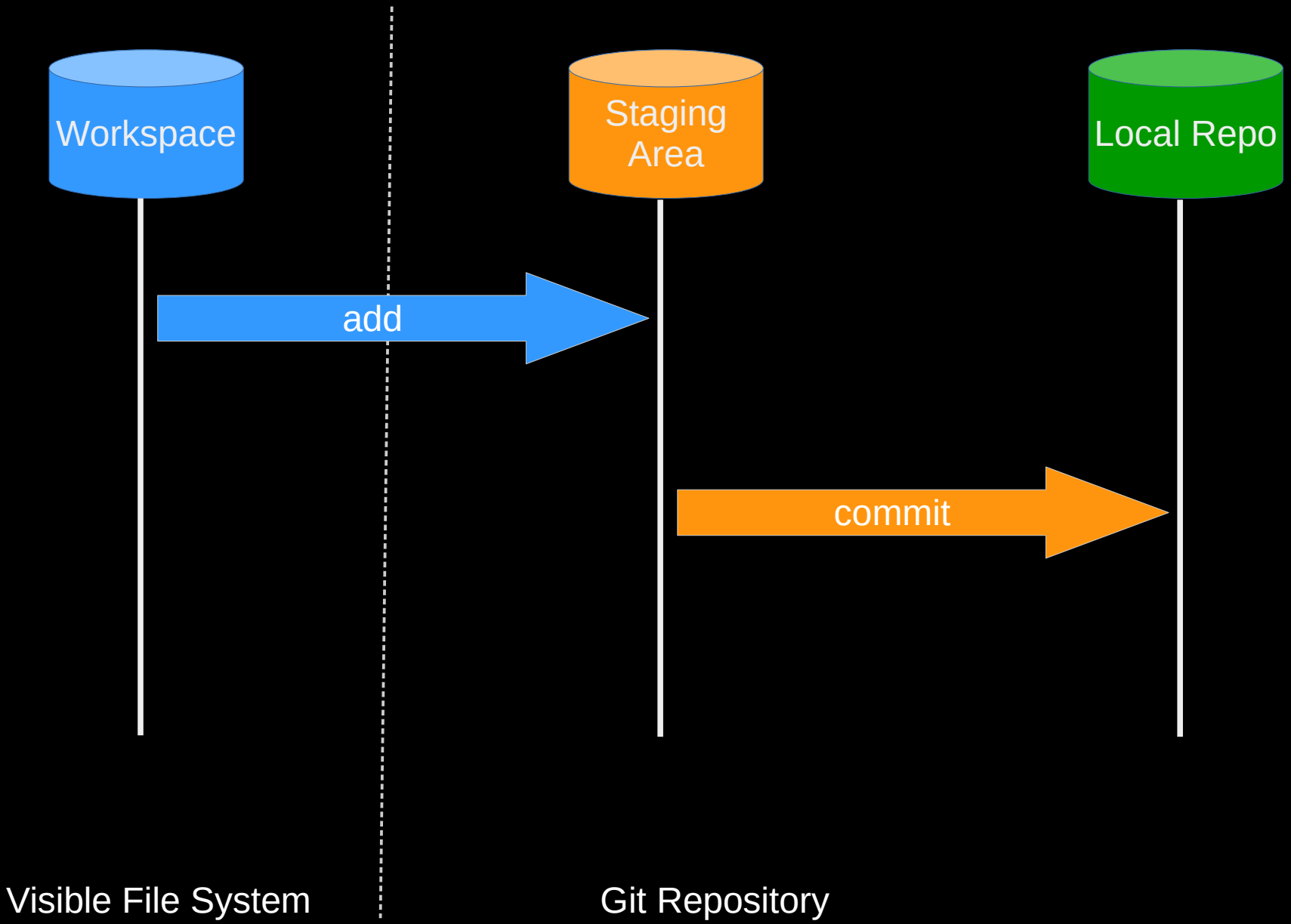
Creating a Repository

- git init
- git status

Tracking Changes to Files

- git add
- git commit

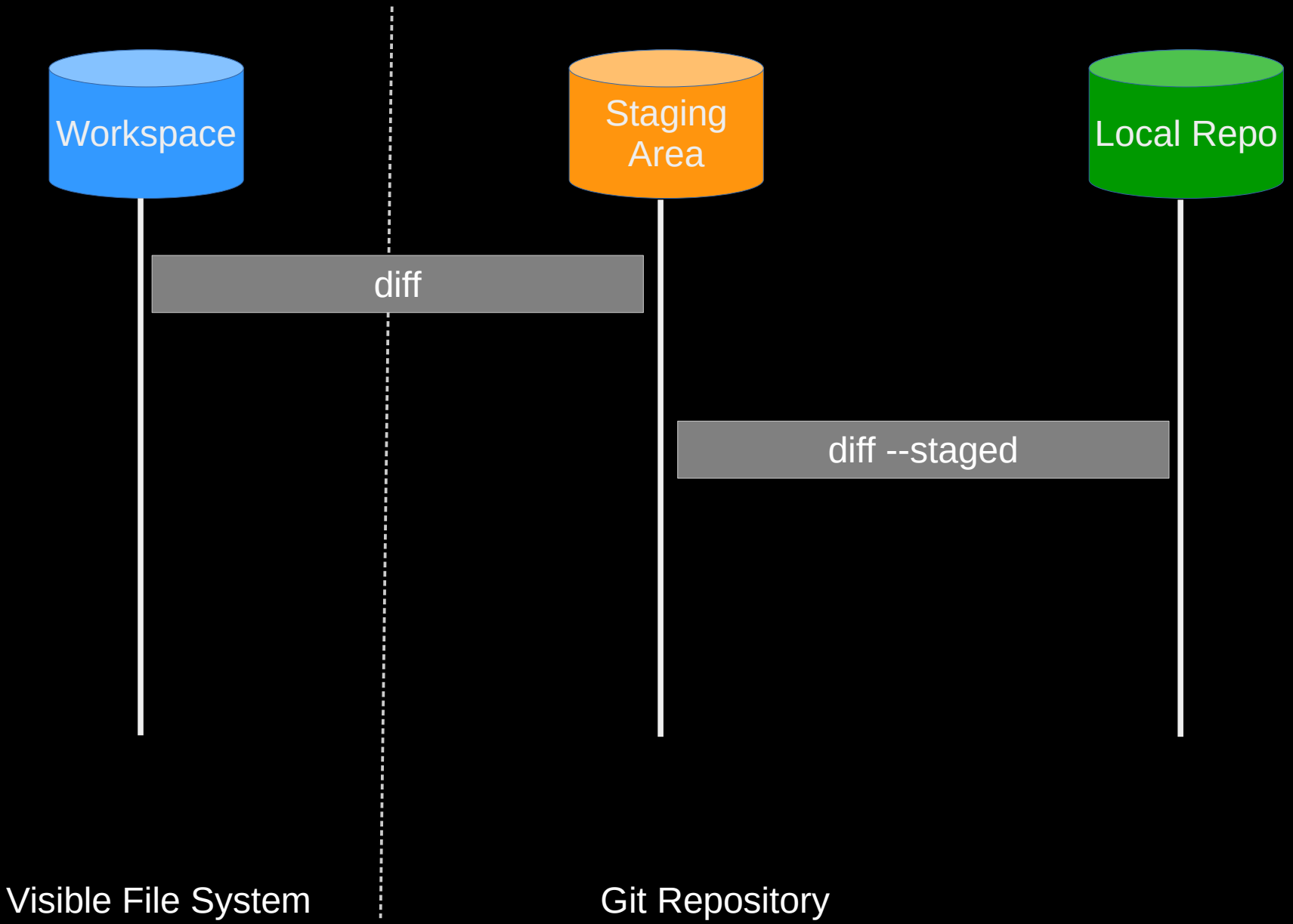
Git – add and commit



