M5 – Version Control (Git)

CS 136L F23 – LEC 7

Yiqing Irene Huang, Qianqiu Zhang

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

Disclaimer

- The following slides will not be presented page by page in class.
- They are my own study notes to share with students.
- In the lab session, we will cover key points, do small demos and give hints on commonly seen errors

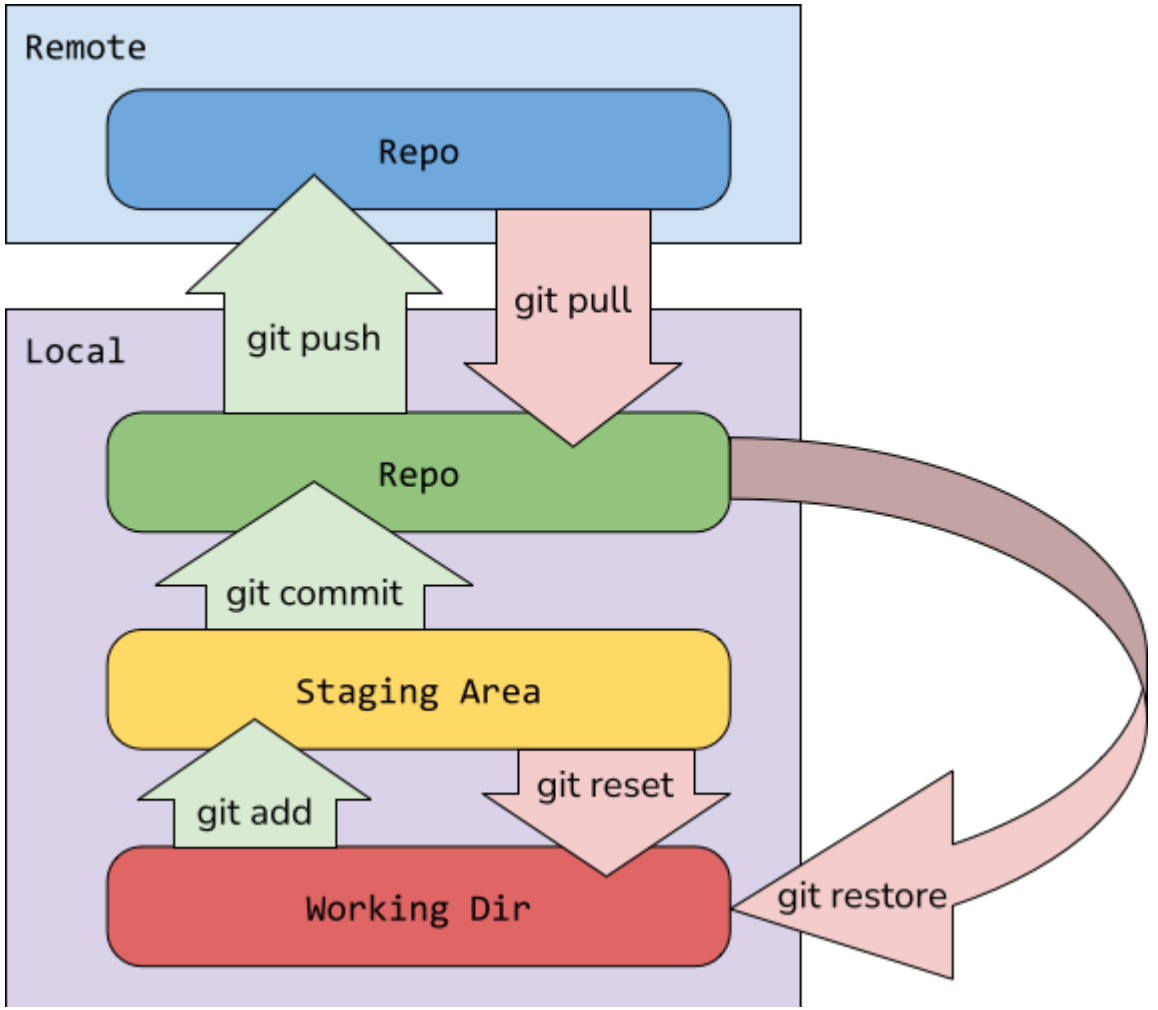
Main Points

Learn to use the git Version Control Software to keep track of changes made to files in projects

- Basic git commands,
  - Create repo, add files to be tracked
  - Monitor file status and track the change history of the repo
- Create, switch and merge branches
- Repository: clone , pull and push
- Merging branches
- Undo
- VS Code Git Extension

Introduction

- Why Version Control
  - Keep track of changes
  - Revert back
- Changeset
  - A group of files that represent a change to the system
- Repository (repo)
  - The location of the canonical/main version
- Working Directory
  - A copy of the repo where you make changes before saving them
- Staging Area
  - A logical collection of changes from the working directory



Git Help and Configuration

Git commands	Notes
git help git help add	Shows details about how to use Git's add command Shows a list of commonly used commands

Git configuration has three levels

- System Level: `/etc/gitconfig`
- Global Level: `~/.gitconfig`
- Local Level: `repodir/.git/config`

Git commands	
git config --list git config --global user.name "jsmith" git config --global user.email "jsmith@jsmith.com"	git config --global core.excludesfiles ~/.gitignore_global git config --global core.editor "vim" git config --global color.ui true

