

Git

Disclaimer





- I am not a git guru
 - I just use git
 - I often ask for help

What is Git

- “free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency”
- Safe/manageable way of storing your files
 - Keeps a history
 - Enables you to view changes
- Encourages experimentation
- Encourages collaboration







Why use Git?

This PC > Local Disk (C:) > my_super_program >

Name	Date modified	Type
 17.07.2018	28/08/2018 11:25 ...	File folder
 20.07.2018	28/08/2018 11:25 ...	File folder
 27.07.2018	28/08/2018 11:25 ...	File folder
 28.07.2018	28/08/2018 11:25 ...	File folder


Why use Git?

is PC > Local Disk (C:) > my_super_program >

Name	Date modified	Type
 17.07.2018	28/08/2018 11:25 ...	File folder
 20.07.2018	28/08/2018 11:25 ...	File folder
 27.07.2018	28/08/2018 11:25 ...	File folder
 28.07.2018	28/08/2018 11:25 ...	File folder
 Trial_add_x	28/08/2018 11:29 ...	File folder
 Trial_add_y	28/08/2018 11:31 ...	File folder

Why use Git?

PC > Local Disk (C:) > my_super_program > Trial_add_x >








Name	^	Date modified	Type
 17.06.2018		28/08/2018 11:25 ...	File folder
 20.06.2018		28/08/2018 11:25 ...	File folder
 27.06.2018		28/08/2018 11:25 ...	File folder
 28.06.2018		28/08/2018 11:25 ...	File folder

is PC > Local Disk (C:) > my_super_program > Trial_add_y >

Name	^	Date modified	Type
 15.08.2018		28/08/2018 11:25 ...	File folder
 20.08.2018		28/08/2018 11:25 ...	File folder

Why use Git?

PC > Local Disk (C:) > my_super_program >



Name	Date modified	Type
 17.07.2018	28/08/2018 11:25 ...	File folder
 20.07.2018	28/08/2018 11:25 ...	File folder
 27.07.2018	28/08/2018 11:25 ...	File folder
 28.07.2018	28/08/2018 11:25 ...	File folder
 Final	28/08/2018 11:30 ...	File folder
 Trial_add_x	28/08/2018 11:29 ...	File folder
 Trial_add_y	28/08/2018 11:32 ...	File folder

Why use Git?

his PC > Local Disk (C:) > my_super_program > Final >

Name	Date modified	Type
 20.08.2018	28/08/2018 11:25 ...	File folder
 22.07.2018	28/08/2018 11:25 ...	File folder
 Actual_final_version	28/08/2018 11:25 ...	File folder
 Try_add_z	28/08/2018 11:25 ...	File folder

Why use git?

 Send	To...	<u>my_best_friend</u> @gmail.com
	Cc...	
	Subject	My final version
Attached	 Actual_final_version.zip 176 bytes	

Why use git?

From: fake best friend
Sent: Monday, 27 August 2018 5:19 PM
To: David Boland <my_best_friend@gmail.com>
Subject: Re: my final version

Hi David,

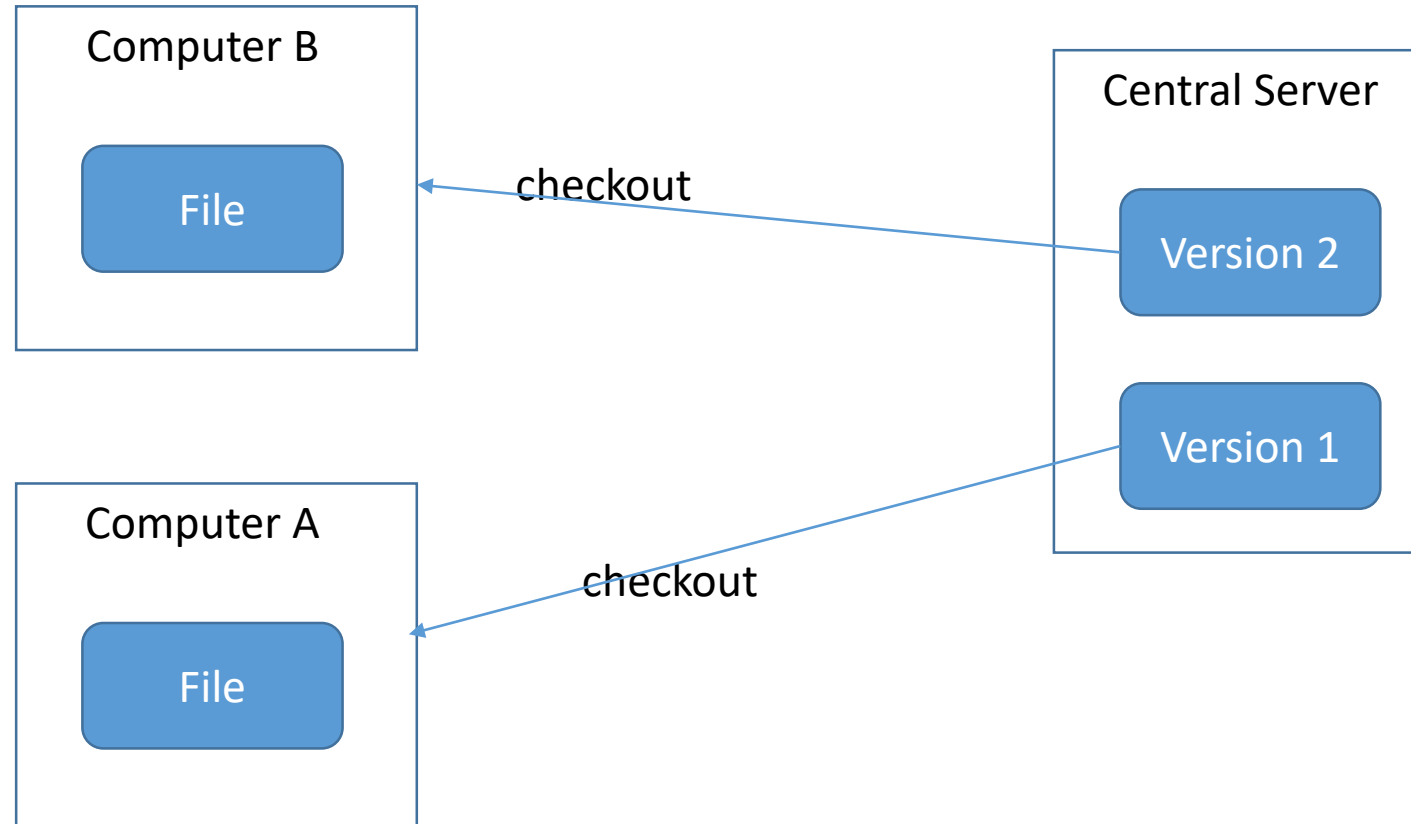
Um...when you were working on it, I also made some edits. Here's my version