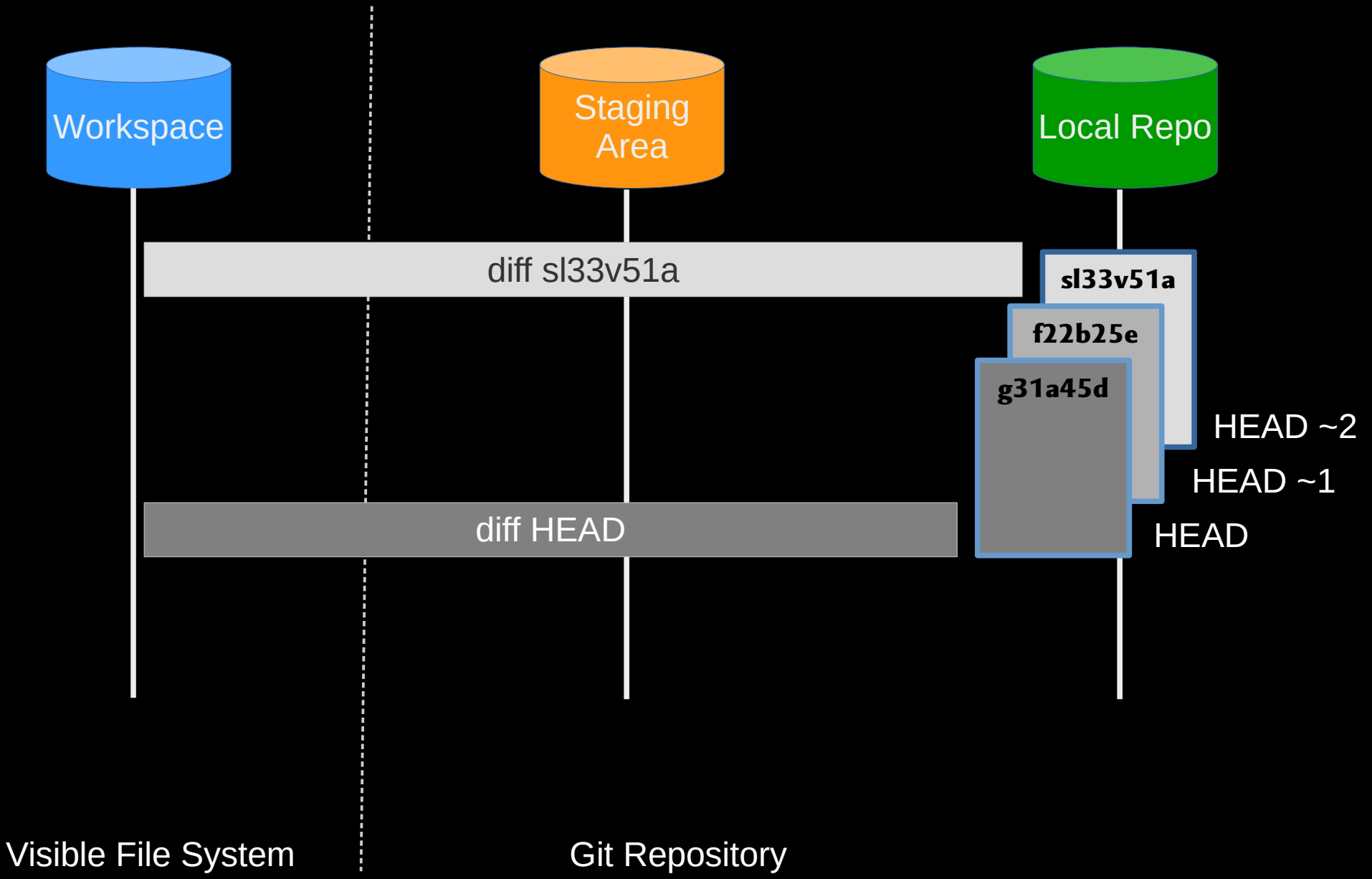
Exploring History #1

- git log
- git diff

Git – diff #1


Workspace
Staging Area
Local Repo
diff
diff --staged
Visible File System
Git Repository