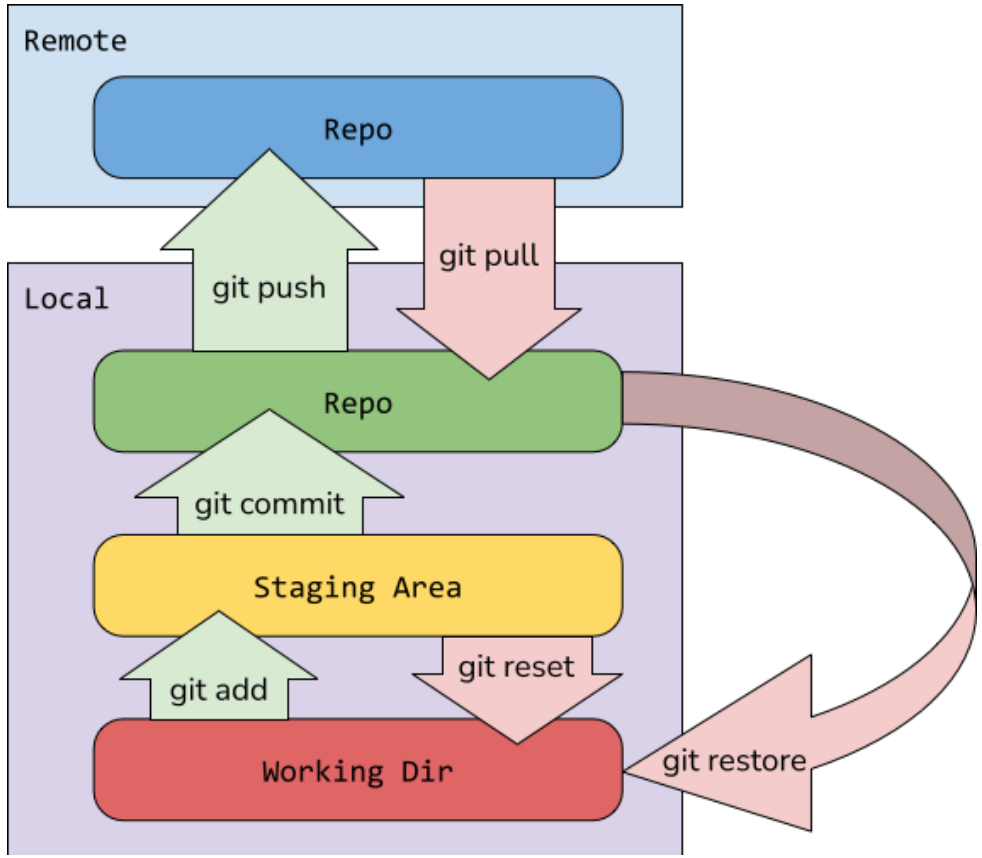
https://git-scm.com/docs/git

Git Basics

Commands	Example Commands
`git init`	`cd myrepo` `git init`
`git add <file|dir>`	`git add README` `git add .`
`git commit –m <msg>`	`git commit –m “commit msg”`

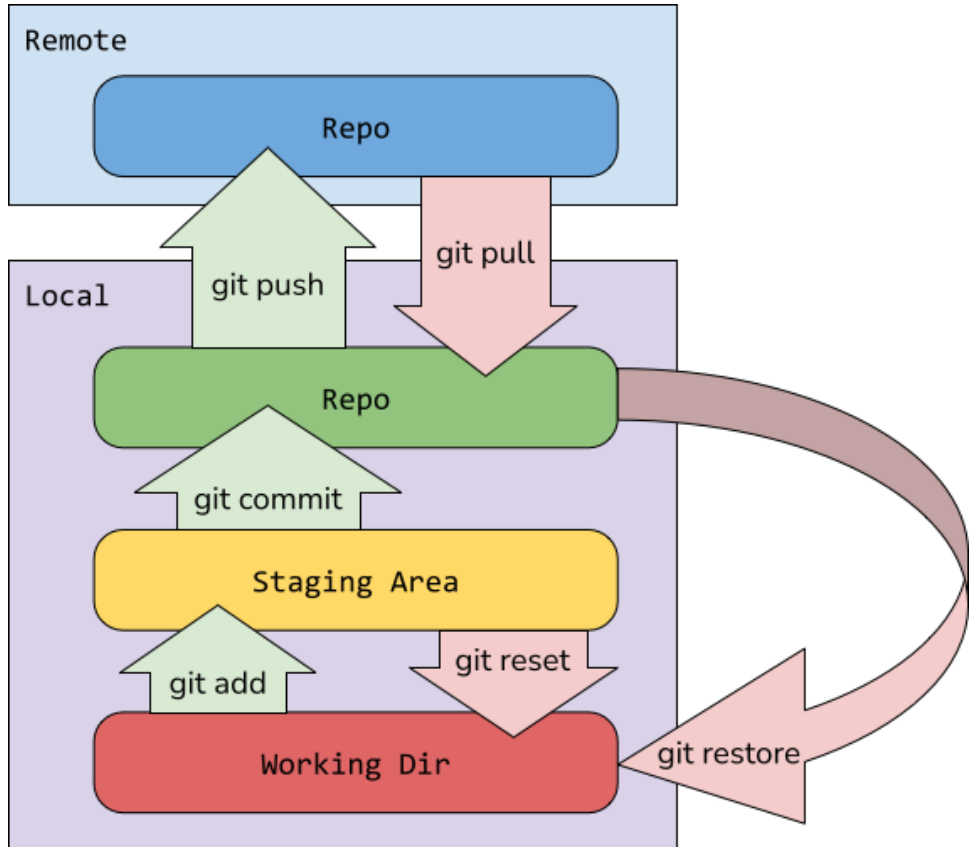
`git status [path]`	`git status`
	`git status .`
	`git status *`

`git diff`	`git diff --color-words hello.c` `git diff --staged hello.c`
`git mv <oldname> <newname>`	`git mv hello.c hello1.c`
`git rm`	`git rm to_delete.c`
`git clean`	`git clean -n`