From: David Boland

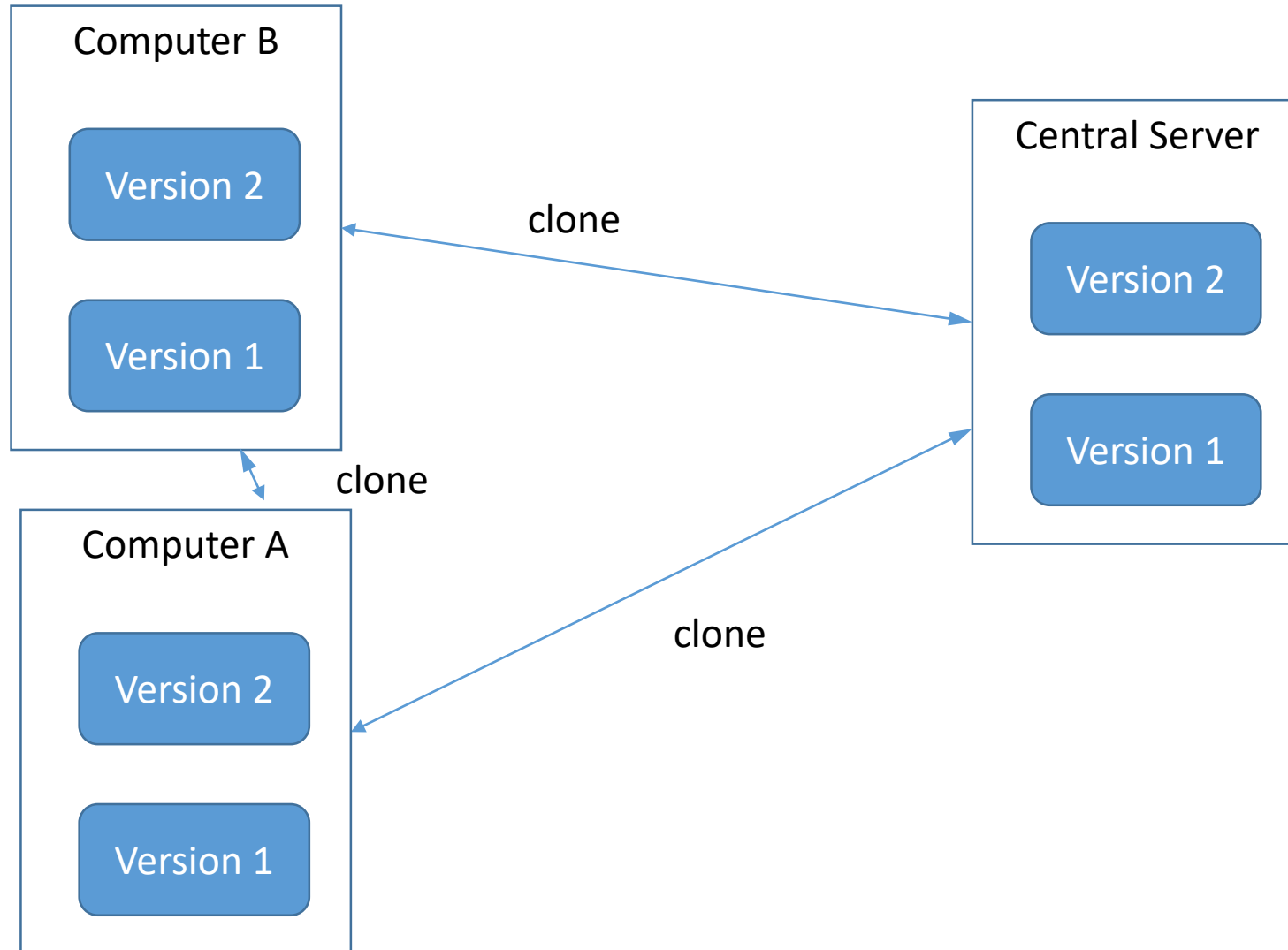
Contents

- How to use Git
- Some background into how Git manages data

How traditional version control system works



How git works



Git clone

- Everything is fast
 - Local storage
- Every clone is a backup
- Work offline
 - Perform diffs
 - View history
 - Commit changes
 - Merge branches
 - Look at old file versions
 - Work under different branches

How git works

- Git almost never removes data

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git init
Initialized empty Git repository in /home/vivado/elec5619/git_practice/.git/
vivado@vivado-VirtualBox:~/elec5619/git_practice$ ls
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim README.txt
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add README.txt
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit
Aborting [commit due to empty commit message.
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'initial commit'[master (root-com
mit) 0f74aea] initial commit
 2 files changed, 8 insertions(+)
 create mode 100644 README.txt
 create mode 100644 hello.c
```



```
#include<stdio.h>

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

vivado-VirtualBox: ~/elec5619/git_practice

```
this is my project
```

How are git files stored?

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git rev-list --objects --all
0f74aeafd6925af575df0075afbcadbcbada5df82
834eae4f14ce7120003c82b7ec22bc6044a8f57c
134dfe300d9c9d528864c37b8eb814d332db3cc4 README.txt
1af7f8f585f649b9e1b28f0a3636606ccae1400c hello.c
```

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'better hello message'
[master 9633740] better hello message
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
vivado-VirtualBox: ~/elec5619/git_practice
#include<stdio.h>

int main(void) {
    printf("Hello man\n");
    return 0;
}
```

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git rev-list --objects --all
963374065dc1dc09be46cfbbc8a119a17e384837
0f74aeafd6925af575df0075afbcadbcbada5df82
8b927dd441d02168c58fb22784581c7811c5f13e
134dfe300d9c9d528864c37b8eb814d332db3cc4 README.txt
499c80f3e1a0711ae76570b1ab5b5af23cad2abf hello.c
834eae4f14ce7120003c82b7ec22bc6044a8f57c
1af7f8f585f649b9e1b28f0a3636606ccae1400c hello.c
```

Commit A

Commit B

134 Readme.txt
1af Hello.c

134 Readme.txt
499 Hello.c

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git rev-list --objects --all
963374065dc1dc09be46cfbbbc8a119a17e384837
0f74aeafd6925af575df0075afbcadbcbada5df82
8b927dd441d02168c58fb22784581c7811c5f13e
134dfe300d9c9d528864c37b8eb814d332db3cc4 README.txt
499c80f3e1a0711ae76570b1ab5b5af23cad2abf hello.c
834eae4f14ce7120003c82b7ec22bc6044a8f57c
1af7f8f585f649b9e1b28f0a3636606ccae1400c hello.c
```