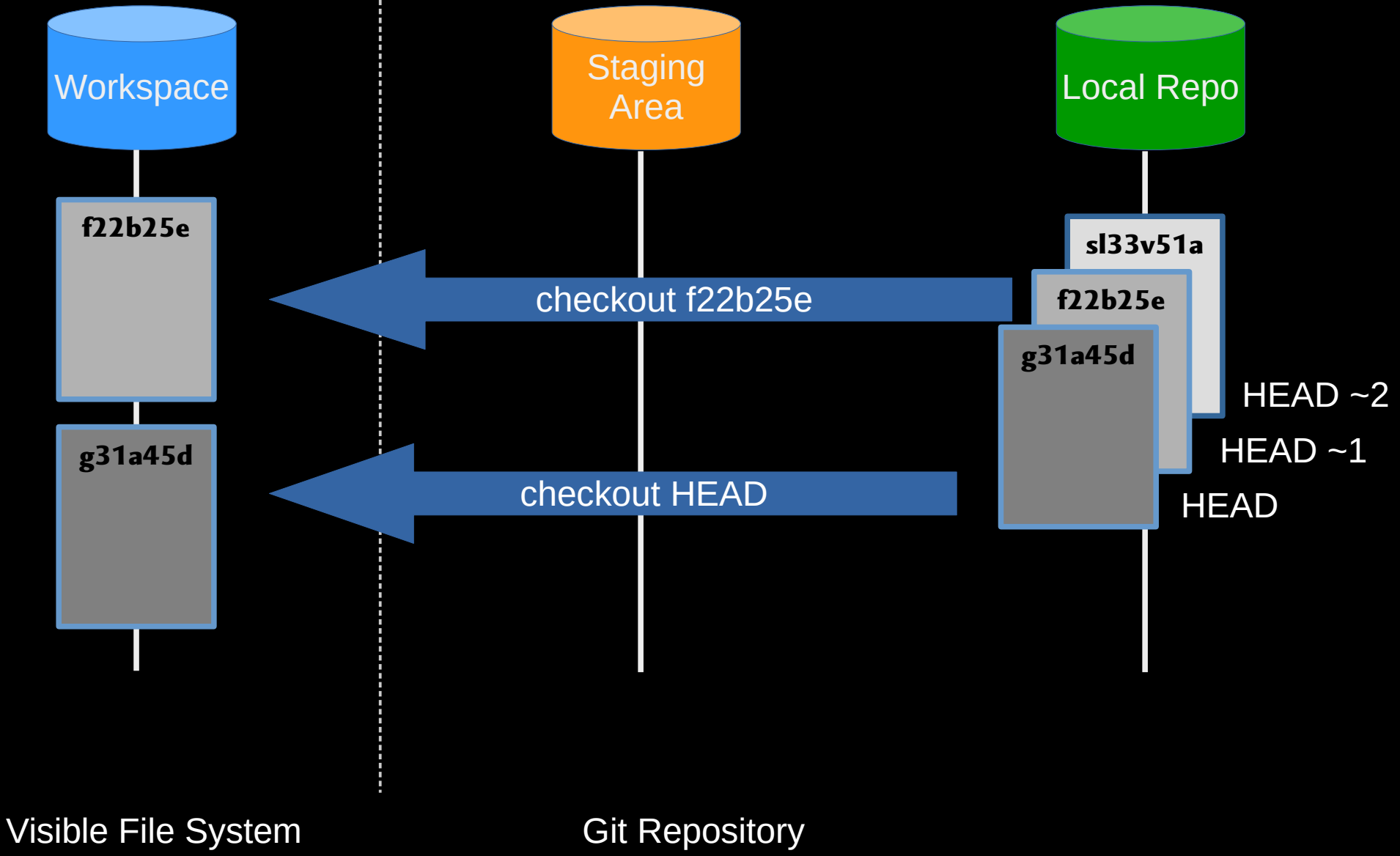
Git – diff #2


Workspace
Staging Area
Local Repo
diff sl33v51a
sl33v51a
f22b25e
g31a45d
HEAD ~2
HEAD ~1
HEAD
diff HEAD
Visible File System
Git Repository