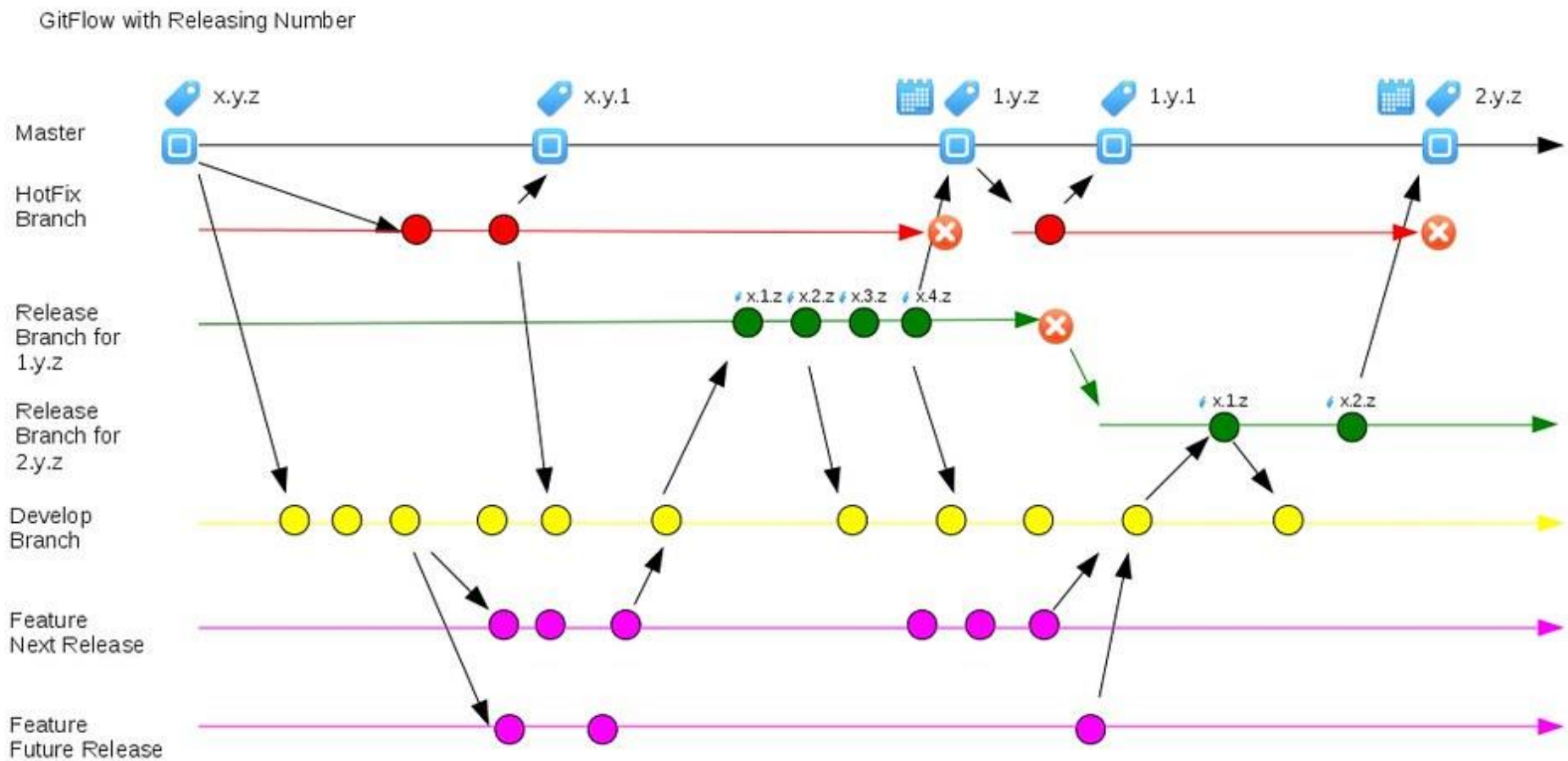


Git Remote Repo

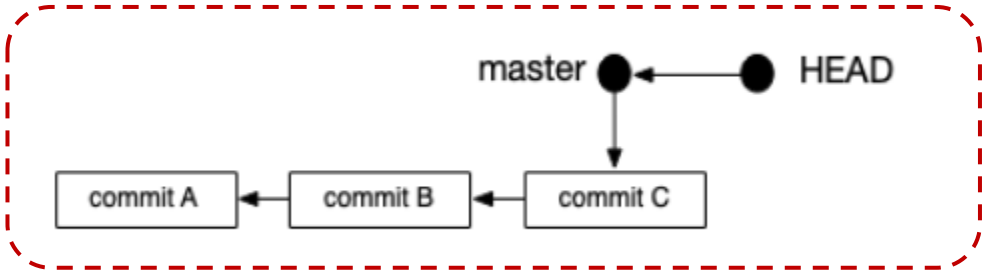
- Create a remote repo
  - git.uwaterloo.ca
  - github.com
  - bitbucket.org
  - on linux.student.cs.uwaterloo.ca
- Clone the repo
  - git clone <url>
  - ssh://<userid>@linux.student.cs.uwaterloo.ca/<repo_path>
- git push origin master
  - git push
- git pull
  - git fetch
  - git merge



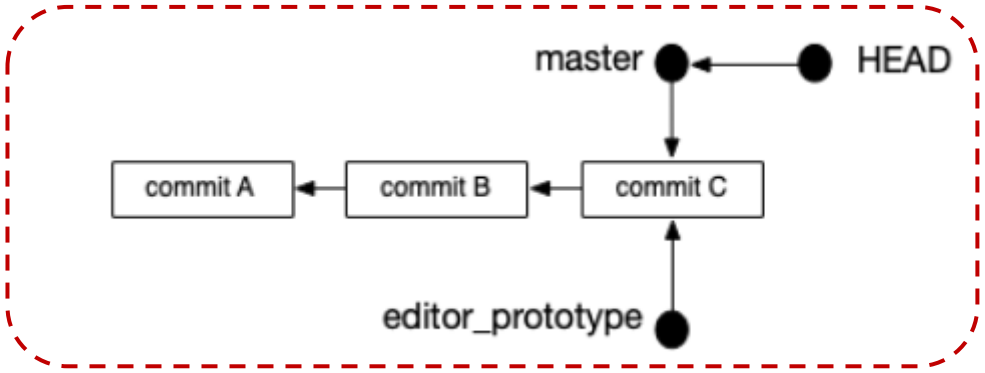
Git Branch


GitFlow with Releasing Number
x.y.z
x.y.1
1.y.z
1.y.1
2.y.z
Master
HotFix Branch
x.1.z
x.2.z
x.3.z
x.4.z
Release Branch for 1.y.z
x.1.z
x.2.z
Release Branch for 2.y.z
Develop Branch
Feature Next Release
Feature Future Release

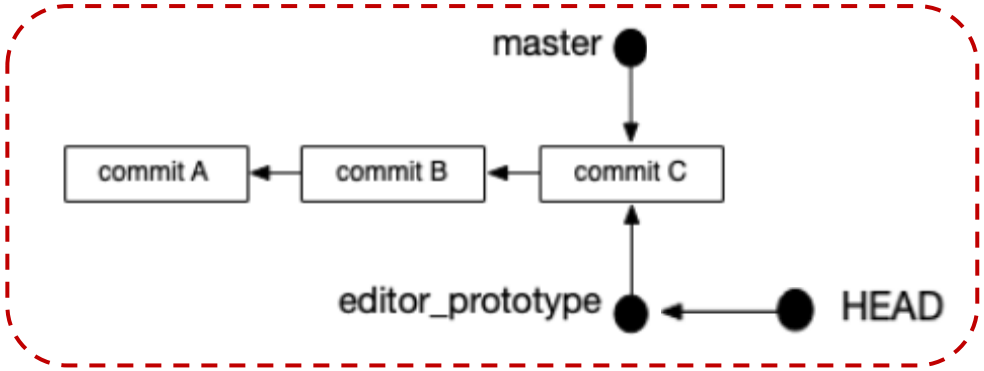
Git Branch




```
$git branch editor_prototype
$git branch
  editor_prototype
* master
```

