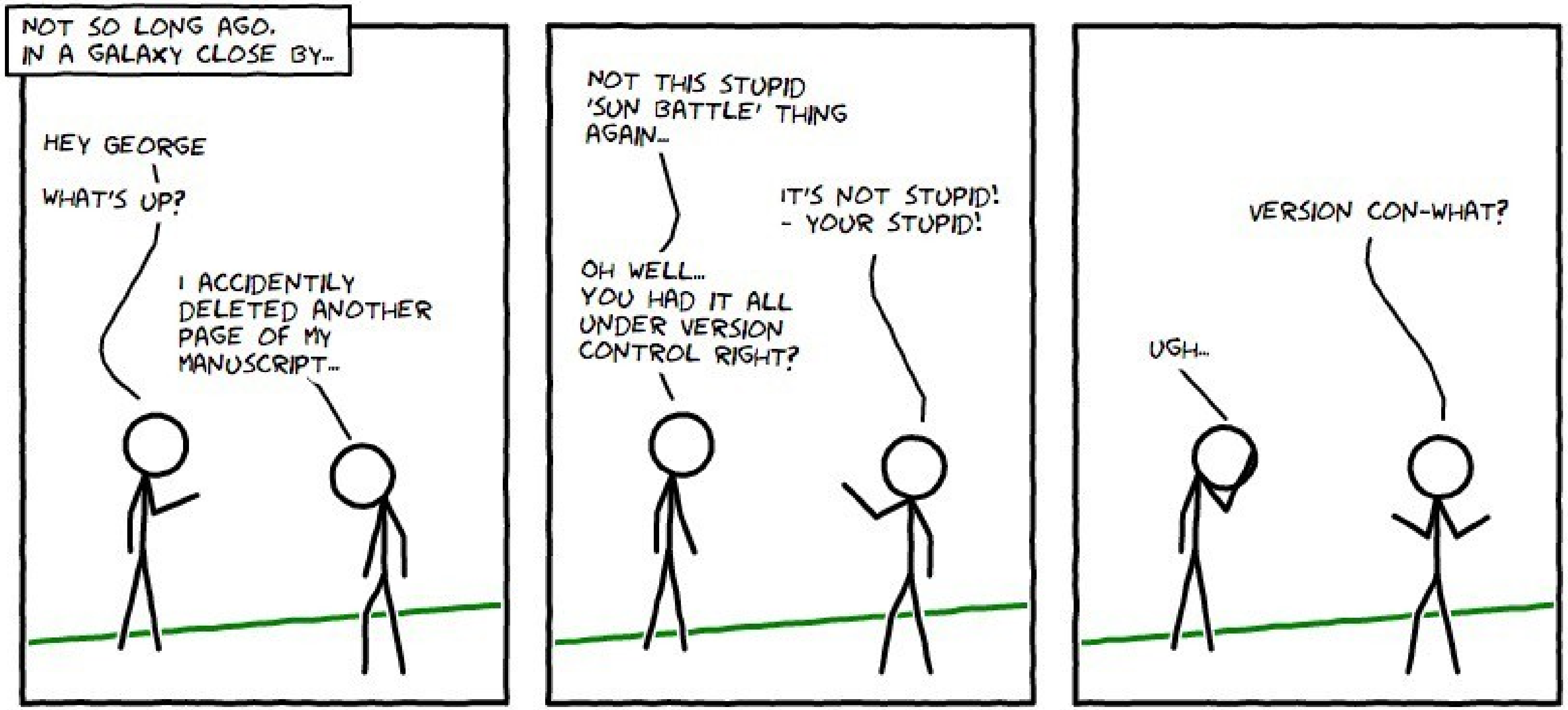
Full Stack Development

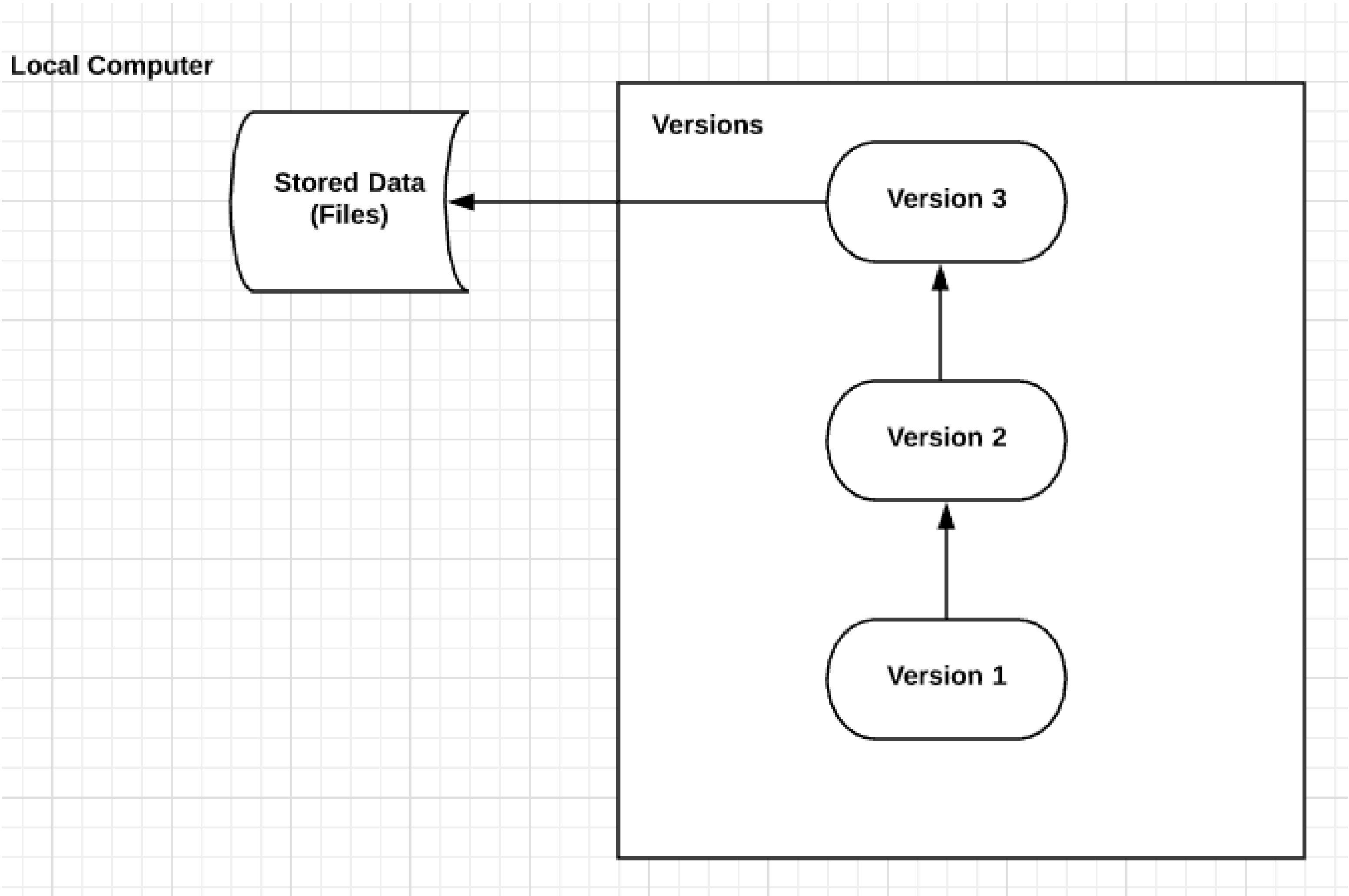
Containers, Microservices and UI

6. Code Management

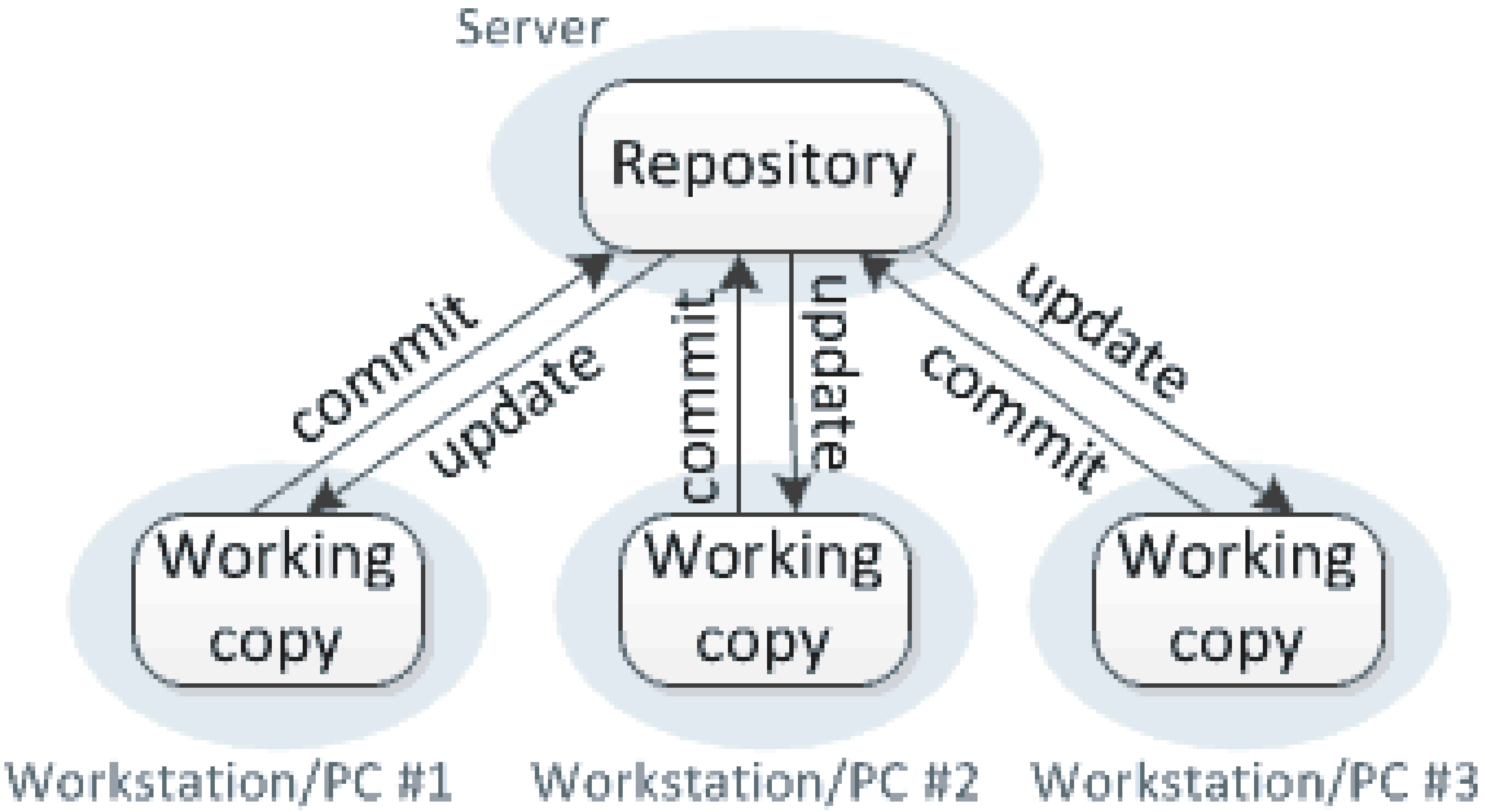
Version Control


NOT SO LONG AGO, IN A GALAXY CLOSE BY...
HEY GEORGE - WHAT'S UP?
I ACCIDENTILY DELETED ANOTHER PAGE OF MY MANUSCRIPT...
NOT THIS STUPID 'SUN BATTLE' THING AGAIN...