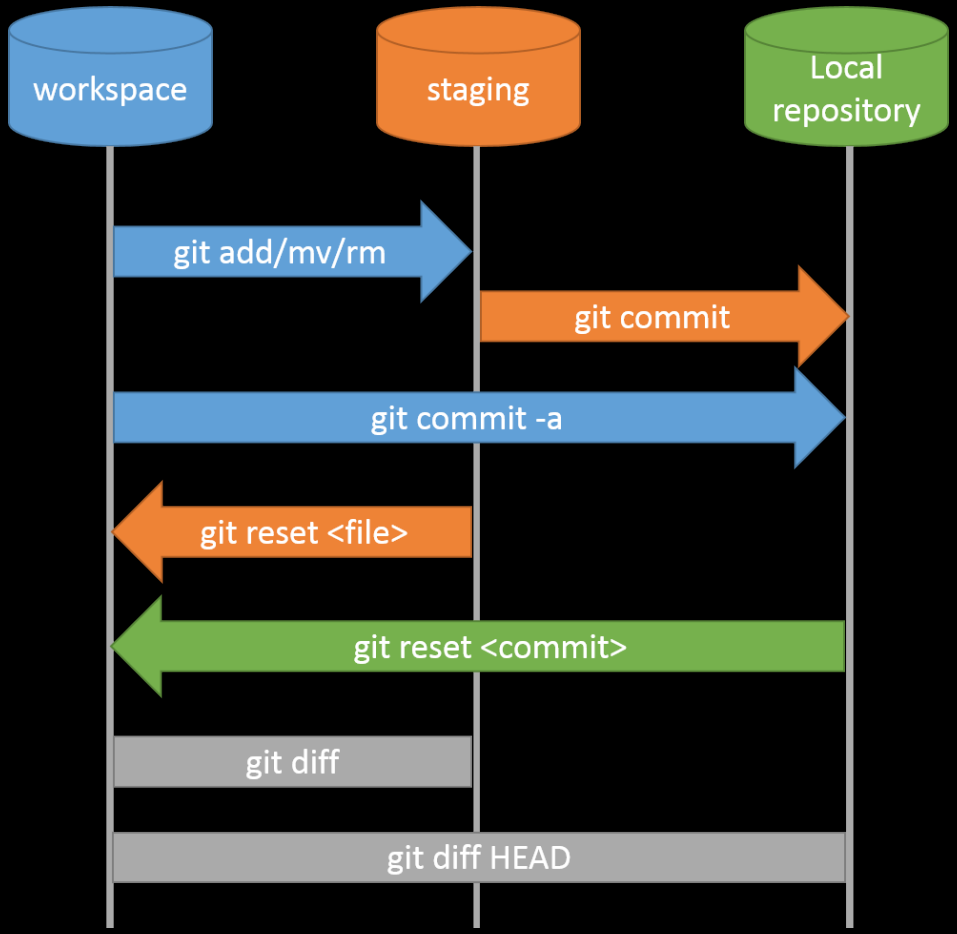
Restoring Files

- git checkout

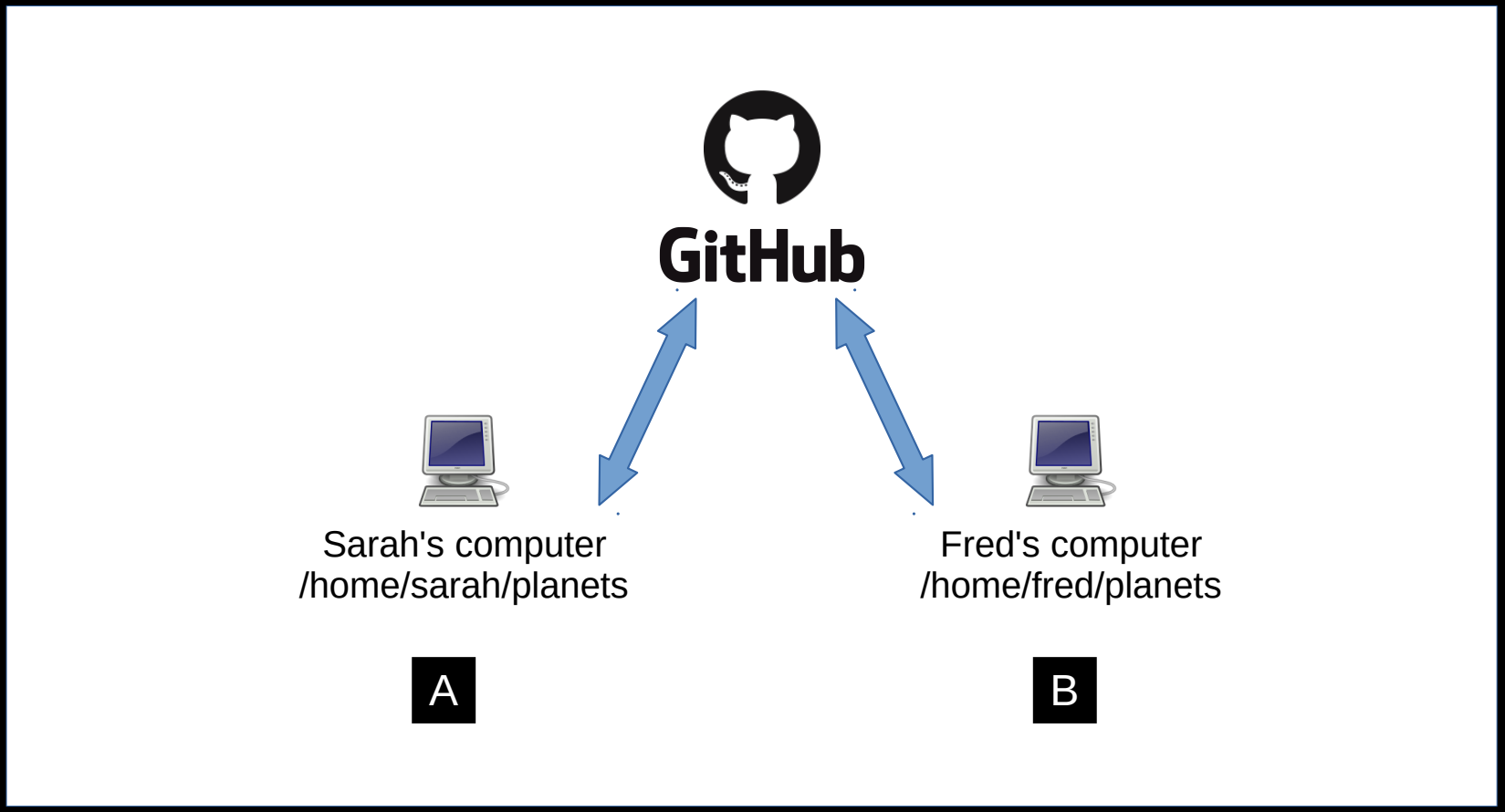
Git - restoration



Git Workflow – Local Repo.