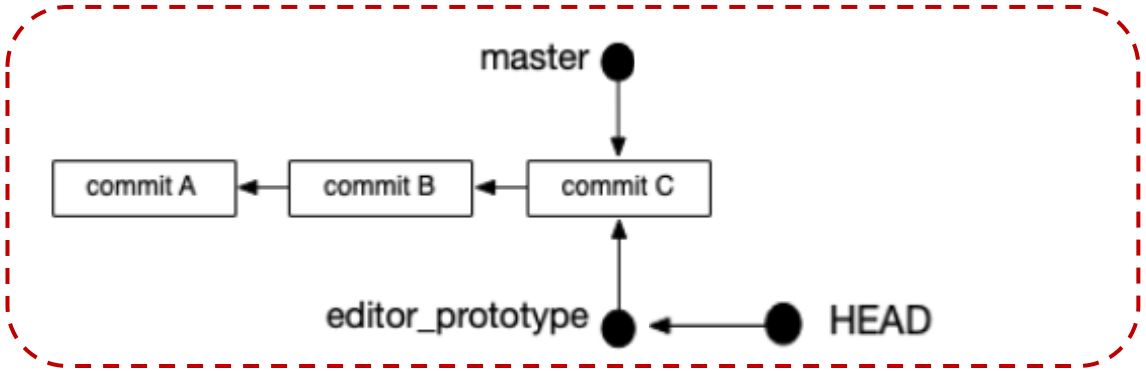


- HEAD points to the current active branch

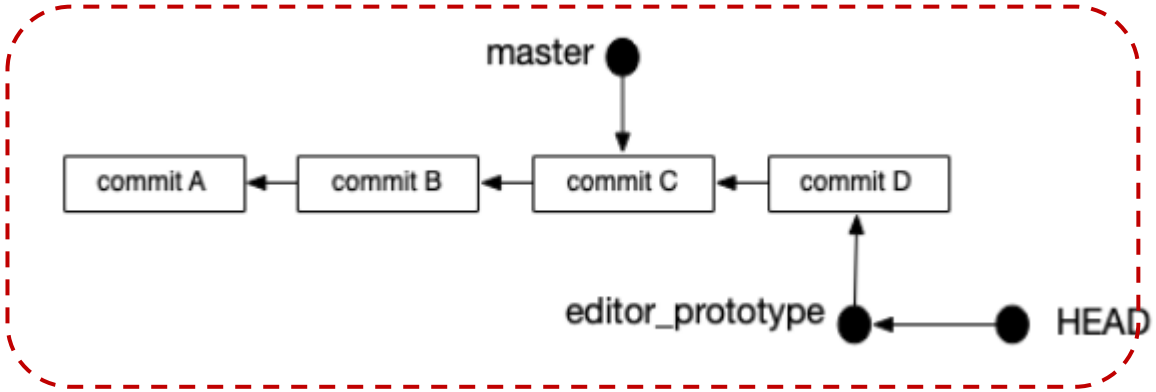



```
$git checkout editor_prototype
Switched to branch 'editor_prototype'
$git branch
* editor_prototype
  master
```

Git Branch Cont'd

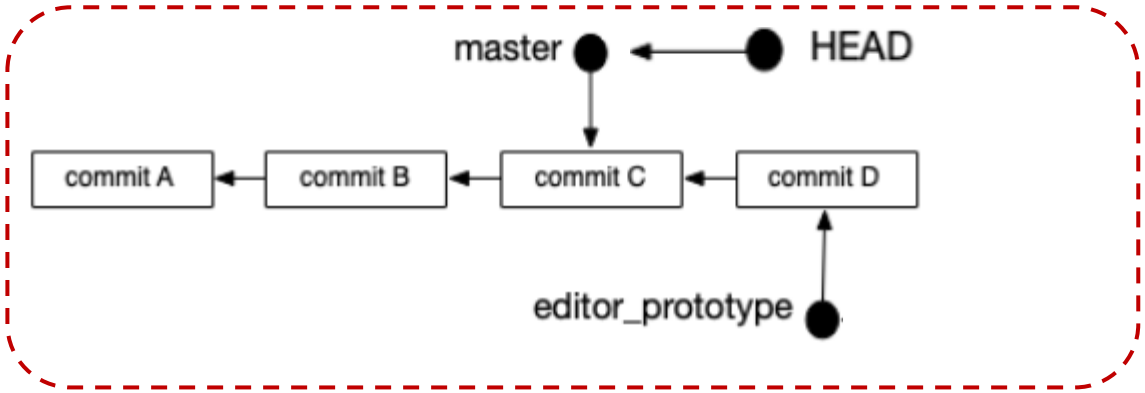



```
1 $git add editor.c
2 $git commit -m 'Added editor functionality'
```



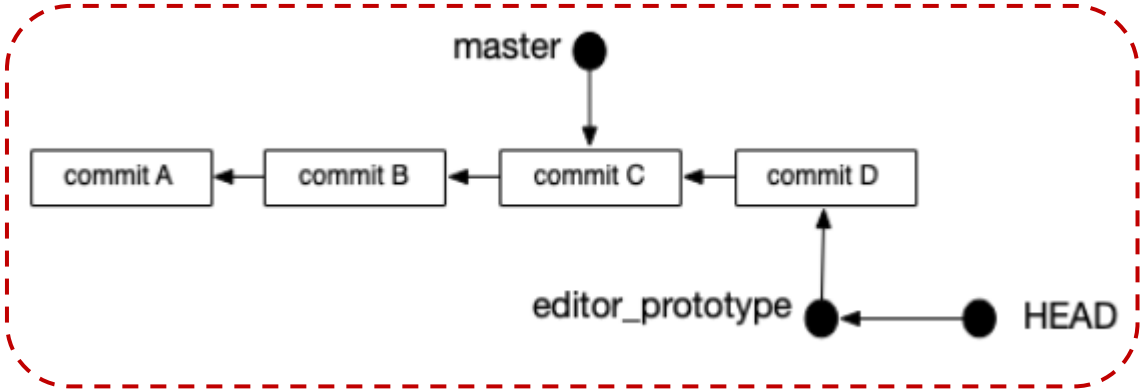
When Switching Branches

- Untracked files remains
- Files only committed in the other branch disappear
  - The editor.c disappears!

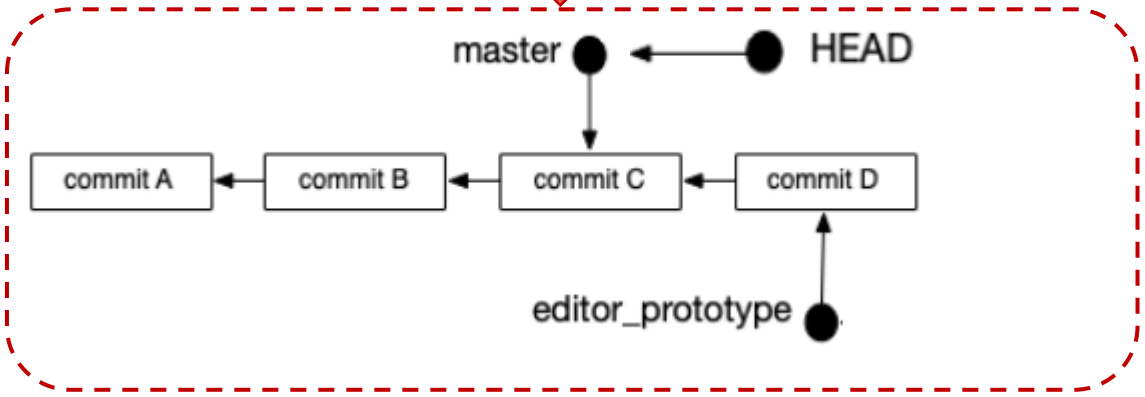



```
1 $git checkout master
2 Switched to branch `master'
```