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit
Aborting commit due to empty commit message.
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'change file name'
[master 93d60ae] change file name
1 file changed, 7 insertions(+)
create mode 100644 better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git rev-list --objects --all
93d60ae3ab97a37e9228ea58469106349dfe5edb
963374065dc1dc09be46cfbbc8a119a17e384837
0f74aeafd6925af575df0075afbcadbcbada5df82
fbdb94c598e18bb47efc26caca4e481a7cf2d44b
134dfe300d9c9d528864c37b8eb814d332db3cc4 README.txt
499c80f3e1a0711ae76570b1ab5b5af23cad2abf better_hello.c
8b927dd441d02168c58fb22784581c7811c5f13e
834eae4f14ce7120003c82b7ec22bc6044a8f57c
1af7f8f585f649b9e1b28f0a3636606ccae1400c hello.c
```

Commit A

Commit B

Commit C

134 Readme.txt
1af Hello.c

134 Readme.txt
499 Hello.c

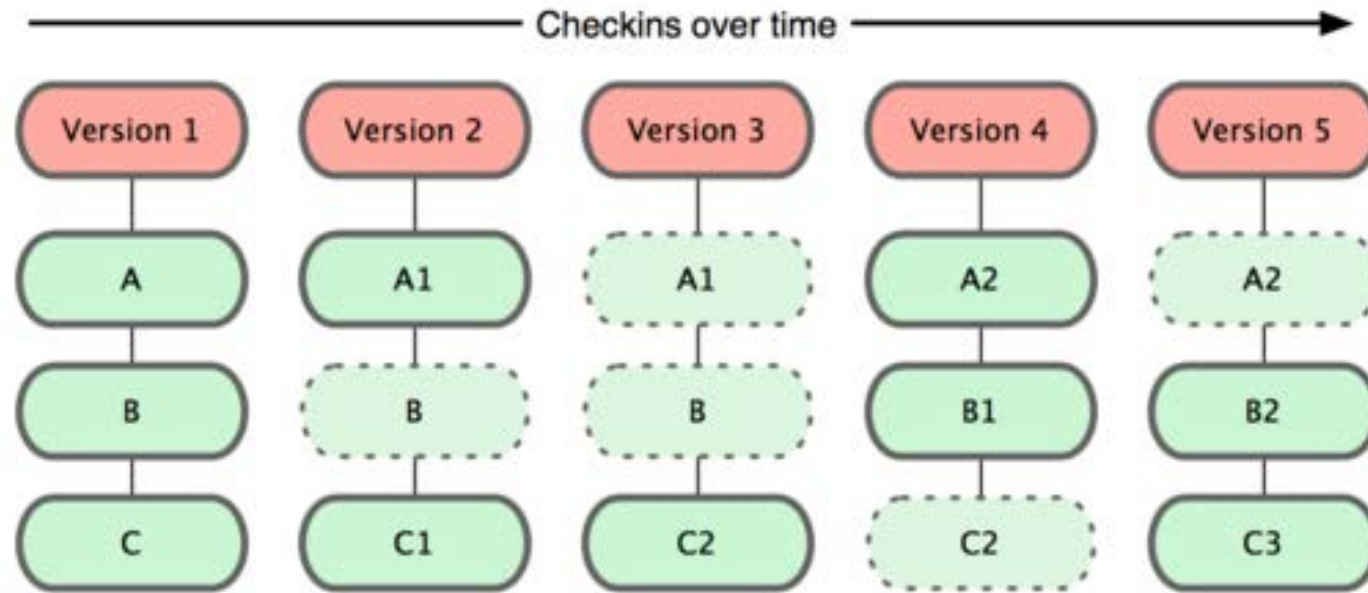
134 Readme.txt
499 better_hello.c

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit
Aborting commit due to empty commit message.
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'change file name'
[master 93d60ae] change file name
1 file changed, 7 insertions(+)
create mode 100644 better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git rev-list --objects --all
93d60ae3ab97a37e9228ea58469106349dfe5edb
963374065dc1dc09be46cfbbc8a119a17e384837
0f74aeafd6925af575df0075afbcadbcbada5df82
fbdb94c598e18bb47efc26caca4e481a7cf2d44b
134dfe300d9c9d528864c37b8eb814d332db3cc4 README.txt
499c80f3e1a0711ae76570b1ab5b5af23cad2abf better_hello.c
8b927dd441d02168c58fb22784581c7811c5f13e
834eae4f14ce7120003c82b7ec22bc6044a8f57c
1af7f8f585f649b9e1b28f0a3636606ccae1400c hello.c
```

How git works

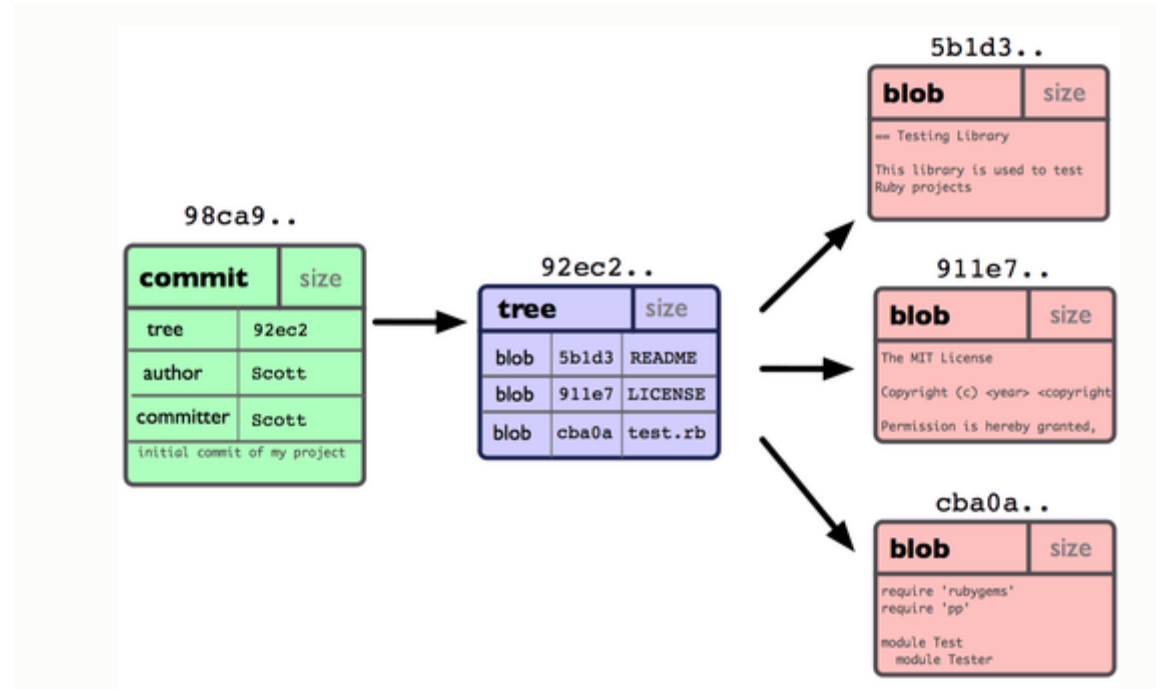
- Current version has pointers to all of the relevant files