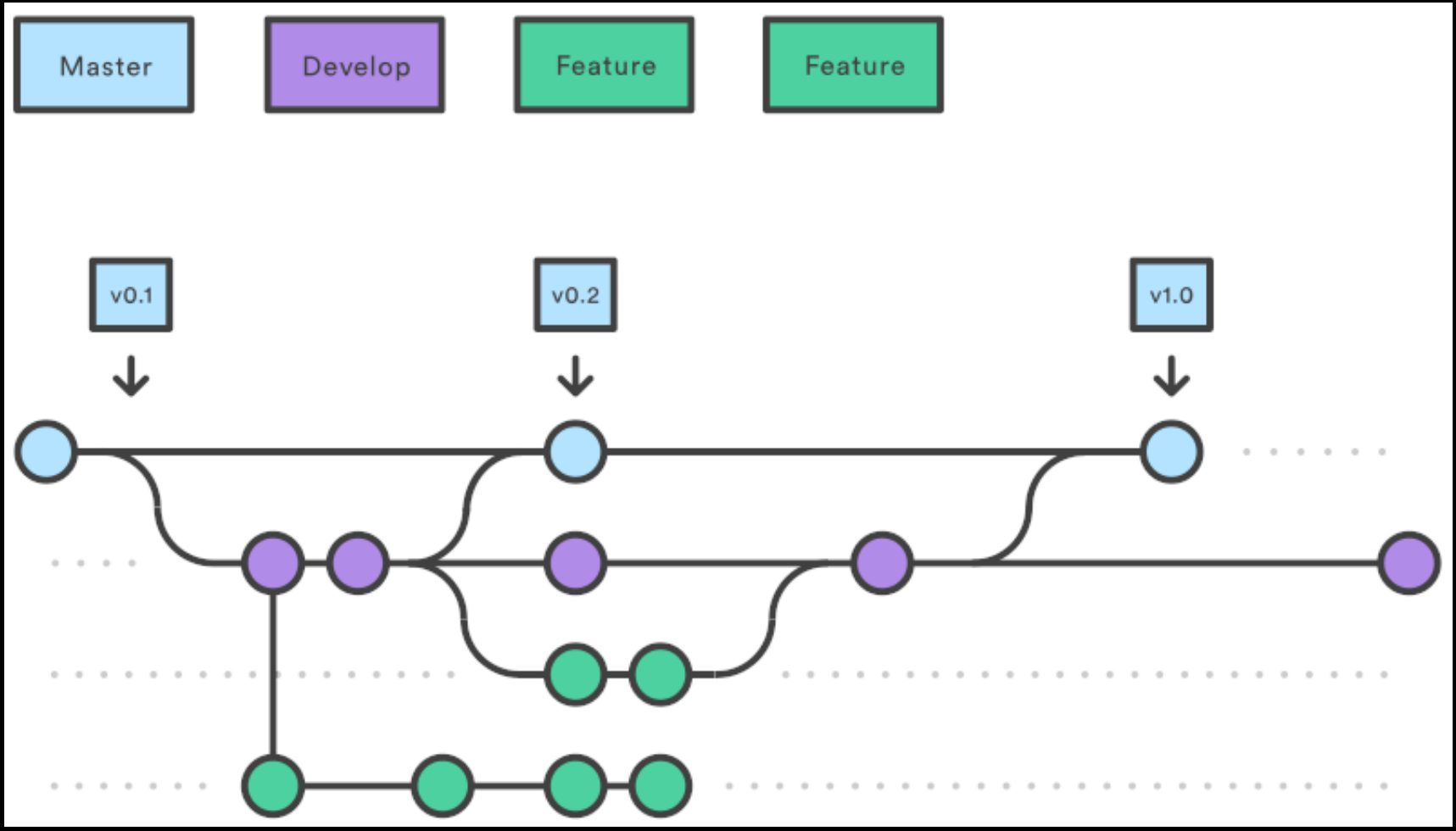
Collaboration



Collaboration: Remote Repositories

- Sign in https://github.com/
- Create repository
- git remote add
- git push

Collaboration: Branches



https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

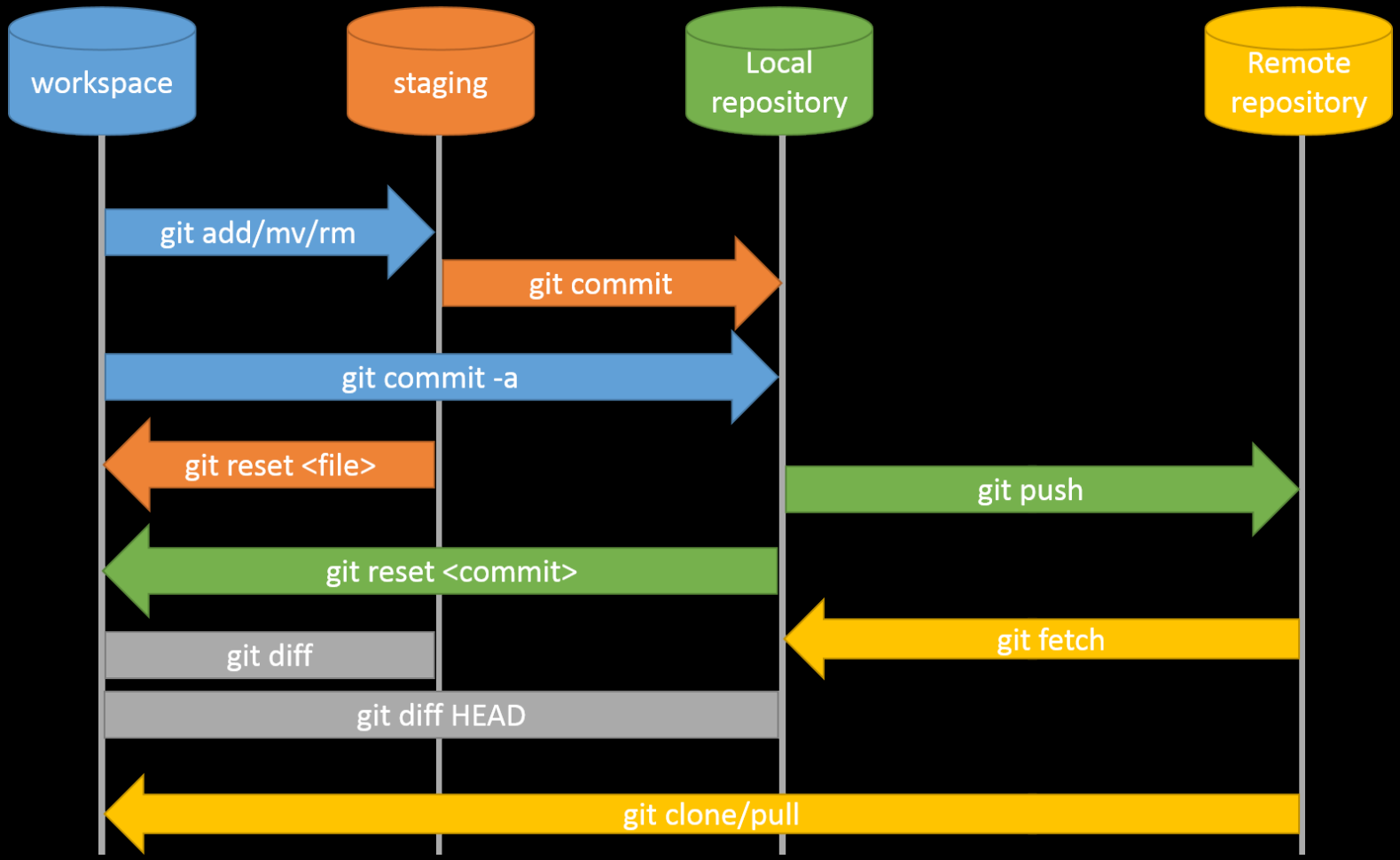
Collaboration: Creating Branches