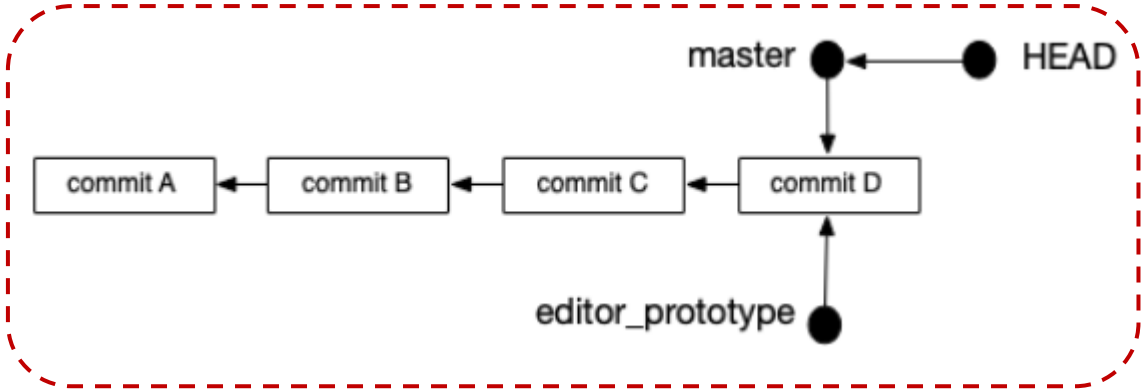
Git Branch Cont'd




```
1 $git checkout master
2 Switched to branch `master'
```



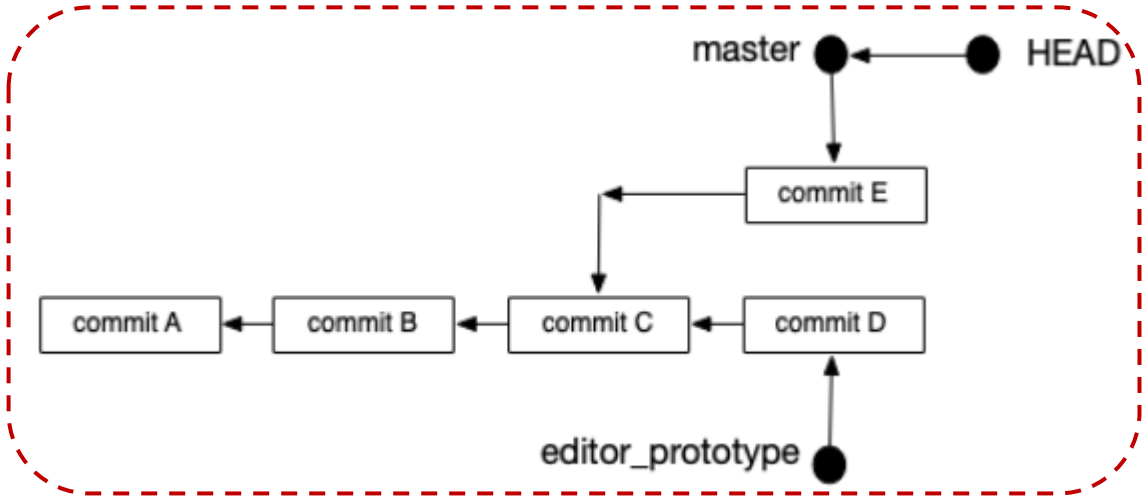
When merging has no conflicts




```
1 $git merge editor_prototype
2 Updating bfd09ac..707ddeb
3 Fast-forward
4  editor.c | 1 +
5  1 file changed, 1 insertion(+)
6  create mode 100644 editor.c
```