How git works



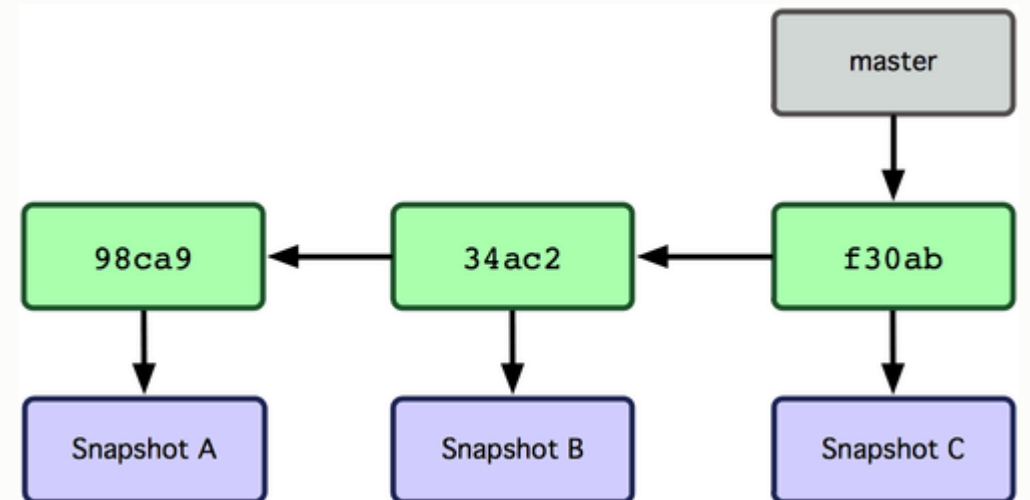
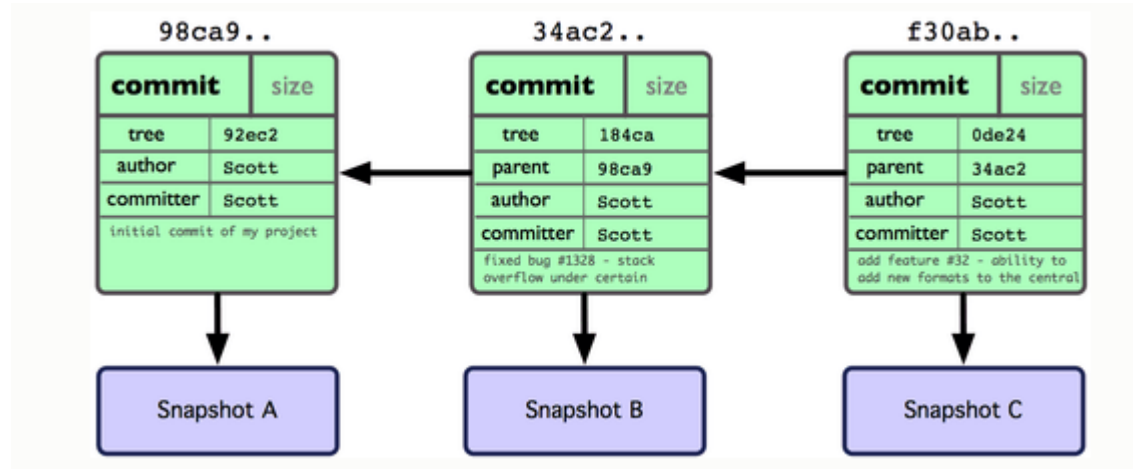
- *Internally, all copies of similar files are not stored, instead 'deltas', or changes between the files
 - This is only to save on storage space
 - It is easiest to think of different copies of similar files

What does a commit contain



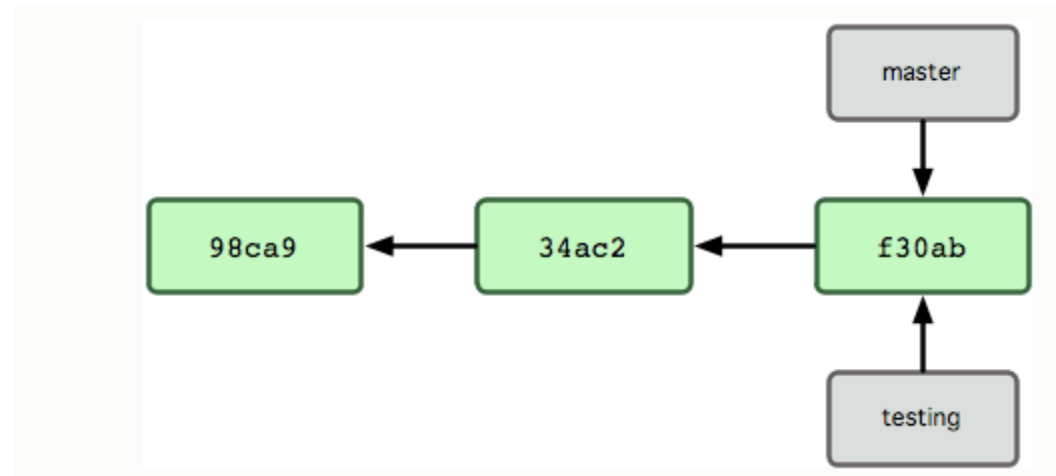
<https://git-scm.com/book/en/v1/Git-Branching-What-a-Branch-Is>