- git branch dev
- git checkout dev

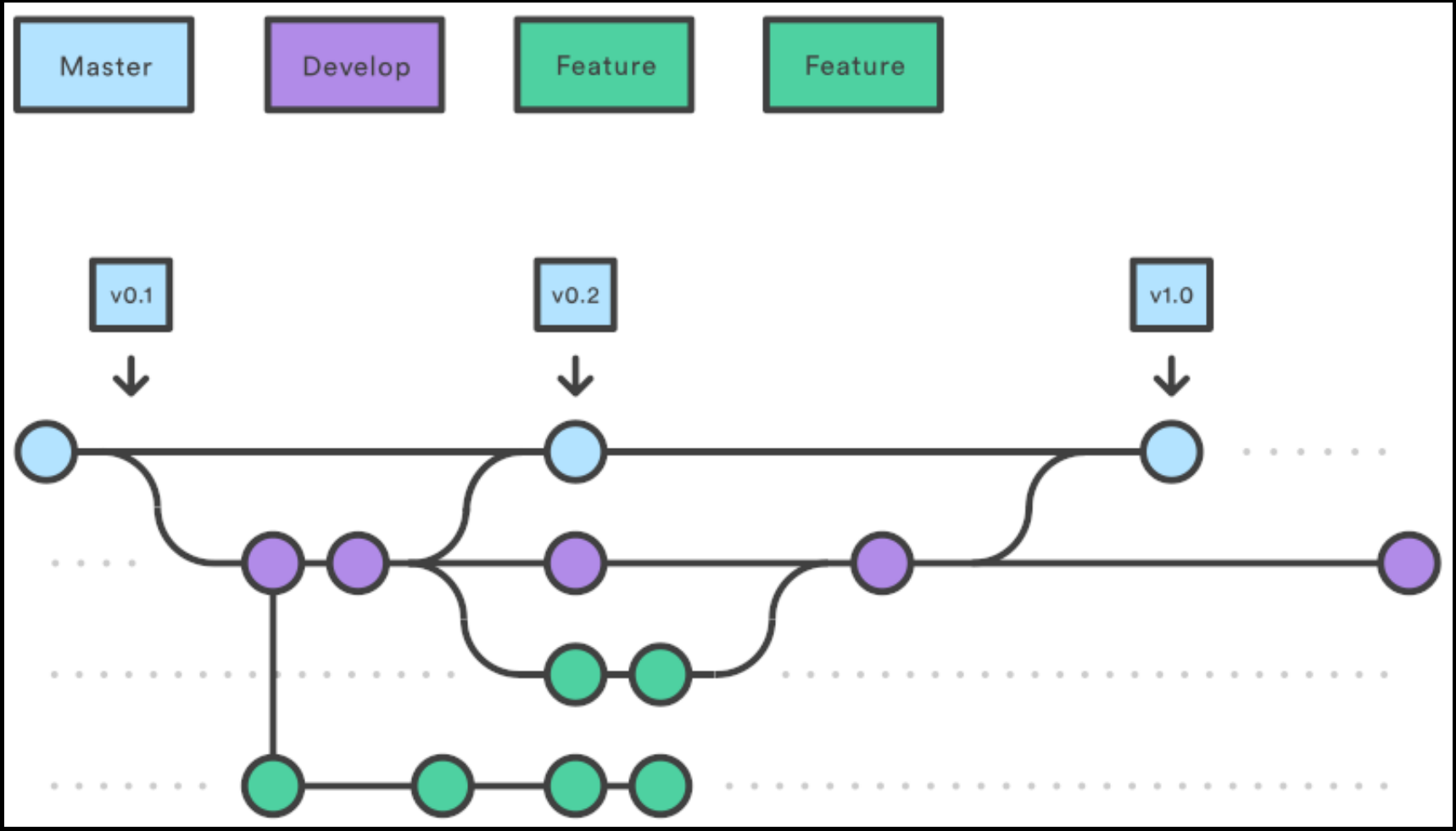
Collaboration: Creating Branches 2

- Create rainfall_conversion.py
- git add rainfall_conversion.py
- git commit -m
- git push -u origin dev

Git Workflow – Remote Repo.



Collaboration: Branches



https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Collaboration: Feature Branch Exercise

- Check out ‘dev’, and create a new branch called ‘docs’
- Create, add and commit README.md
- Push to GitHub and merge back to ‘dev’
- Pull the changes back to your local ‘dev’ branch
- Make sure you consider which branch you're branching *off*, pushing *to*, merging *with* and pulling *from* and *to*.