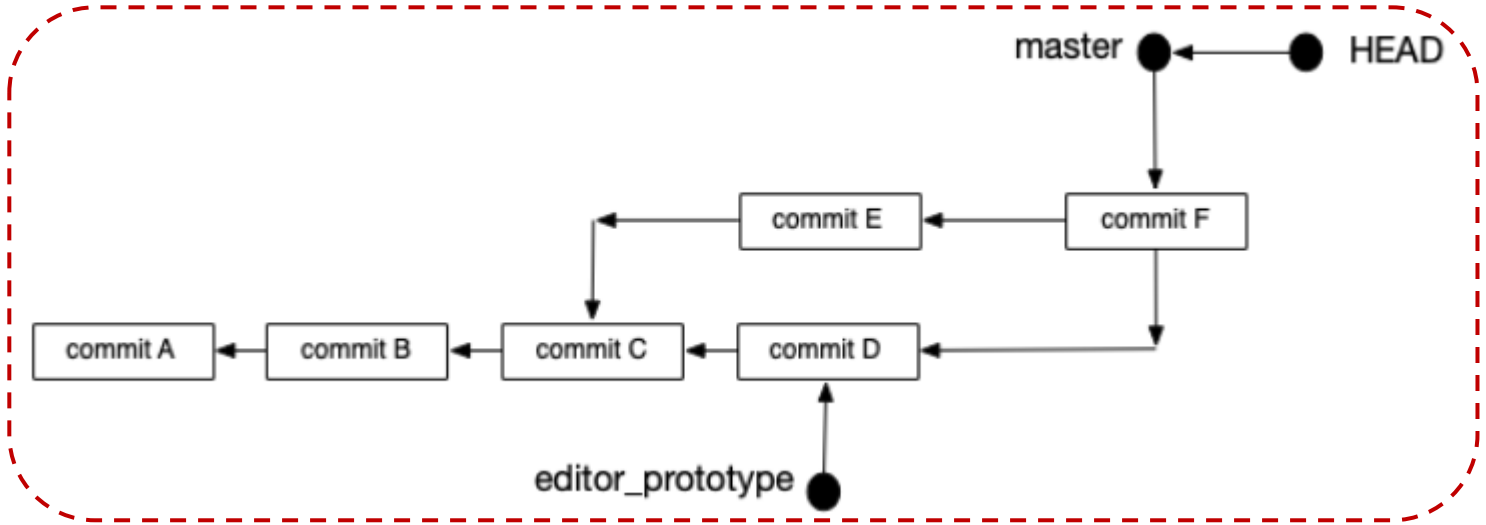
Git Merging

When there is no conflict



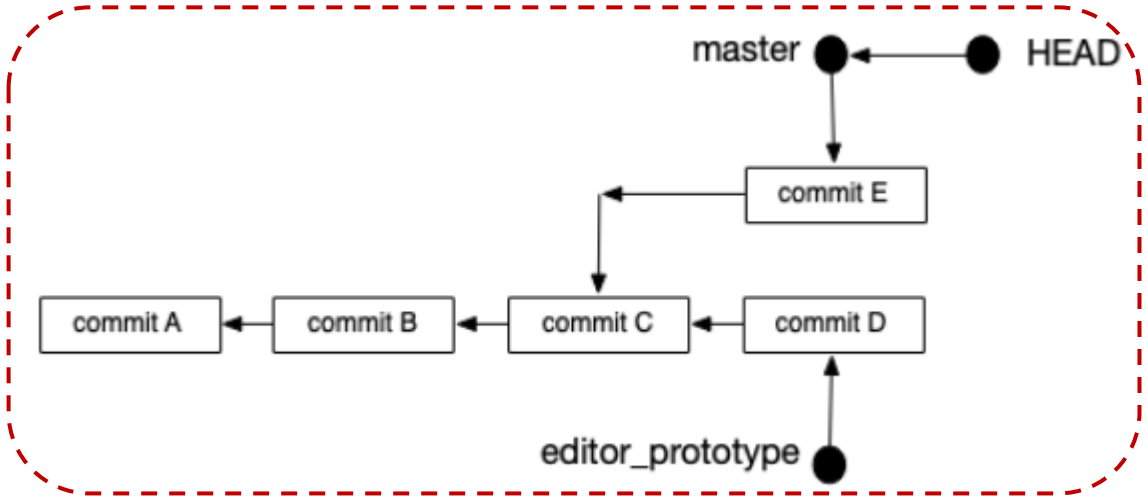

```
commit c61249ac47aa7d30922ab19b02beb942808b7812 (HEAD -> master)
Merge: d3755ca 59ee0c7
Author: Nomair Naeem <nanaeem@uwaterloo.ca>
Date:   Wed Jan 13 18:59:29 2021 -0500

    Merge branch 'editor_prototype'
```

```
$git merge editor_prototype
```



Git Merging

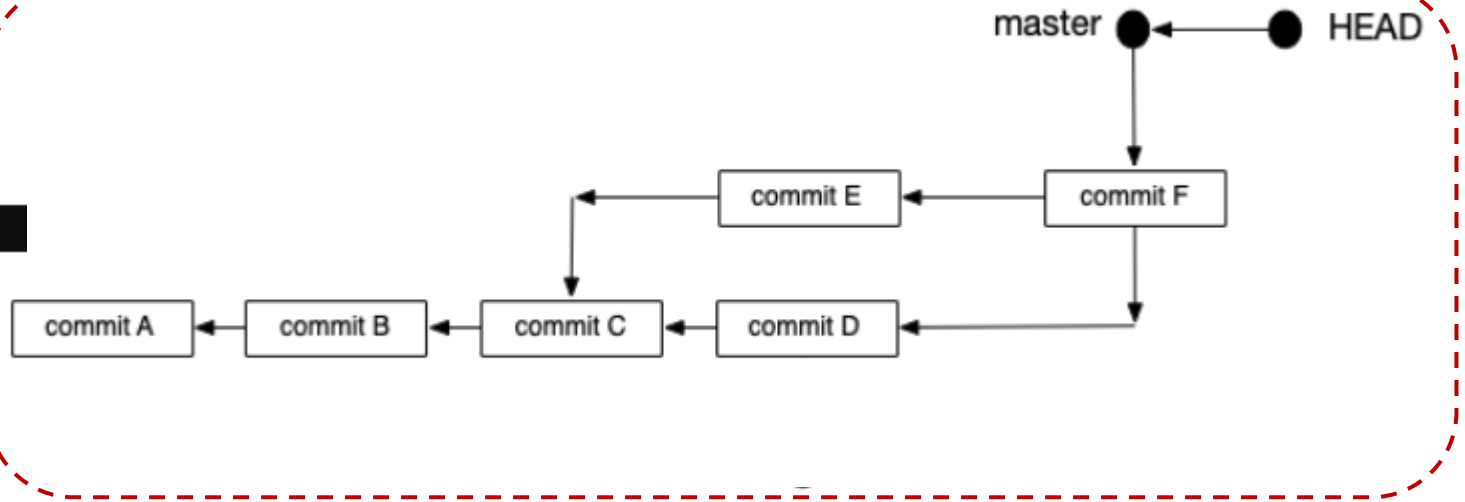
When there is no conflict




```
commit c61249ac47aa7d30922ab19b02beb942808b7812 (HEAD -> master)
Merge: d3755ca 59ee0c7
Author: Nomair Naeem <nanaeem@uwaterloo.ca>
Date:   Wed Jan 13 18:59:29 2021 -0500

    Merge branch 'editor_prototype'
```

```
$git merge editor_prototype
```

```
$git branch -d editor_prototype
```

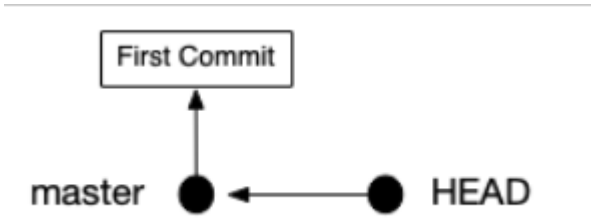


Git Merging

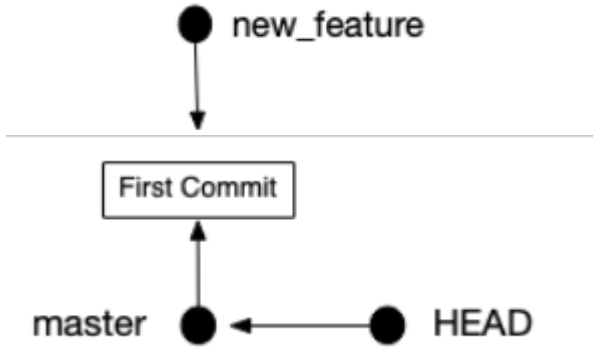
```c
#include <stdio.h>

int main(void){
  printf("Hello World");
}
```

Once this file with the contents has been created, run the following comma

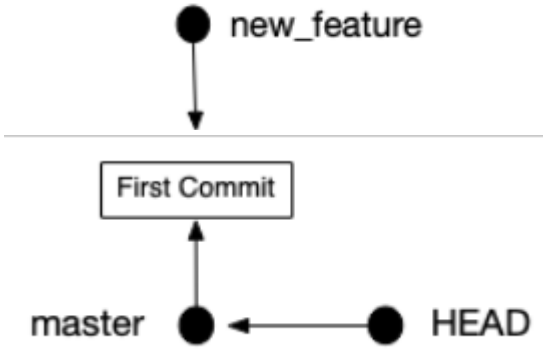
```
$git add hello.c
$git commit -m 'First commit'
[master (root-commit) 0eb559d] First Commit
 1 file changed, 5 insertions(+)
 create mode 100644 hello.c
```




```
$git branch new_feature
$git branch
* master
  new_feature
```