OH WELL... YOU HAD IT ALL UNDER VERSION CONTROL RIGHT?
IT'S NOT STUPID! - YOUR STUPID!
UGH...
VERSION CON-WHAT?

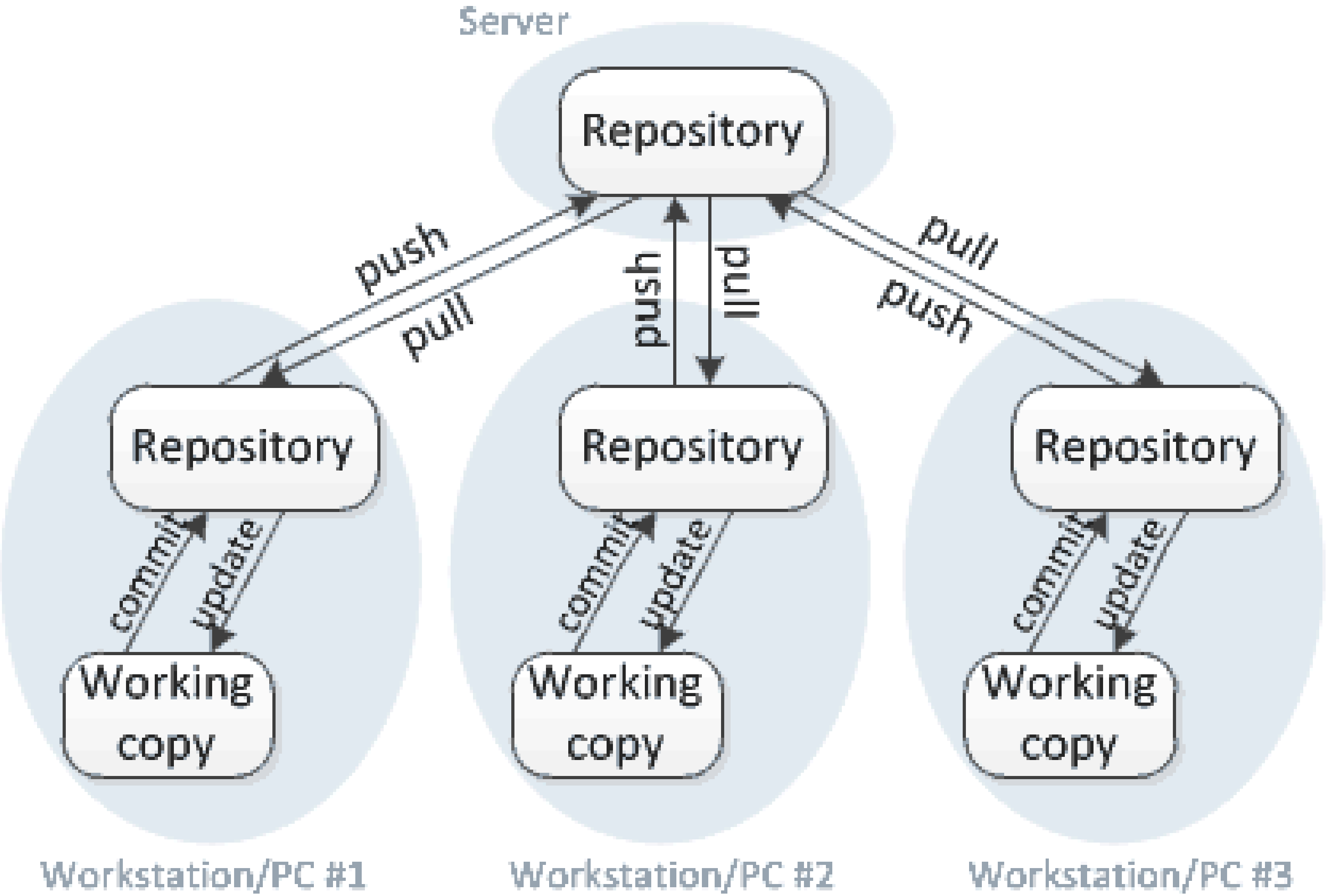
Local Version Control



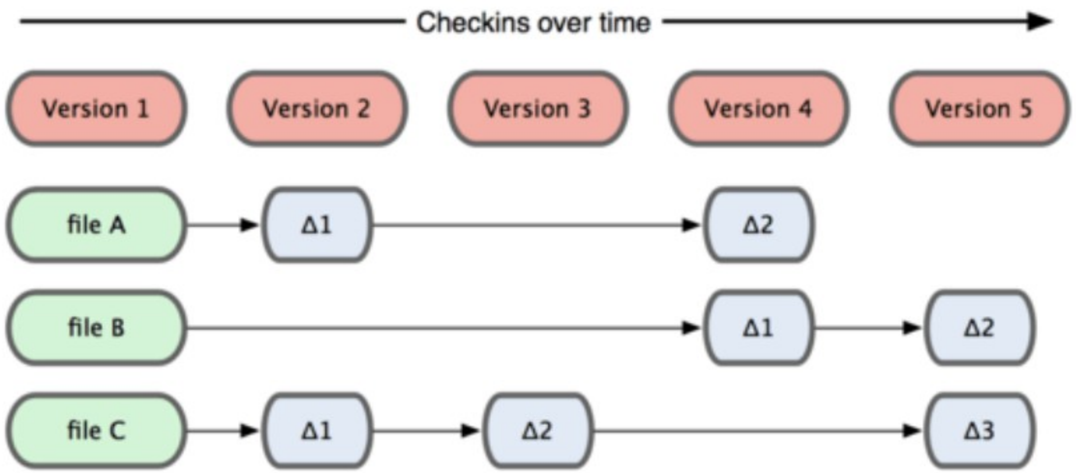
Centralized Version Control



Distributed Version Control



Git Version Model



Checkins over time

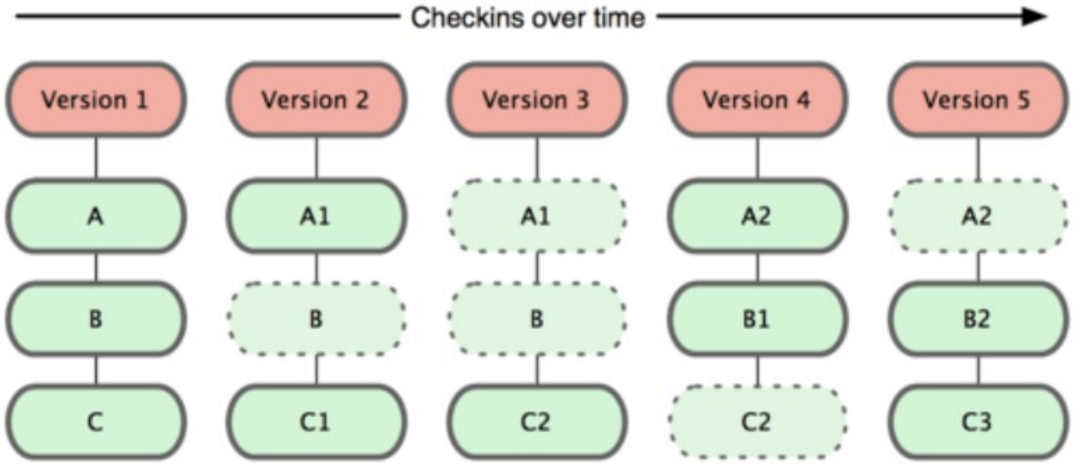
Version 1 Version 2 Version 3 Version 4 Version 5