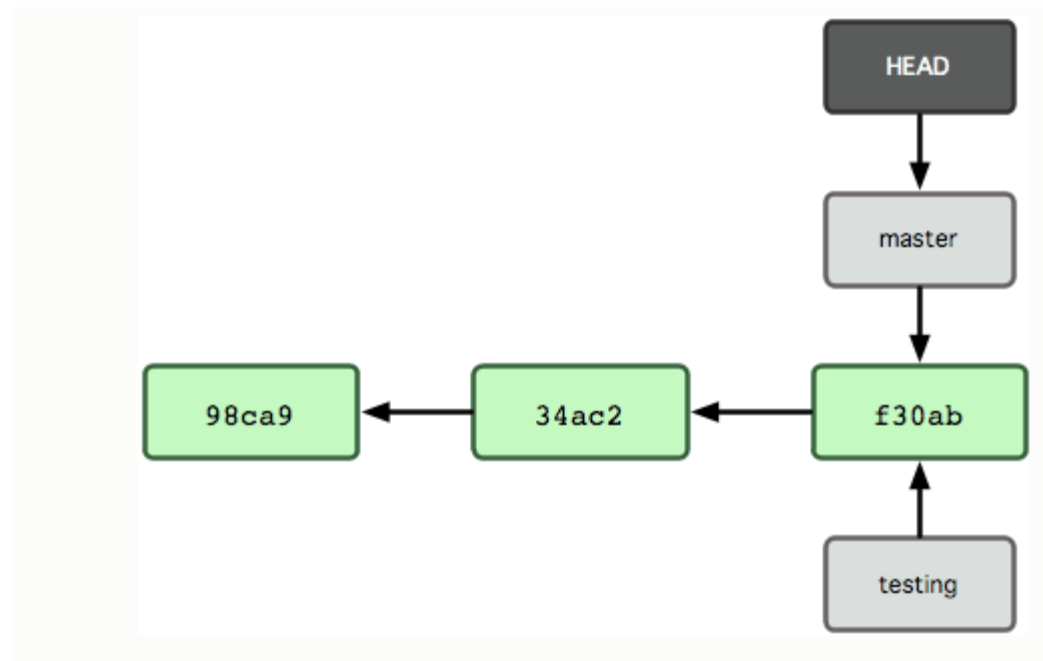
After a series of commits



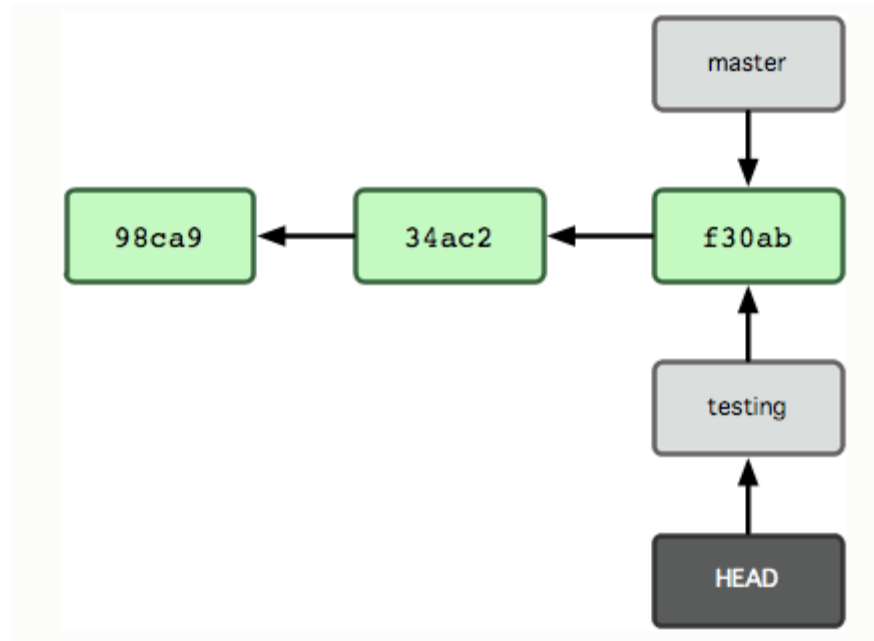
Git branching

- Type: git branch testing

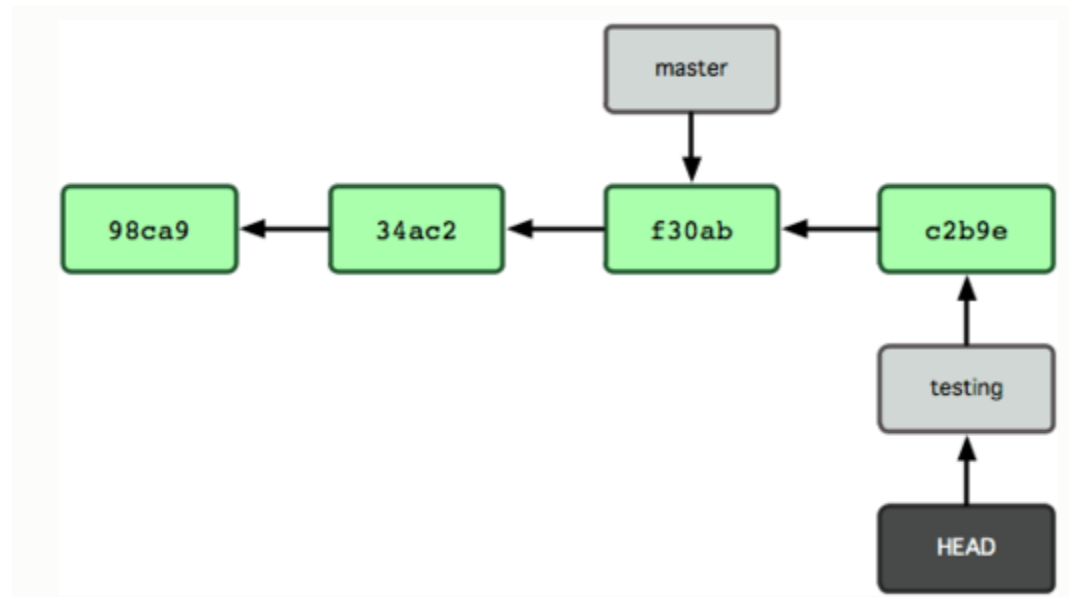




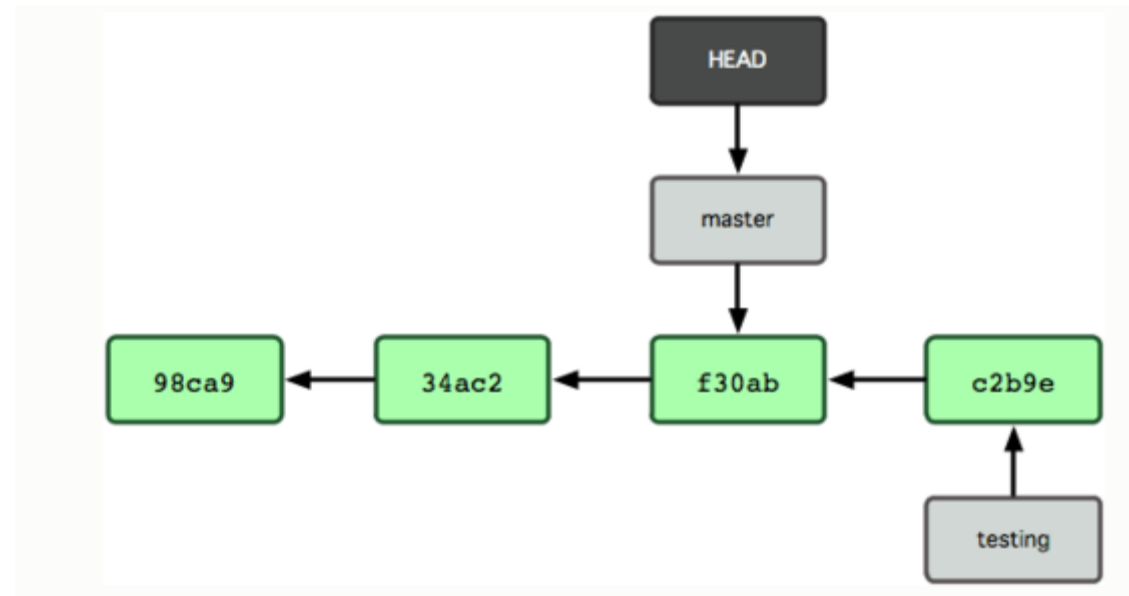
- Type git checkout testing



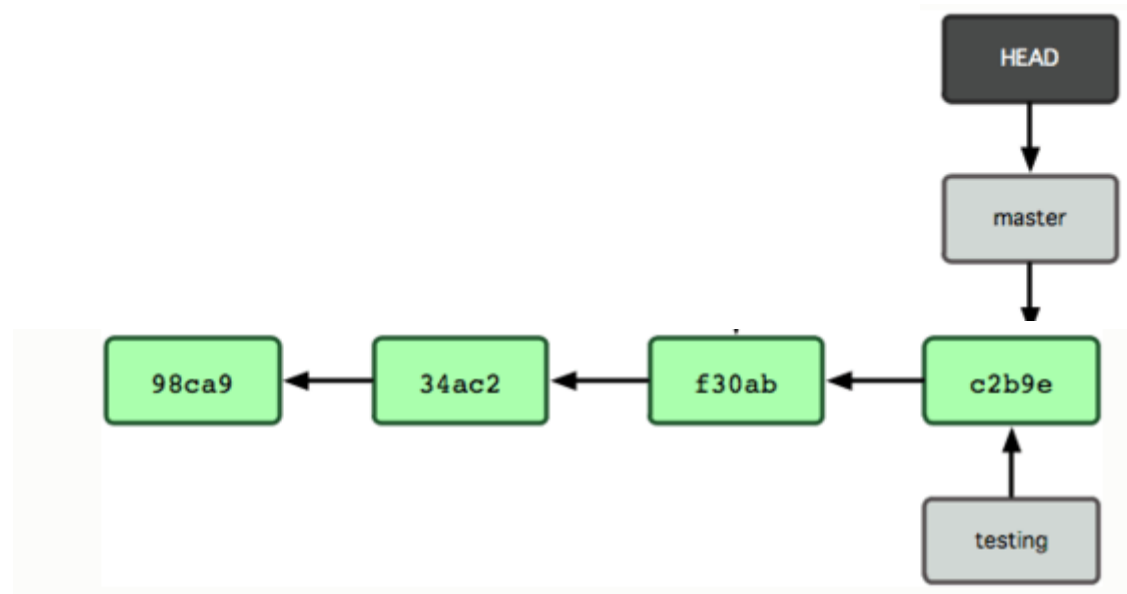
- vim hello.c
- git commit -a -m 'made a change'