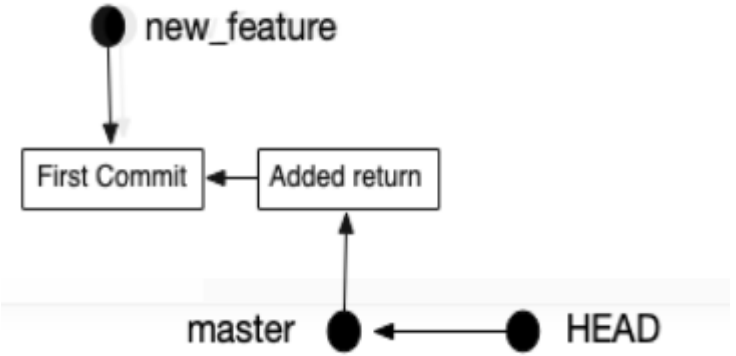
Git Merging



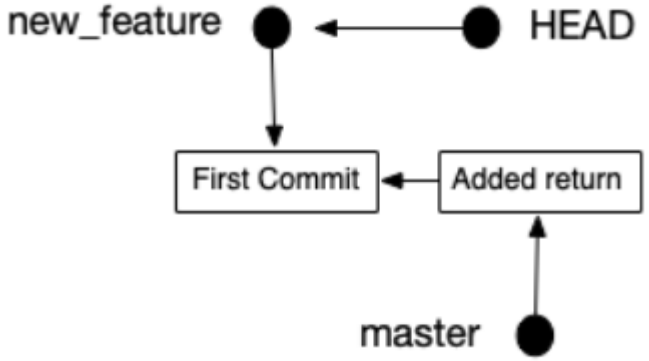

```
1  #include <stdio.h>
2
3  int main(void){
4    printf("Hello World");
5    return 0;
6  }
```

Then run the commands:

```
1  $git add hello.c
2  $git commit -m 'Added return'
3  [master 5e088da] Added return
4   1 file changed, 1 insertion(+)
```




```
1  $git checkout new_feature
2  Switched to branch 'new_feature'
3  $cat hello.c
4  #include <stdio.h>
5
6  int main(void){
7    printf("Hello World");
8  }
```



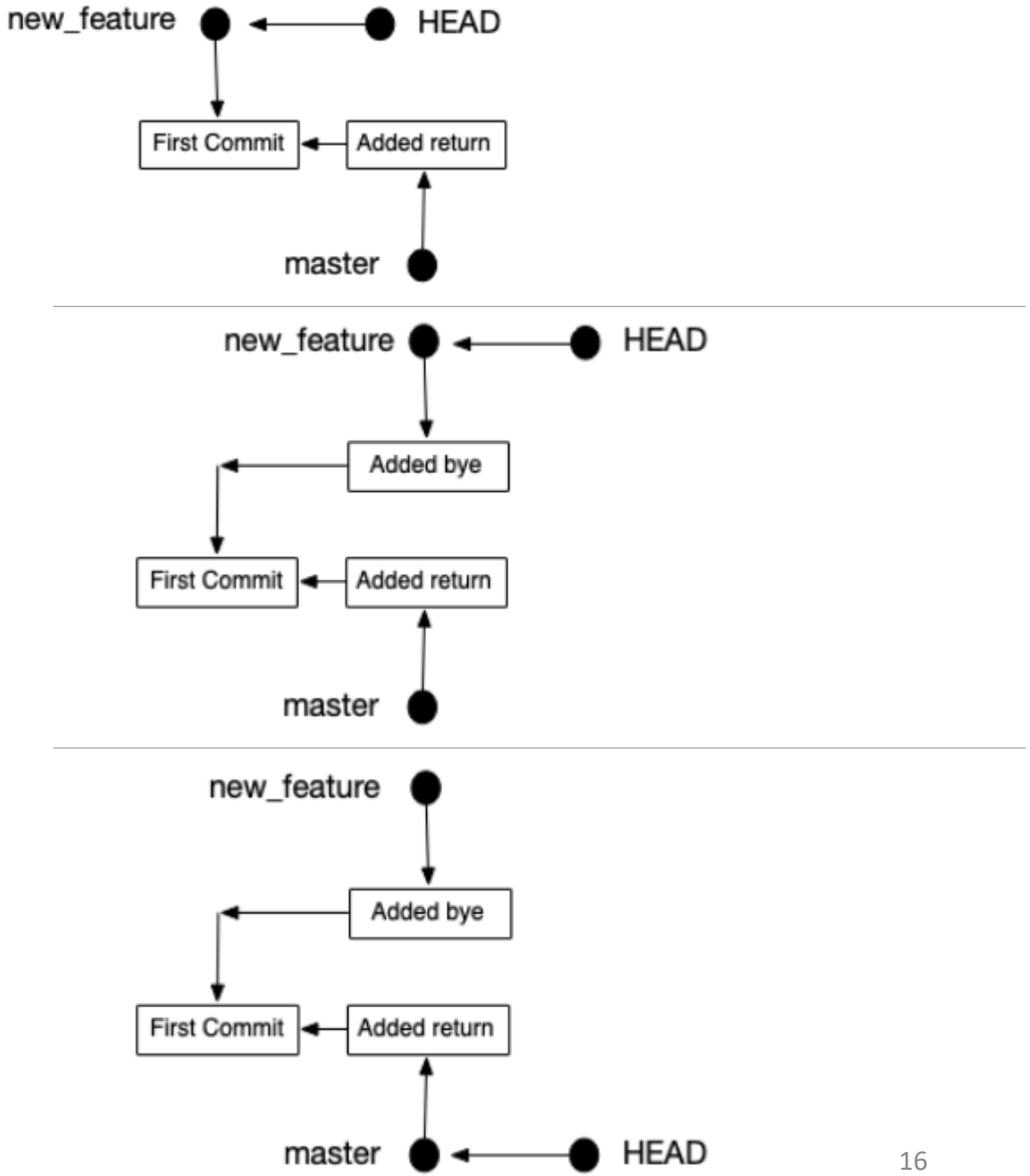
Git Merging

```c
#include <stdio.h>

int main(void){
  printf("Hello World");
  printf("Bye World");
}
```

And then, let's commit this change to the `new_feature` branch:

```
$git add hello.c
$git commit -m 'Added feature to say bye'
[new_feature 3e76a98] Added feature to say bye
 1 file changed, 1 insertion(+)
```

```c
$git checkout master
$cat hello.c
#include <stdio.h>

int main(void){
  printf("Hello World");
  return 0;
}
```



Git Merging - Conflicts

```
$git merge new_feature
Auto-merging hello.c
CONFLICT (content): Merge conflict in hello.c
Automatic merge failed; fix conflicts and then commit the result.
```