file A → Δ1 → Δ2

file B → Δ1 → Δ2

file C → Δ1 → Δ2 → Δ3

Other systems



Checkins over time

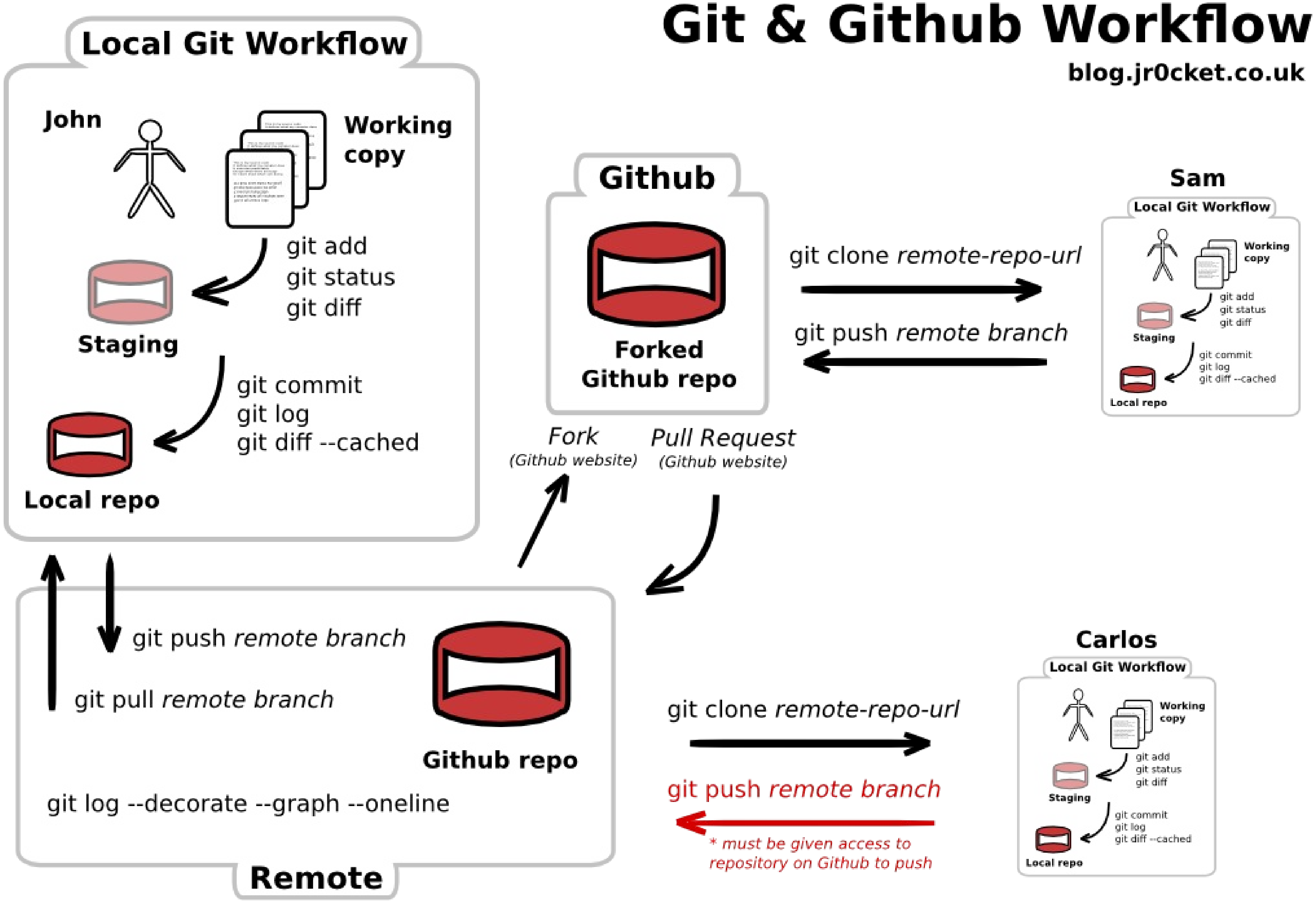
Version 1	Version 2	Version 3	Version 4	Version 5
A	A1	A1	A2	A2
B	B	B	B1	B2
C	C1	C2	C2	C3

Git

Source: Git book

Git & Github Workflow

blog.jr0cket.co.uk



Local Git Workflow

John

Working copy

git add
git status
git diff

Staging

git commit
git log
git diff --cached

Local repo

Github

Forked Github repo

git clone *remote-repo-url*

git push *remote branch*

Sam

Local Git Workflow

Working copy

git add
git status
git diff

Staging

git commit
git log
git diff --cached

Local repo

Fork
(Github website)

Pull Request
(Github website)

git push *remote branch*

git pull *remote branch*

Github repo

git log --decorate --graph --oneline

Remote

git clone *remote-repo-url*

git push *remote branch*

** must be given access to repository on Github to push*

Carlos

Local Git Workflow

Working copy

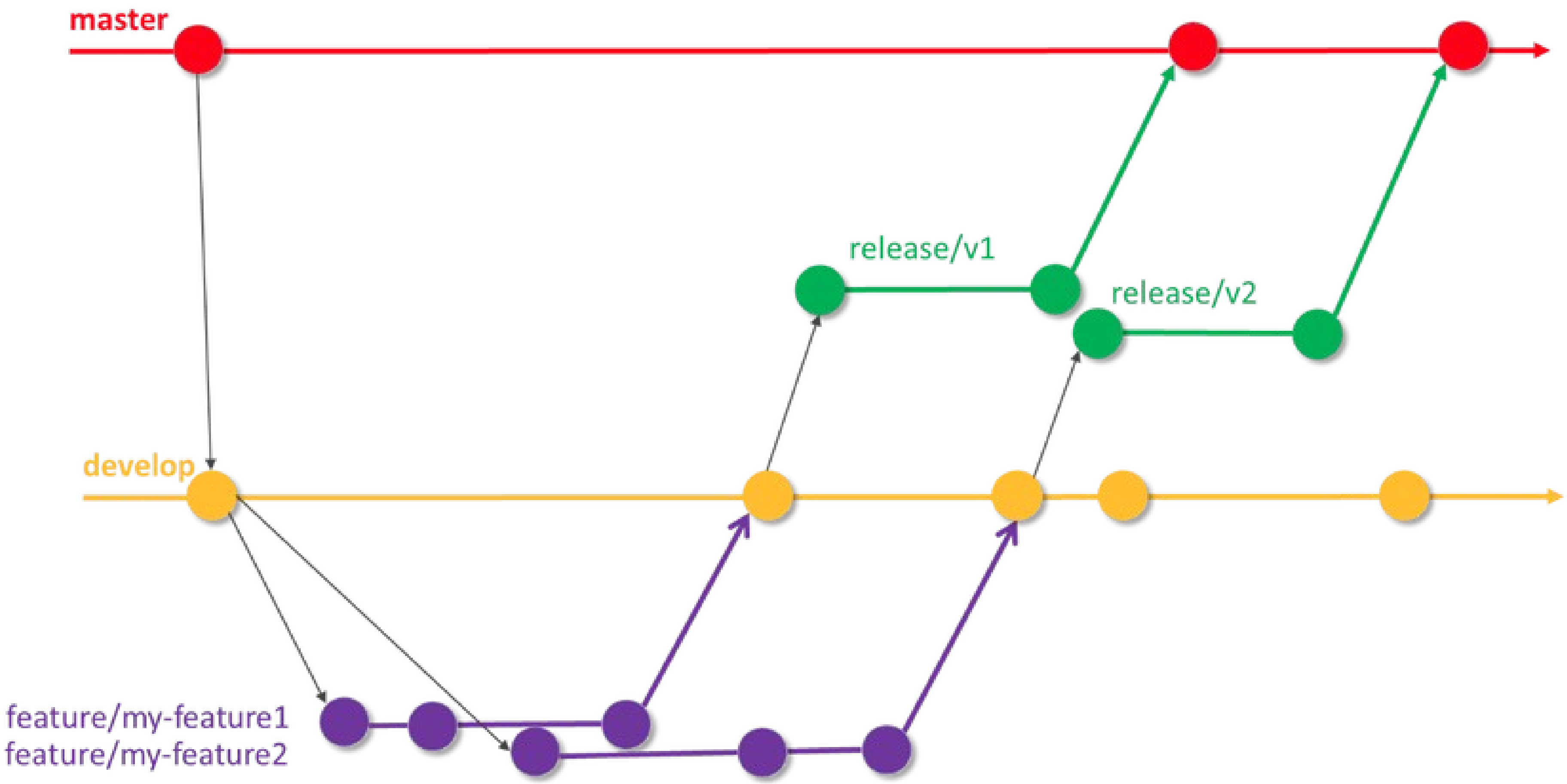
git add
git status
git diff

Staging

git commit
git log
git diff --cached

Local repo

Code Management Branches



Git History

- Prior to 2005: Linux using BitKeeper
- 2005: BitKeeper unfriends Linux
- Linus Torvalds and team design git (named for an uncouth person)
  - Speed
  - Simple design
  - Support for non-linear development
  - Distributed (you can work on the plane)
  - Handle large projects efficiently (speed and data size)
- Not intended to serve as repository for large binary files
  - Main purpose is as code management repository