Collaboration: Feature Branch Solution

- git checkout dev
- git branch docs
- git checkout docs
- nano README.md
- git add README.md
- git commit -m ‘Added documentation’
- git push -u origin docs
- [Pull request on GitHub]
- git checkout dev
- git pull

Collaboration: Summary

- ‘master’ branch is for releases for papers and the public
- ‘dev’ branch is for a stable internal version you and your collaborators share
- Create a new ‘feature’ branch for every new feature or substantial bug

Collaboration: Joining on a Collaboration

- Clone the repository
- Check out ‘dev’ and pull from ‘origin’
- Create a new feature branch for your feature
- Upload and pull request when you’re done!

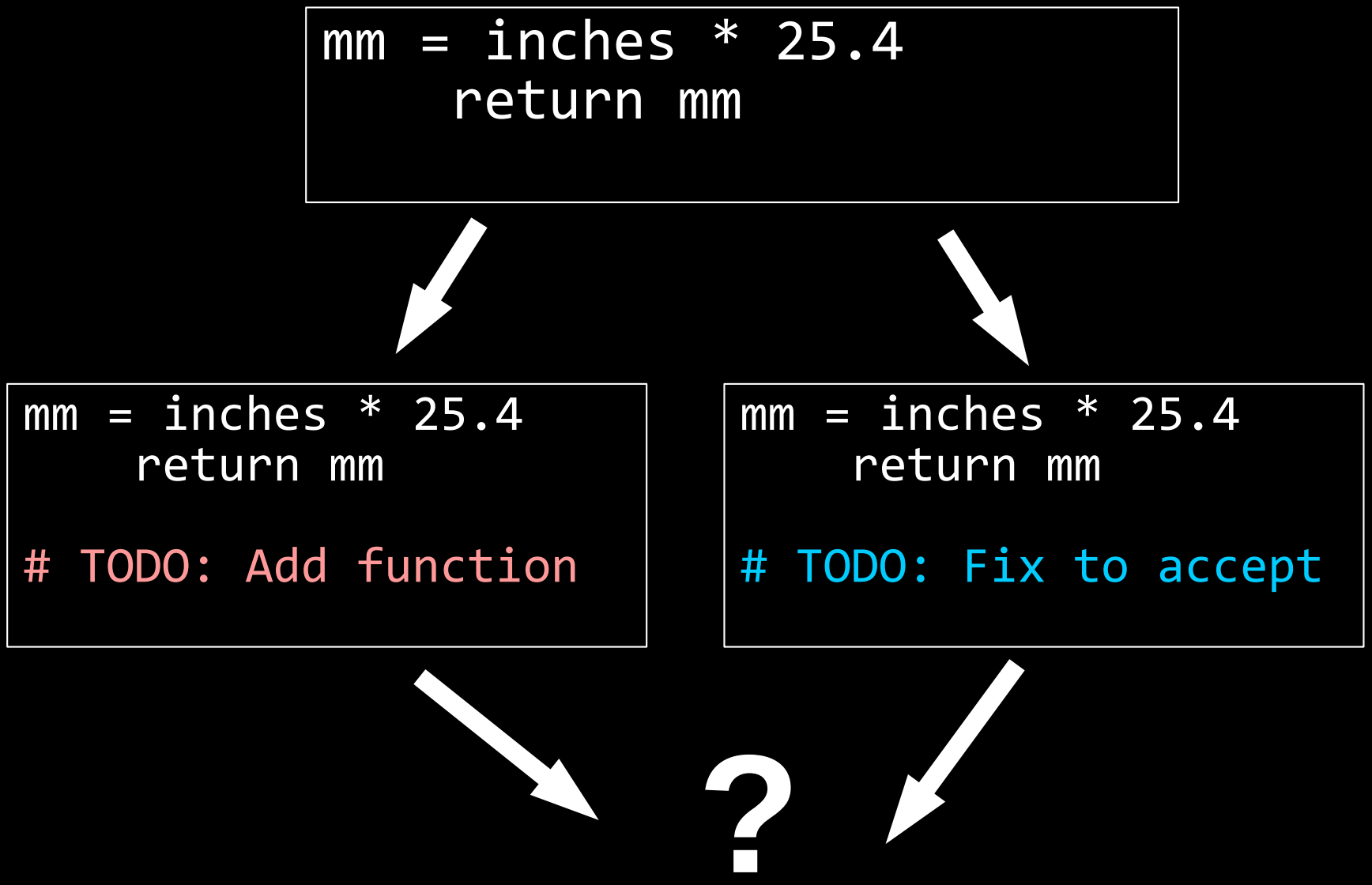
Conflicts: Feature branch

- Check out ‘dev’
- Create a new branch called ‘inches_to_cm’
- Check out ‘inches_to_cm’
- Add a ToDo, commit and push to a new branch on GitHub

Conflicts:
Dev branch

- Check out ‘dev’ again
- Add a ToDo, commit and push to GitHub
- Create a pull request on GitHub from inches_to_cm to dev

Conflicts


mm = inches * 25.4
return mm
mm = inches * 25.4
return mm
# TODO: Add function
mm = inches * 25.4
return mm
TODO: Fix to accept
?

Conflicts: Resolution

- Check out ‘inches_to_cm’ again
- Git pull origin dev
- Fix rainfall_conversion.py, commit and push to GitHub
- Finish the pull request

What next?

- Ignore files / Forks
- https://software-carpentry.org