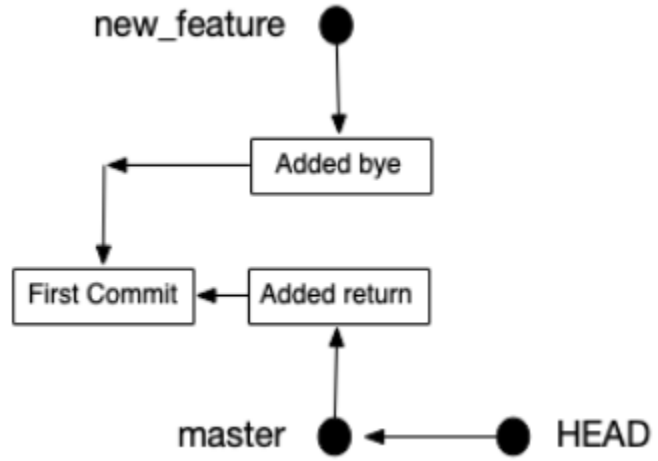
```
$git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   hello.c

no changes added to commit (use "git add" and/or "git commit -a")
```

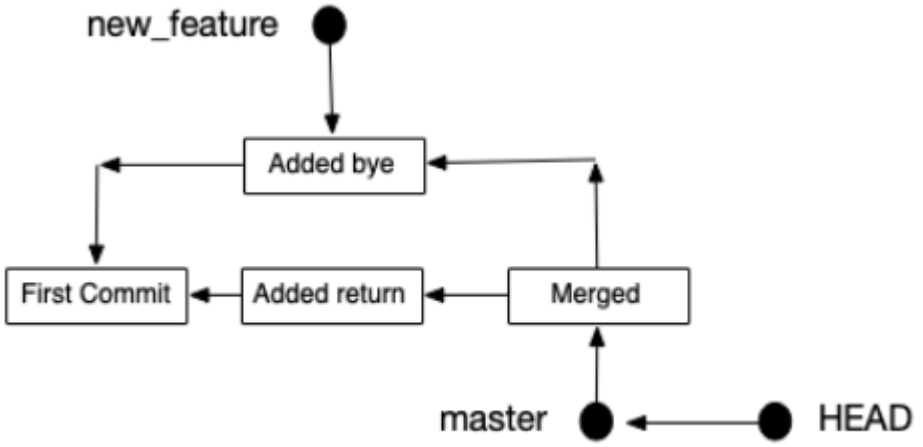
```
#include <stdio.h>

int main(void){
  printf("Hello World");
<<<<<<< HEAD
  return 0;
=======
  printf("Bye World");
>>>>>>> new_feature
}
```

Git Merging – Fixing Conflicts

```
$cat hello.c
#include <stdio.h>

int main(void){
  printf("Hello World");
  printf("Bye World");
  return 0;
}
$git add hello.c

$git status

On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
    modified:   hello.c
$git commit -m 'Merged new_feature'
[master 644fa94] Merged new_feature
```

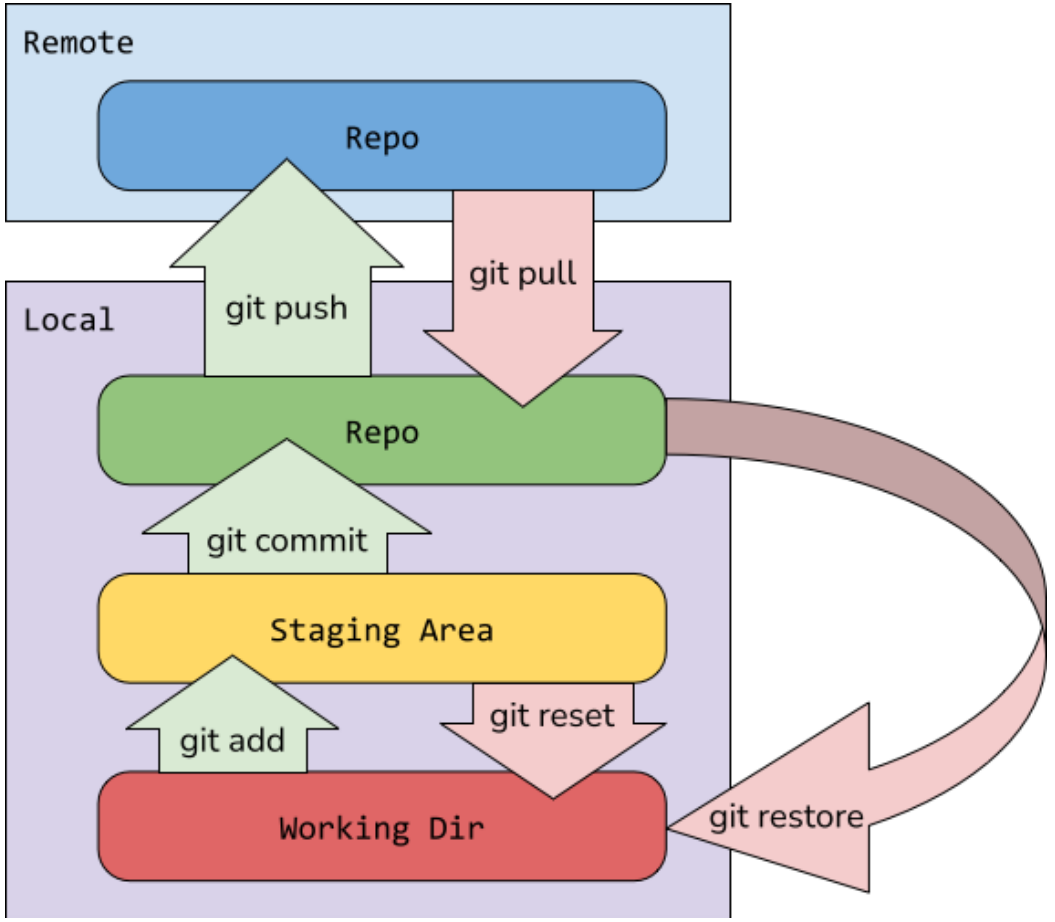
```
#include <stdio.h>

int main(void){
  printf("Hello World");
<<<<<<< HEAD
  return 0;
=======
  printf("Bye World");
>>>>>>> new_feature
}
```





Git: Undo

- git revert <commit>
  - Automatically commits the reverted commit
  - The `-n` or `-no-commit` option makes the reverted commit in staging area
- git reset
  - Modes: --hard, --mixed, --soft
  - Be extra careful about --hard
- git checkout
  - A branch
  - A commit
  - A file



VS Code – Git Extension

- https://code.visualstudio.com/docs/sourcecontrol/overview
- Follow the steps in Section 5.7

Git Commands Cheat Sheet for the Lab

git init	initialize a git repo
git status	see the status of files
git log [--oneline]	show commit logs
git add	stage files
git commit	commit staged files
git branch	list all branches
git branch <branch_name>	create a new branch
git checkout <branch_name>	switch to the specified branch
git merge <branch_name>	merge with the specified branch
git clone ssh://<userid>@<server>/<repo_path>	clone a repo from a server via ssh

Acknowledgement

- Slides by courtesy of Carmen Bruni and Anton Mosunov
- Demo notes from Nomair Naeem
- Demo lectures by Carmen Bruni, Dave Tompkins, and Nomair Naeem

References

- CS 136L edX notes at https://online.cs.uwaterloo.ca/