Terminology

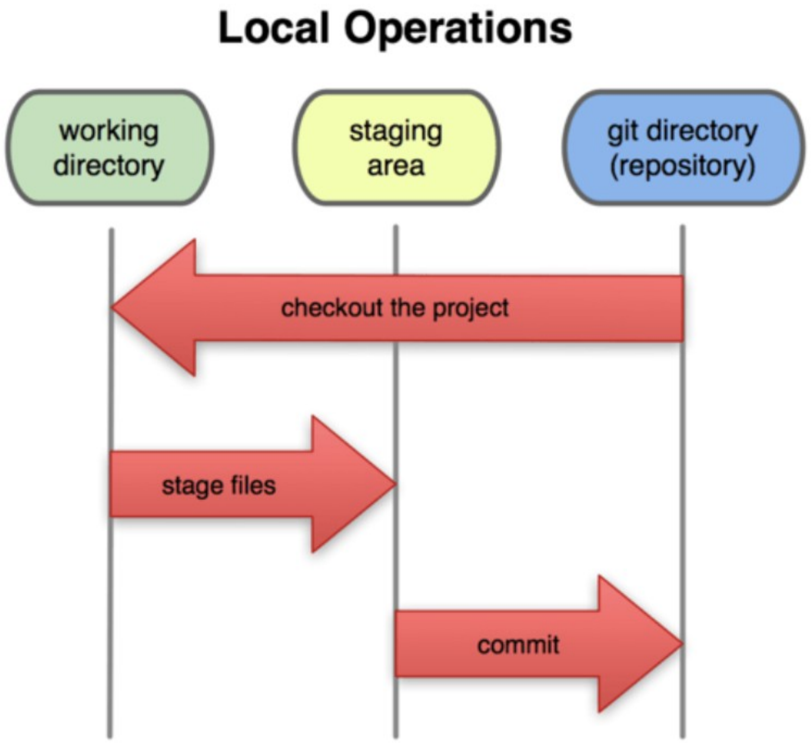
- Repository
- Working Copy
- Index/Staging area
- Blobs, Trees
- Cloning
- Remotes
- Pulling + Pushing
- Local history vs. Public history

Repository

- A set of files and directories
- Historical record of changes in the repository
- A set of commit objects
- A set of references to commit objects, called heads
- Let us give examples of what qualifies as a repository
  - A copy of a project directory?
  - CVS? Subversion?
- Git is a complete repository, either local and remote

Working Copy

- A.k.a "working directory," is a single checkout of one version of the project



Hands-on: analyze the git directory (.git)

Can you have multiple working copies?

Source: Git book

Index and Staging areas

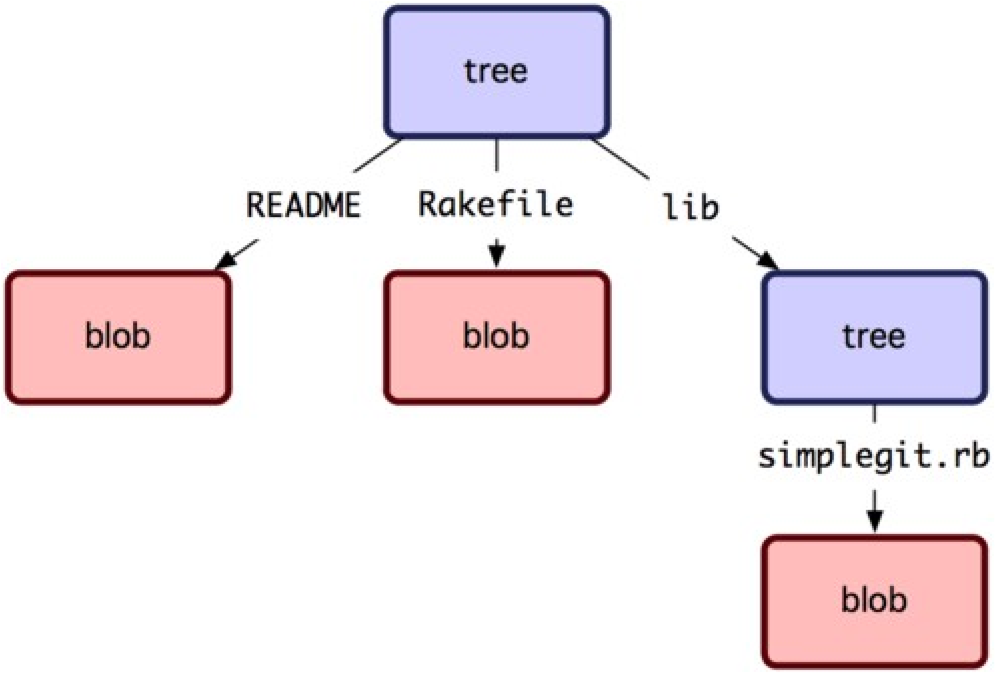
- Index and Staging area are the same
- It is a simple file in the Git directory
- Stores information about the next commit

Basic git Operations

Demo

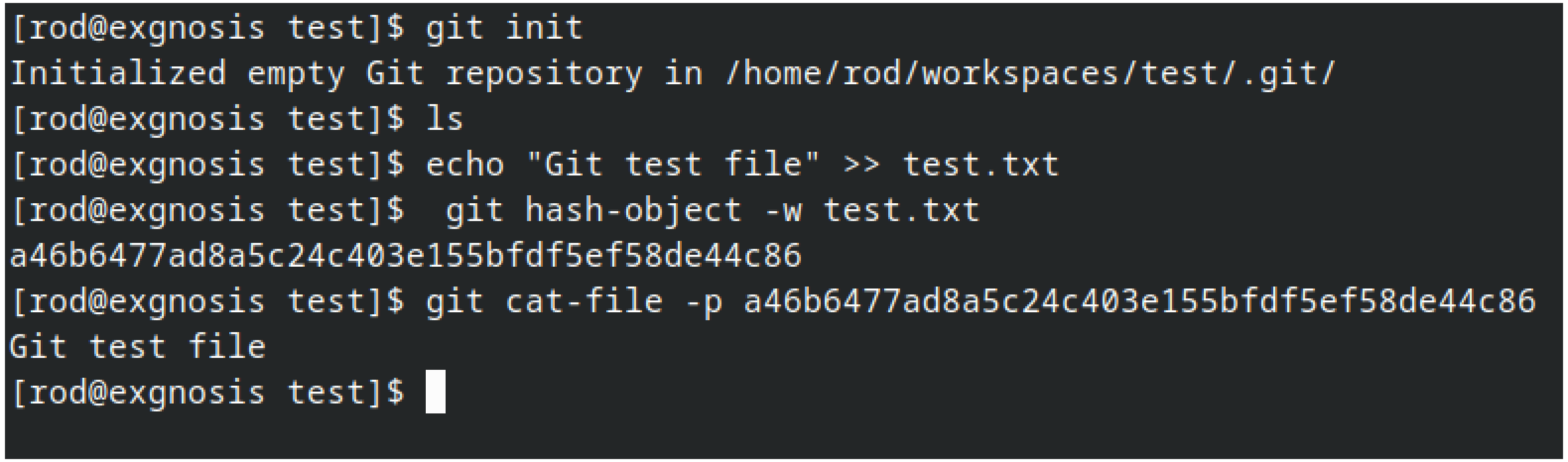
Working Copy

- Git is a key-value data store
  - You can store a value and get back a key
  - All we need to know is “tree” and “blob”



Put and get values

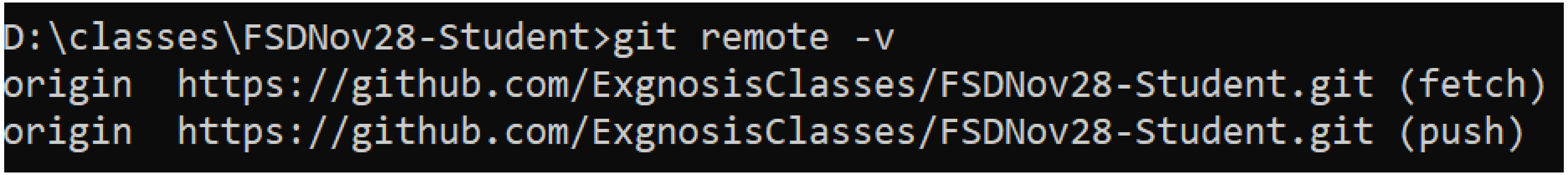
- Put value, observe the key you get in return



```
[rod@exgnosis test]$ git init
Initialized empty Git repository in /home/rod/workspaces/test/.git/
[rod@exgnosis test]$ ls
[rod@exgnosis test]$ echo "Git test file" >> test.txt
[rod@exgnosis test]$  git hash-object -w test.txt
a46b6477ad8a5c24c403e155bfdf5ef58de44c86
[rod@exgnosis test]$ git cat-file -p a46b6477ad8a5c24c403e155bfdf5ef58de44c86
Git test file
[rod@exgnosis test]$ 
```

Cloning and Remotes

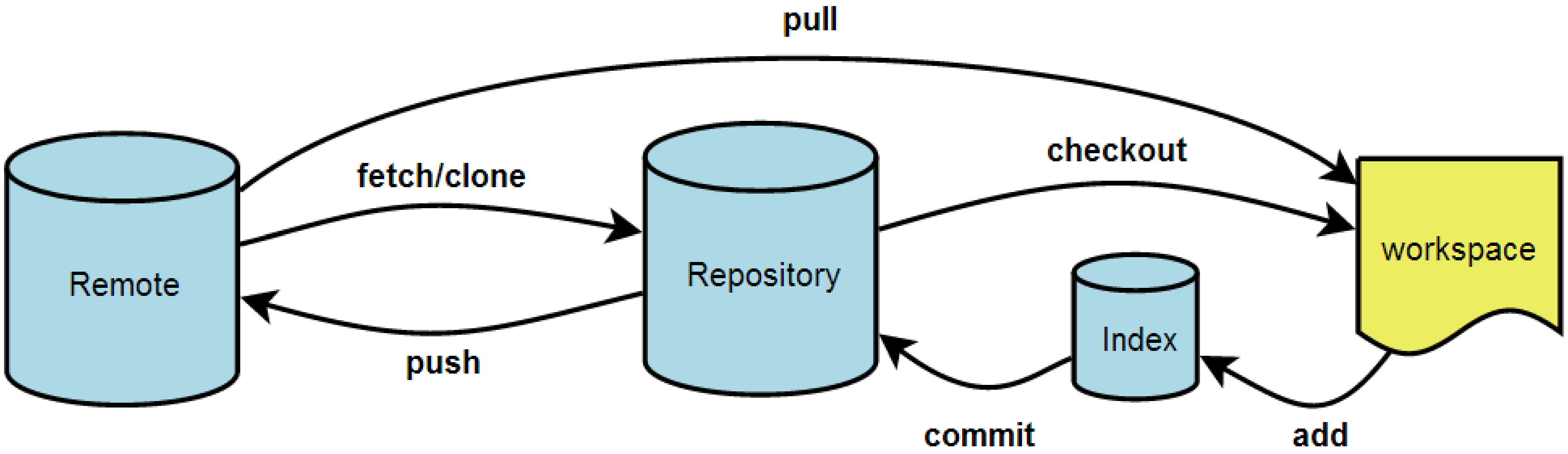
- Getting a copy of the existing get repository
  - How? git clone <url>
  - Also supported by most IDEs
- Remotes are versions of a project that are hosted on the Internet or network for collaboration:
  - There can be multiple remotes
  - Remotes can read only or read-write
  - List remotes for a repository with “git remote -v”
  - The origin remote is where the repo is cloned from



```
D:\classes\FSDNov28-Student>git remote -v
origin  https://github.com/ExgnosisClasses/FSDNov28-Student.git (fetch)
origin  https://github.com/ExgnosisClasses/FSDNov28-Student.git (push)
```