- git checkout master

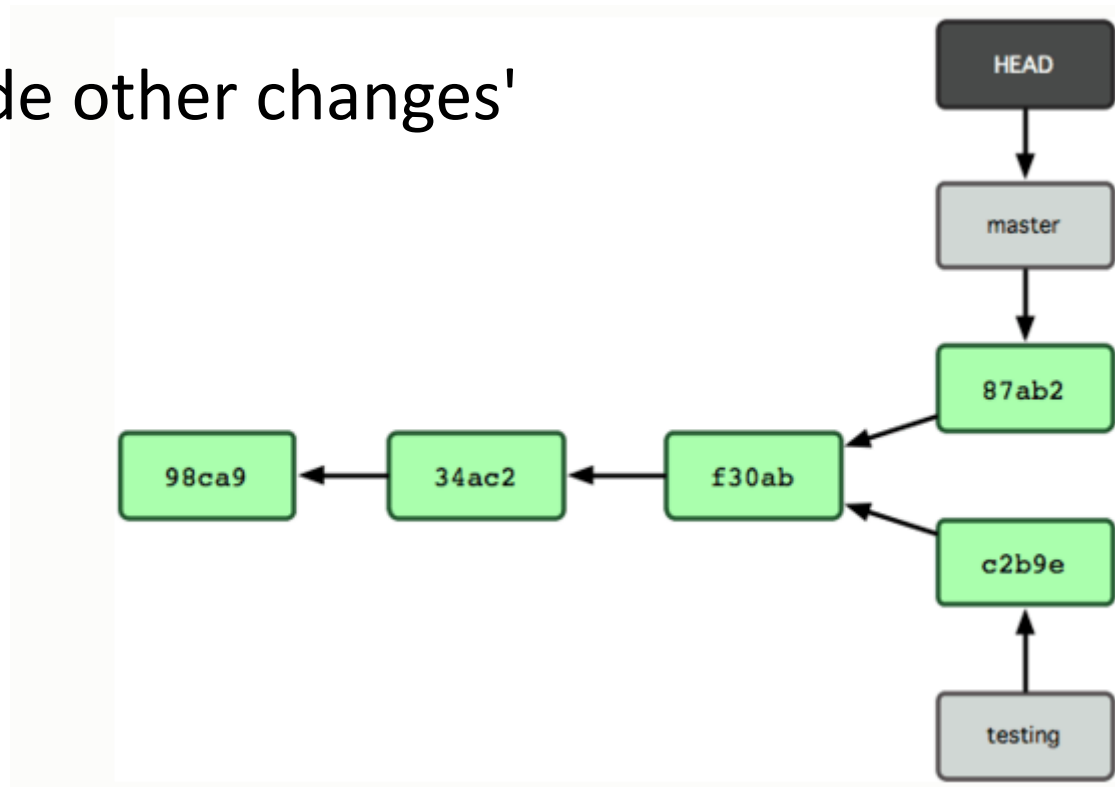


- git merge testing

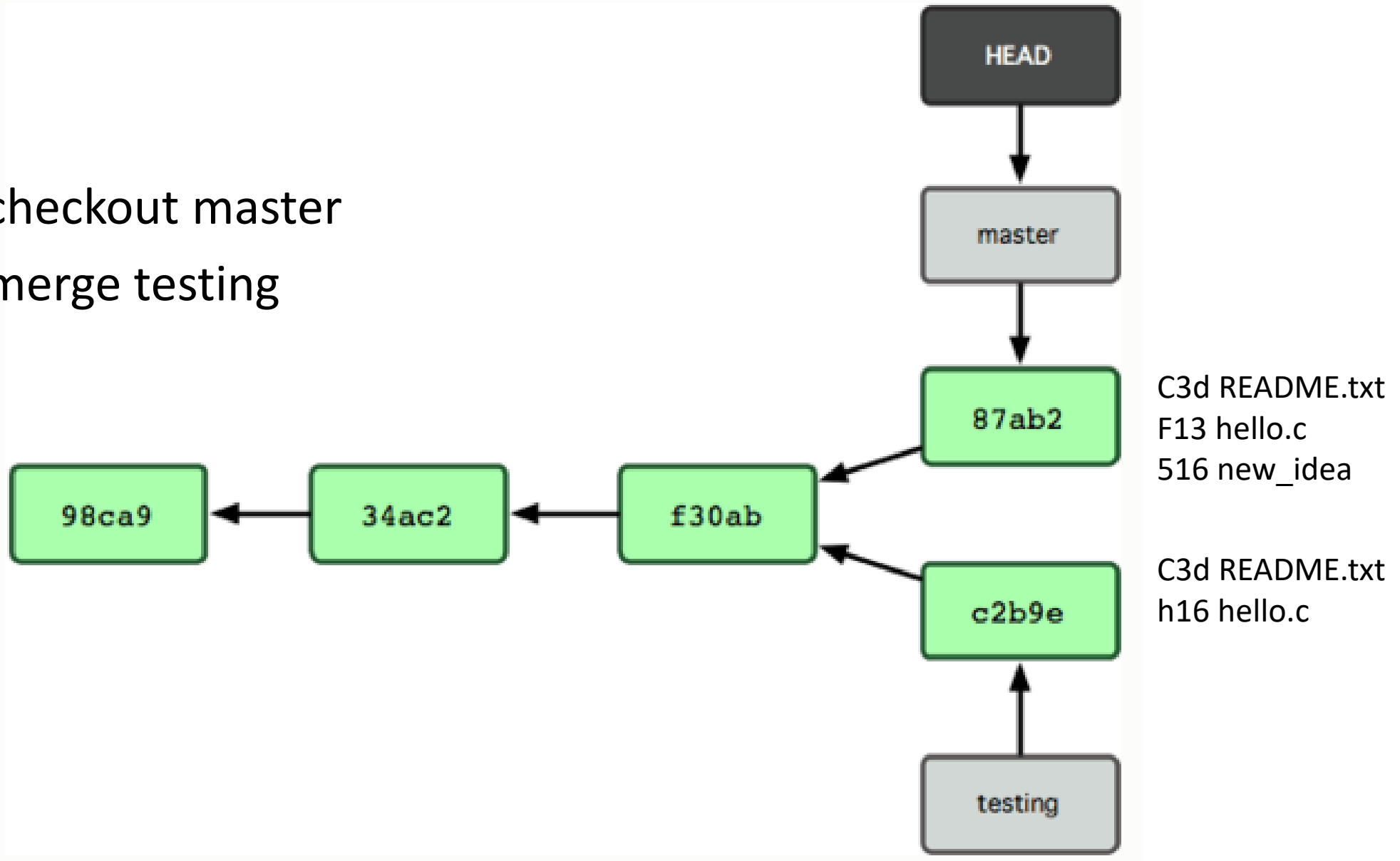


Complex merging

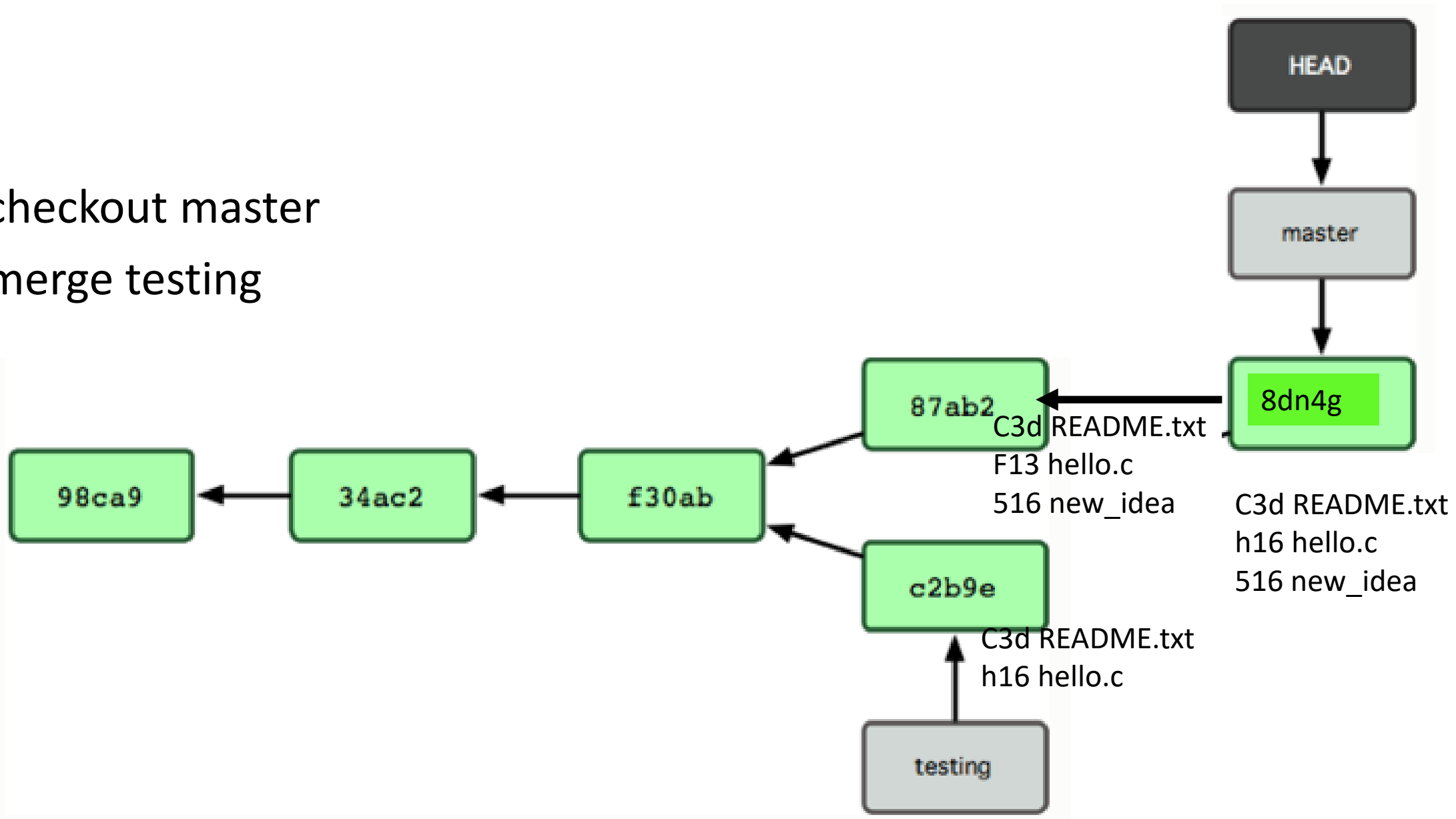
- Vim new_idea.c
- git commit -a -m 'made other changes'



- git checkout master
- git merge testing



- git checkout master
- git merge testing



Switching branches

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout modify
Switched to branch 'modify'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'changed message'
[modify 60fdd4e] changed message
 1 file changed, 1 insertion(+), 1 deletion(-)
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout master
Switched to branch 'master'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'improved message'
[master a0fc2b7] improved message
 1 file changed, 1 insertion(+), 1 deletion(-)
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout modify
Switched to branch 'modify'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout master
Switched to branch 'master'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
```

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout modify
Switched to branch 'modify'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'changed message'
[modify 60fdd4e] changed message
1 file changed, 1 insertion(+), 1 deletion(-)
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout master
Switched to branch 'master'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'improved message'
[master a0fc2b7] improved message
1 file changed, 1 insertion(+), 1 deletion(-)
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout modify
Switched to branch 'modify'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git checkout master
Switched to branch 'master'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
```

```
#include<stdio.h>

int main(void) {
    printf("Hello man!!!\n");
    return 0;
}
```

```
vivado-VirtualBox: ~/elec5619/git_practice
#include<stdio.h>

int main(void) {
    printf("Hello Everybody\n");
    return 0;
}
```