Pulling + Pushing

- Pulling – from a branch on a remote
- Fetching – all that you don’t have yet
- Pushing – back to the branch on a remote

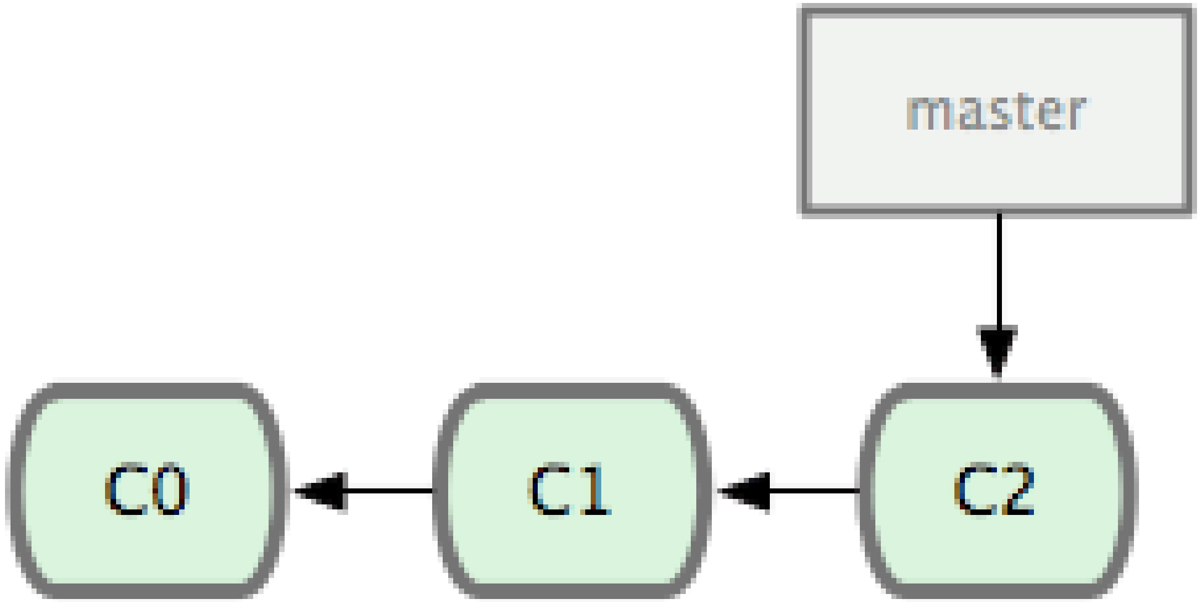


Local history vs. Public history

- Local history is on your computer and allows you to
  - Change commits
  - Change commit messages
  - Reorder
  - Squash
- However, be careful pushing this to the public history
  - Other developers may end up having to merge

Making a commit

- Commit is a record of your changes in a Git directory (repository)
- Making a commit is moving the branch point (master in this case) to the next snapshot



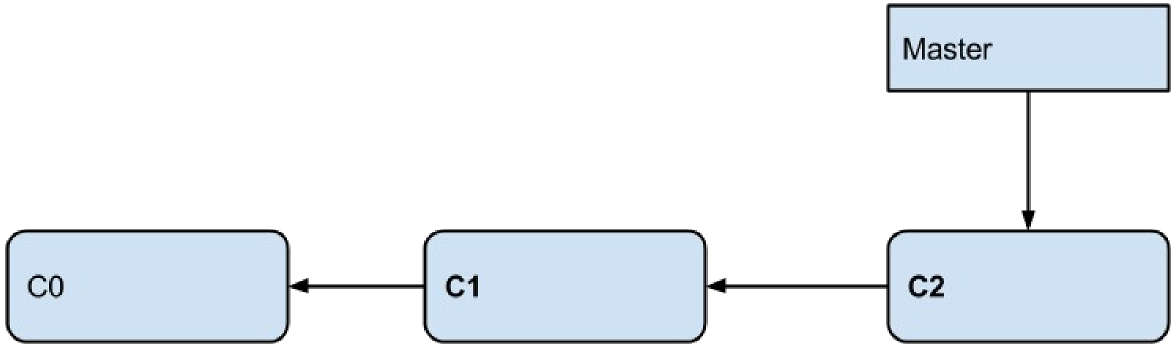
Commit features

- Permanence
  - Commit leaves a record
  - Commit goes into the Git area
  - Commit can be further recorded in a remote
- Impermanence
  - Commits can be taken back (undone locally or reverted)
  - Commits can be erased (rebase)

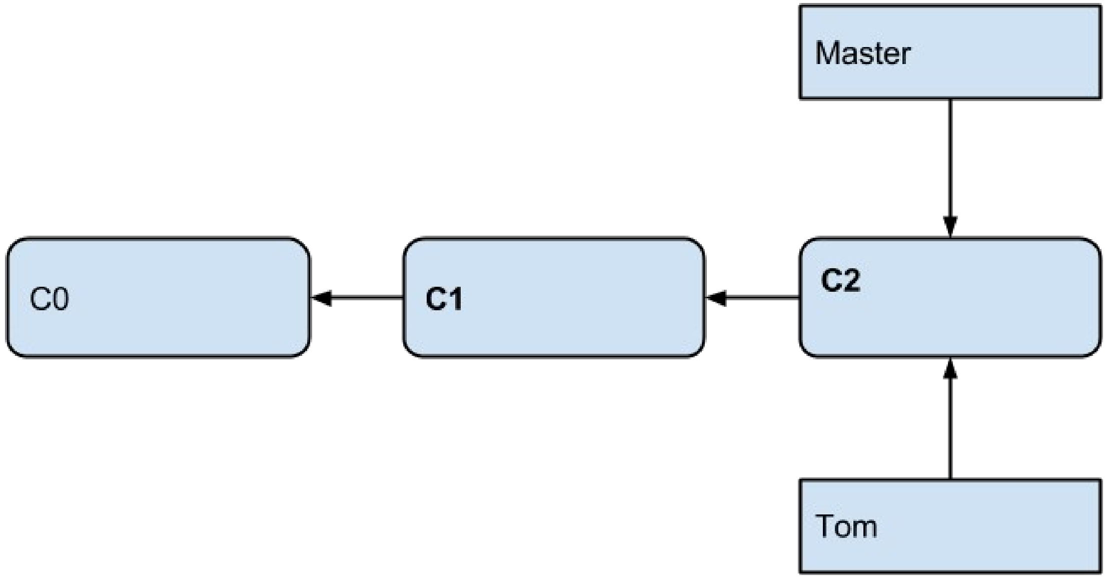
Basic git Commands

Demo

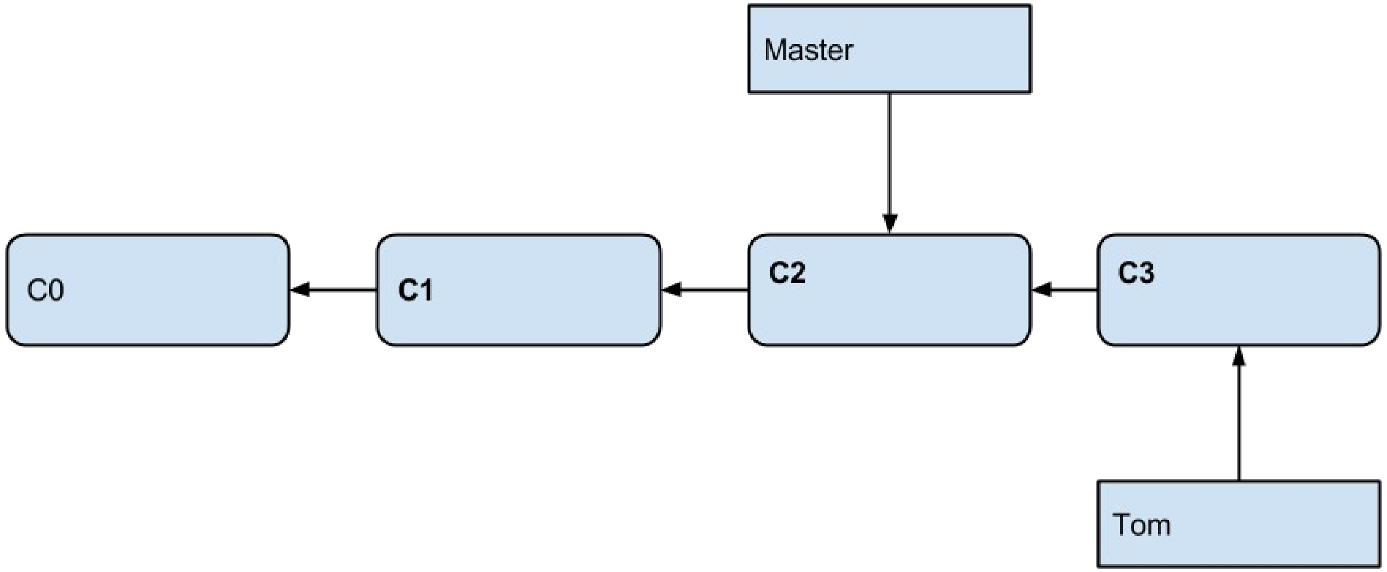
Branching and Merging



Git checkout tom -b

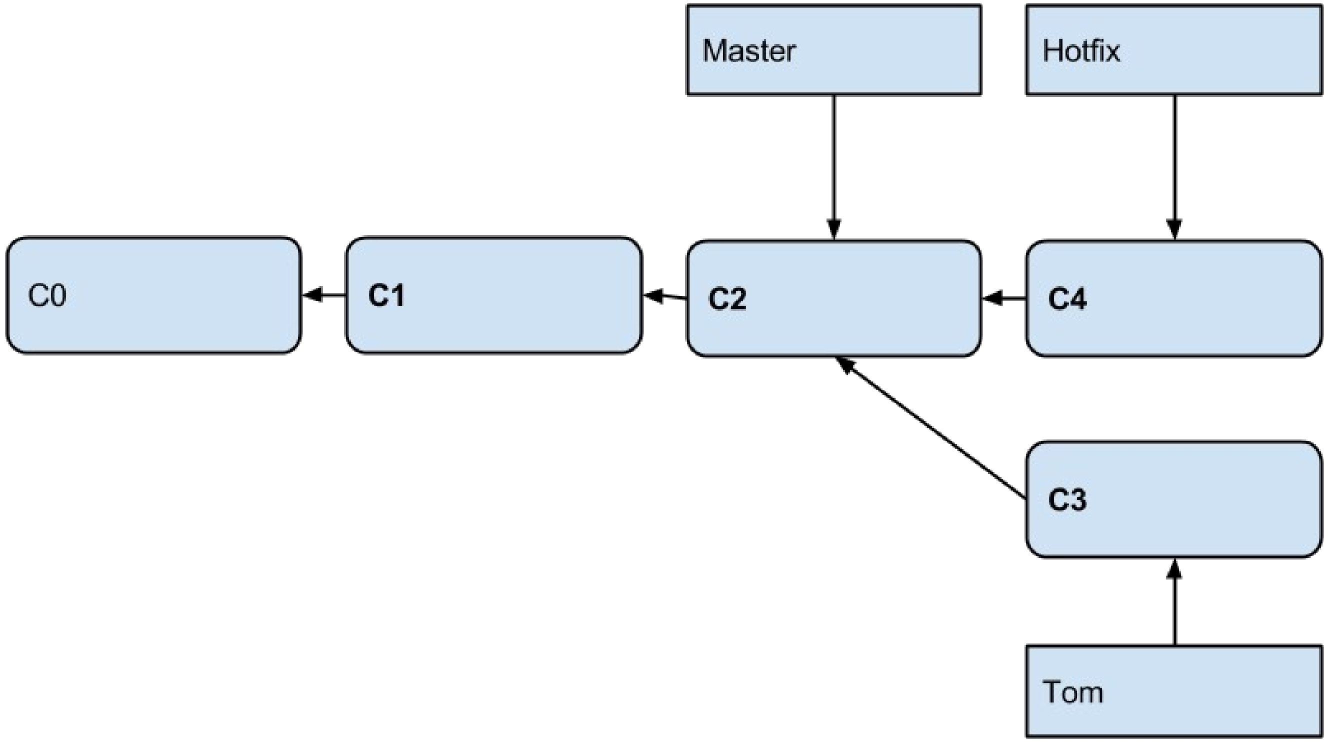


Git commit -a



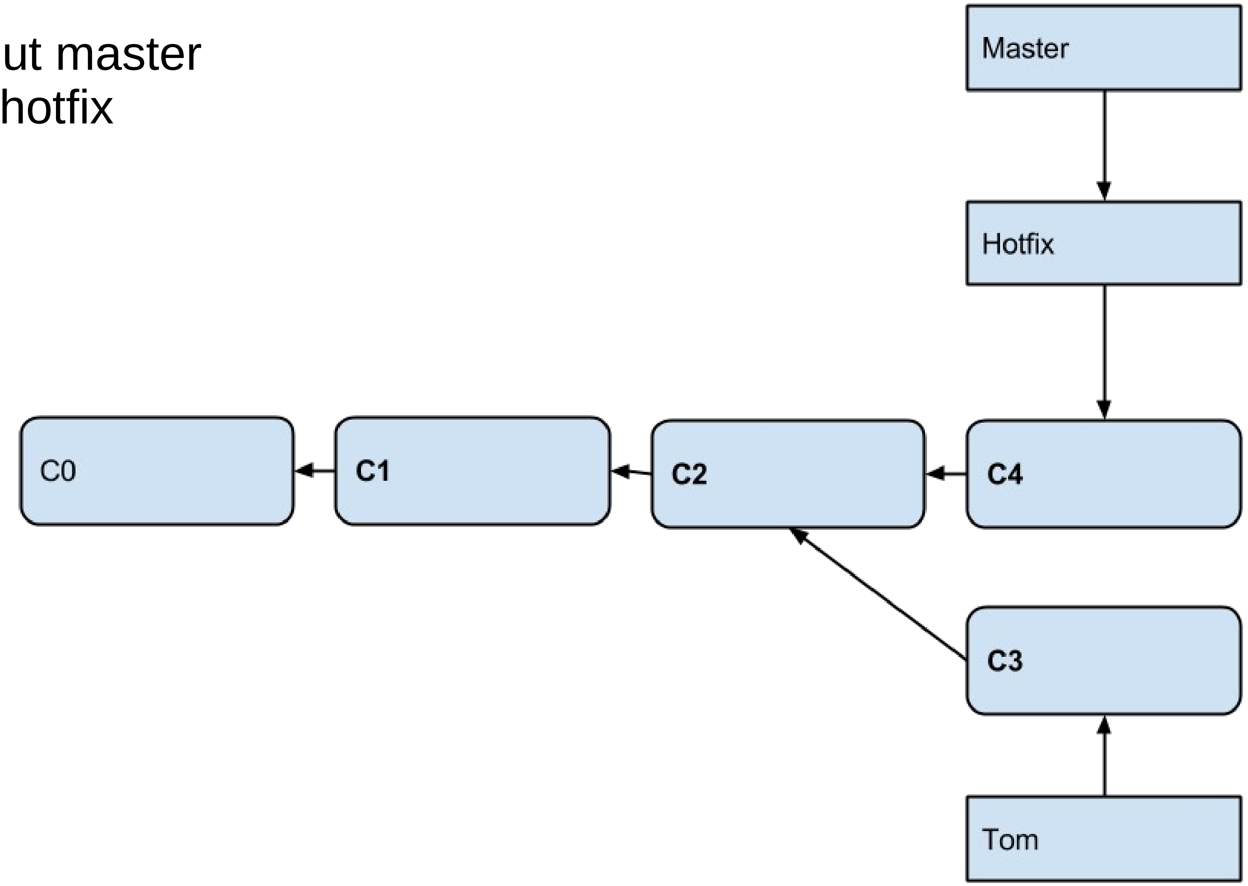
Work on hotfix

```
git checkout -b hotfix
git commit -a -m 'urgent fix'
```



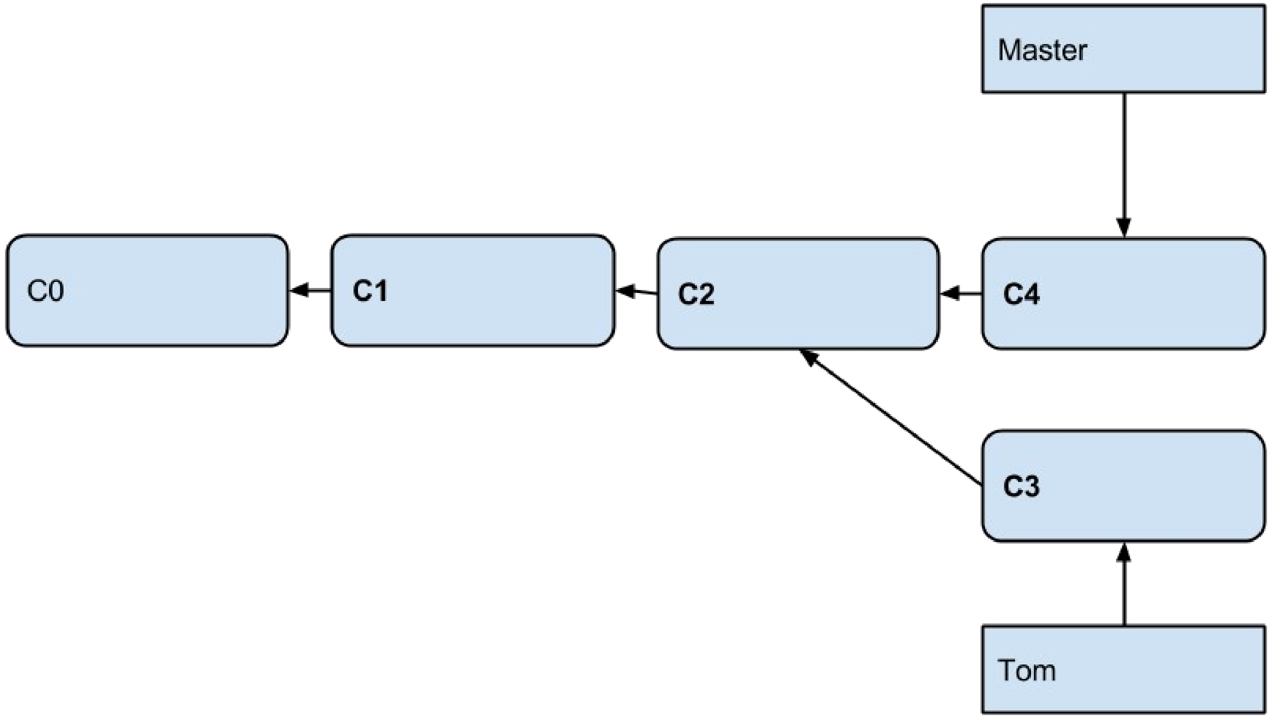
Merge hotfix

```
git checkout master
git merge hotfix
```



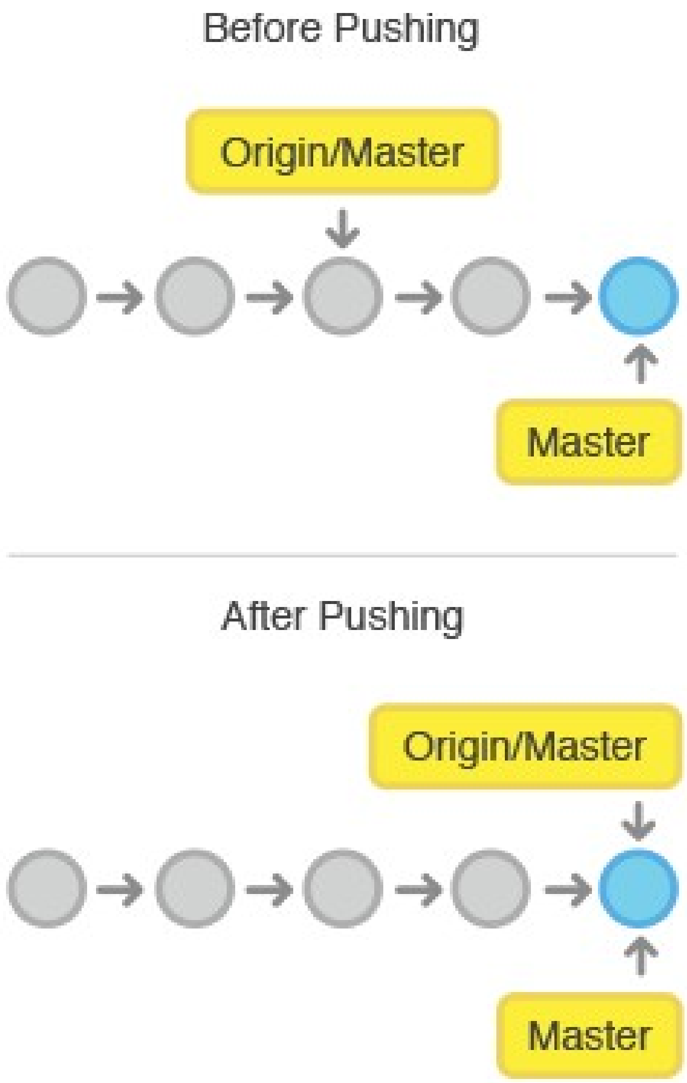
Cleanup

git branch -d hotfix



Pushing your change

```
git push <remote> <branch>
```



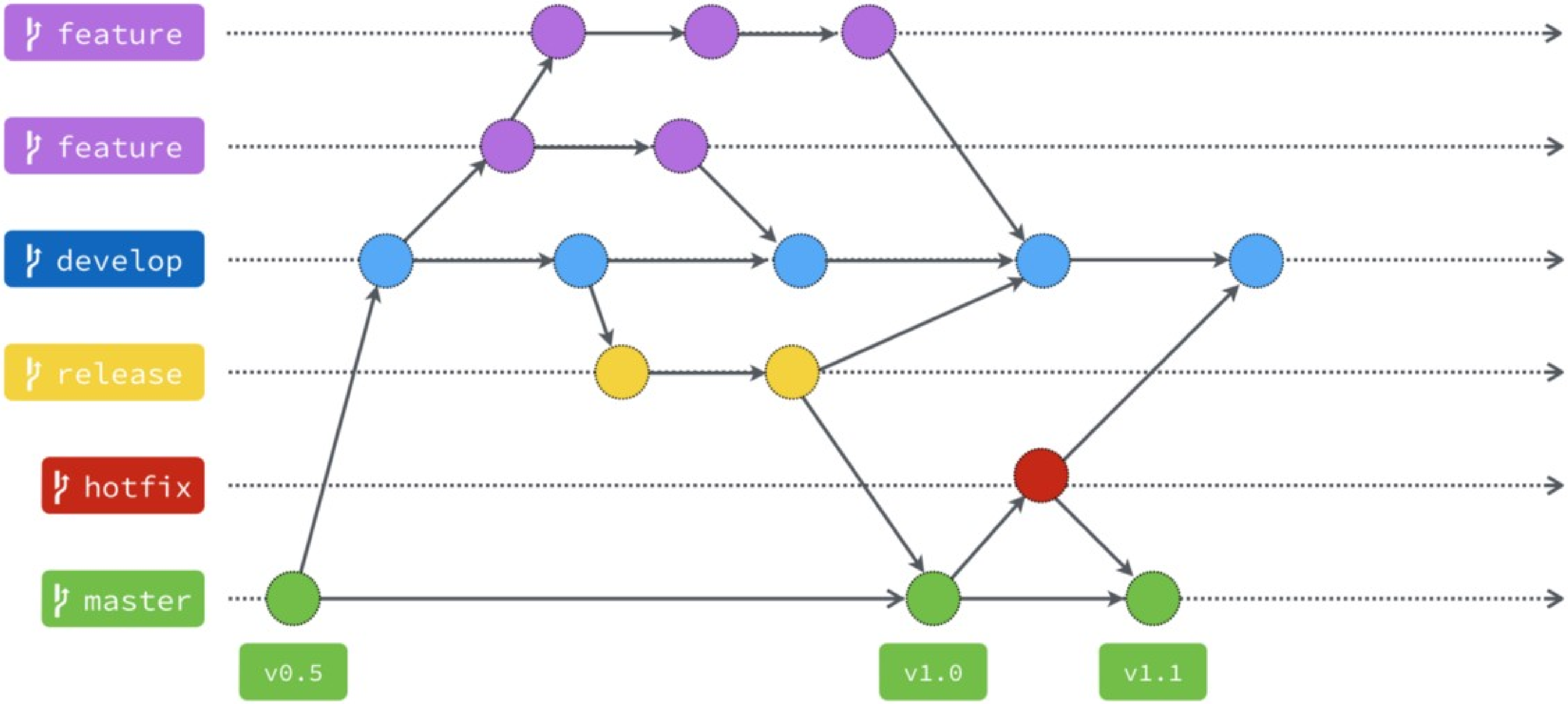
Branching and Merging

Demo

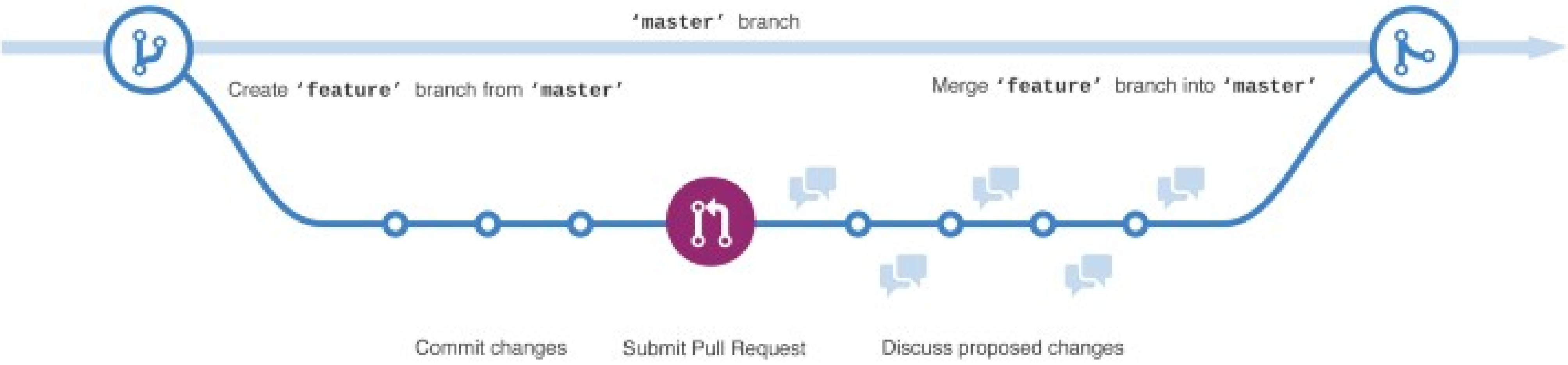
Branching Strategies

- A branching workflow is how developers:
  - Work in parallel on separate tasks and
  - Integrate their work into a codebase
- These are implementations of development models
  - All development on main branch; or
  - All development on feature branches
- All rely on branch and merge events
  - Generally, merges are the events that initiate a CICCD pipeline
- There are three main flows used
  - GitHub flow
  - Git flow
  - GitLab flow

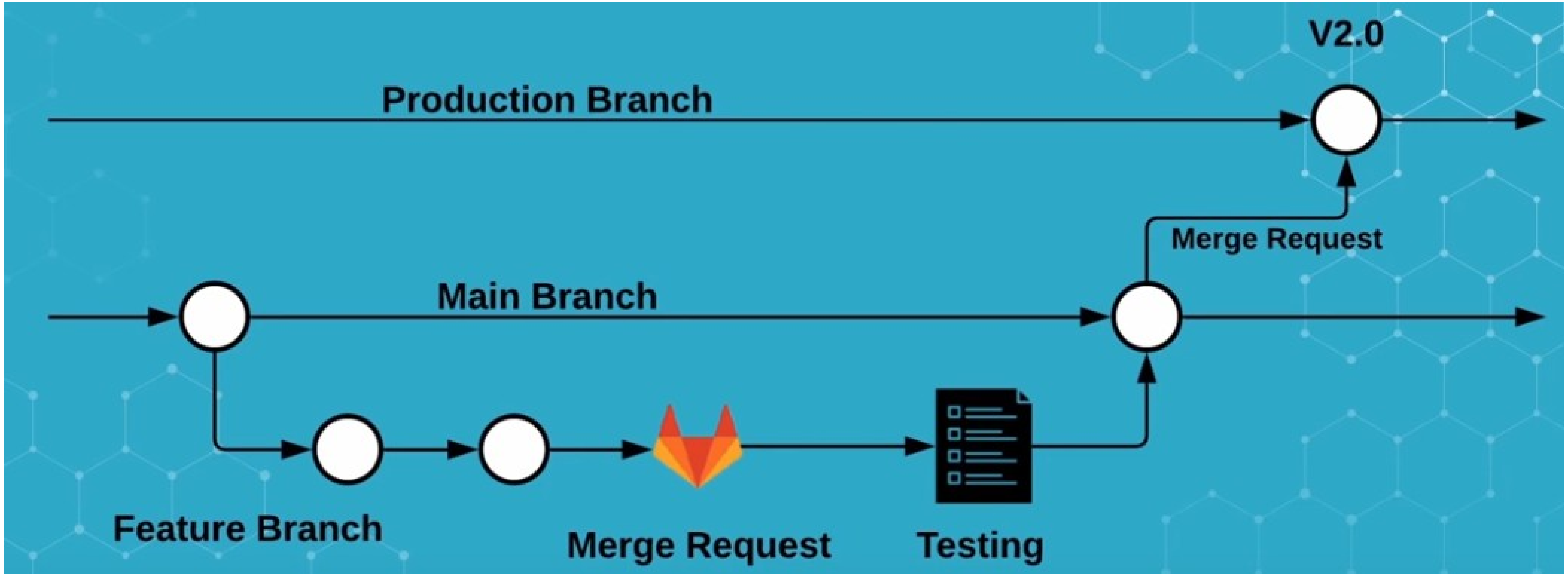
Git Flow



GitHub Flow


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

GitLab Flow

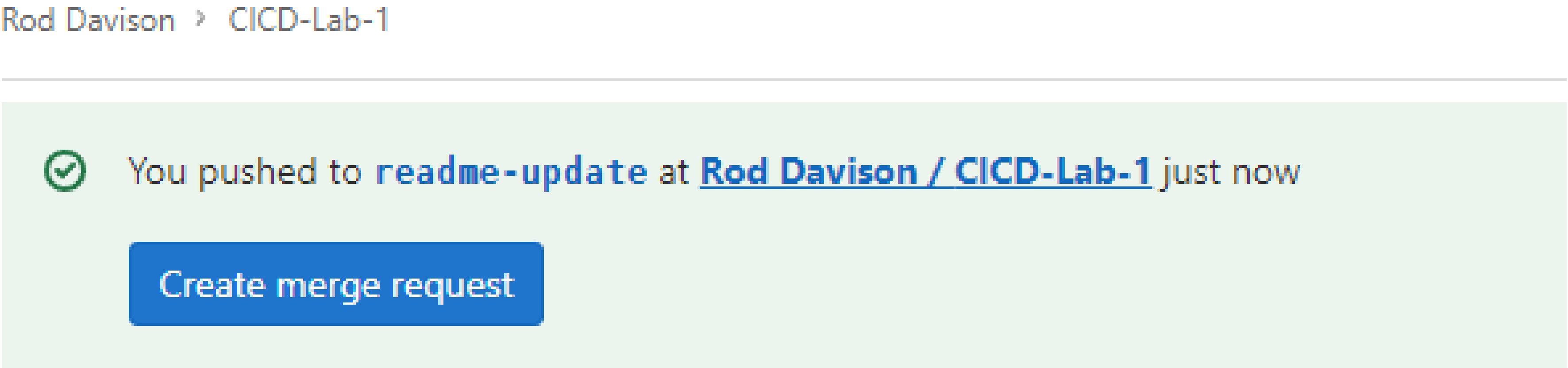

V2.0
Production Branch
Merge Request
Main Branch
Feature Branch
Merge Request
Testing

Feature Branch Workflow

- The main branch is protected
  - Only authorized members can push or merge
- To do any work, create a feature branch
  - These branches should not be created in a remote repository
- Clone the main branch to a local directory
  - Create the feature branch
  - Make changes, commit to the feature branch
  - Push the feature branch to the origin
  - The feature branch will remain until it is merged into the main branch

Feature Branch Merge

- The feature branch has to be merged into main by creating a merge request
- The feature branch can be deleted after the merge is done
- Feature branches should never be long lived



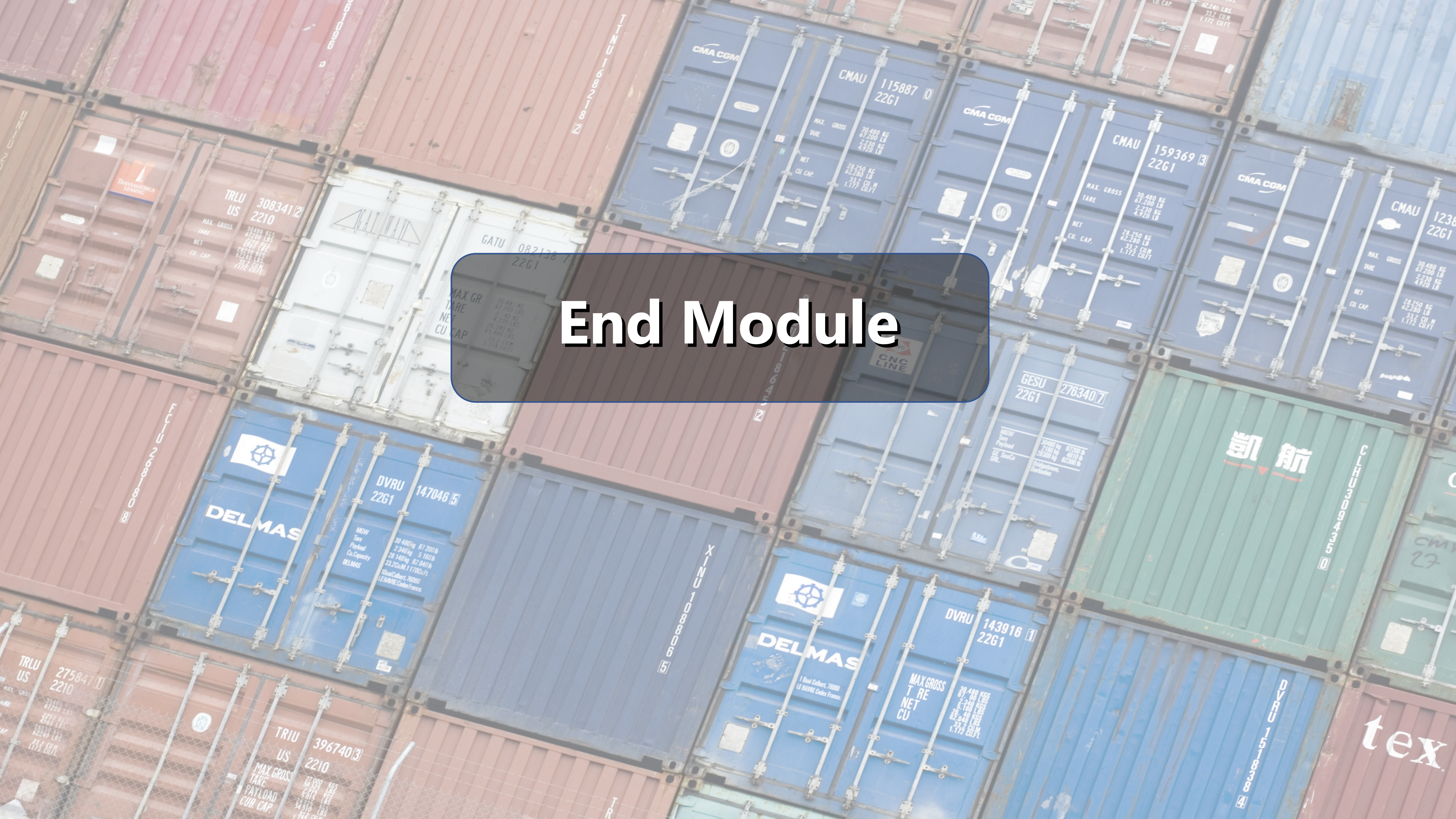
Rod Davison › CICD-Lab-1

You pushed to readme-update at Rod Davison / CICD-Lab-1 just now

Questions?

Class Project Discussion

End Module



CMA CGM
CMAU 115887 0
22G1
CMAU 159369 3
22G1
TRLU 308341 2
US 2210
GATU 082138 7
22G1
DVRU 147046 5
22G1
DELMAS
XINU 108806 5
DVRU 143916 1
22G1
GESU 276340 7
22G1
CLHU 309435 0
TRLU 275847 1
US 2210
TRIU 396740 3
US 2210
FCIU 268180 8
DVRU 151838 4
CNC LINE
凯 航
tex