Merge conflicts

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git merge modify
Auto-merging better_hello.c
CONFLICT (content): Merge conflict in better_hello.c
Automatic merge failed; fix conflicts and then commit the result.
```

```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

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

        both modified:   better_hello.c

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        better_hello_BACKUP_10952.c
        better_hello_BASE_10952.c
        better_hello_LOCAL_10952.c
        better_hello_REMOTE_10952.c

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



```
vivado@vivado-VirtualBox:~/elec5619/git_practice$ vim better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git add better_hello.c
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit
[master 8fa5f4b] Merge branch 'modify'
vivado@vivado-VirtualBox:~/elec5619/git_practice$ git commit -m 'merged messages'
```

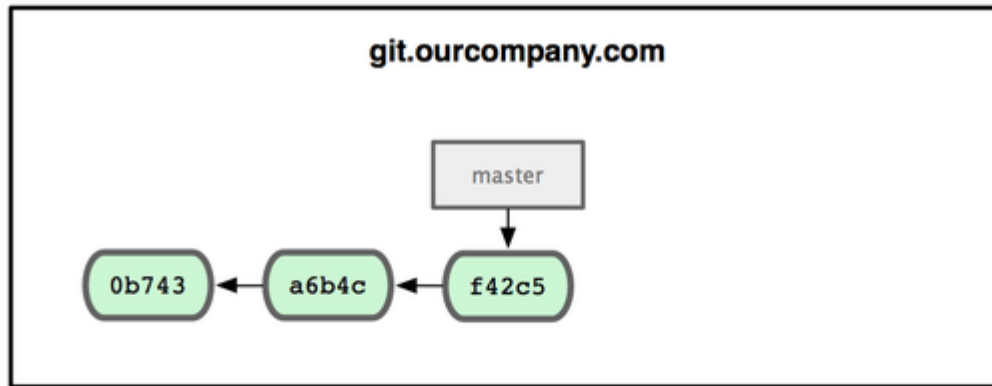
```
#include<stdio.h>

int main(void) {
<<<<<<< HEAD
    printf("Hello Everybody\n");
=====
    printf("Hello man!!!\n");
>>>>>>> modify
    return 0;
}
```

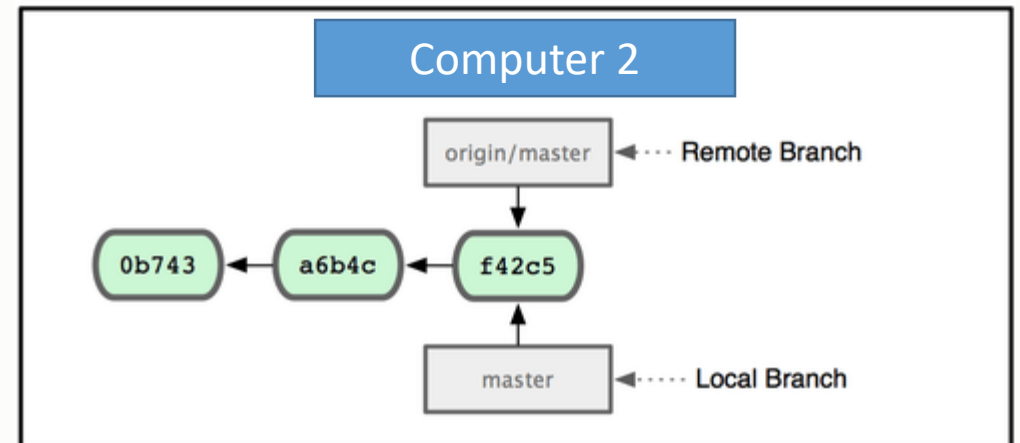
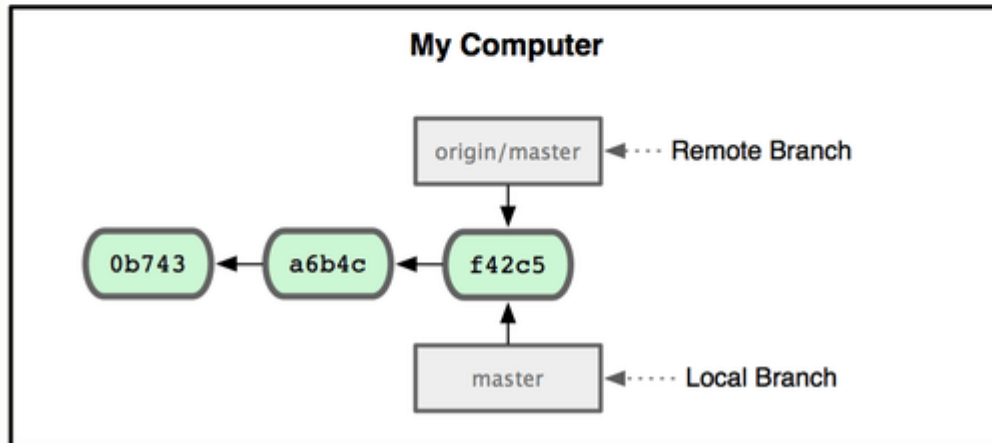
```
#include<stdio.h>

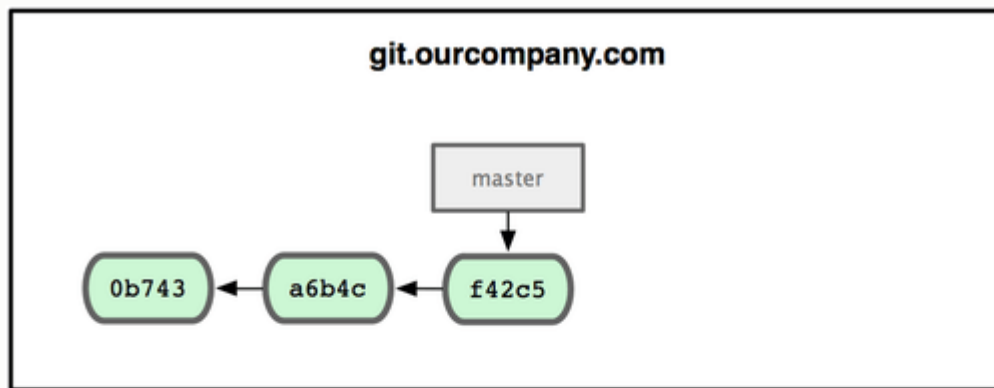
int main(void) {
    printf("Hello Everybody!!!\n");
    return 0;
}
```

Remotes

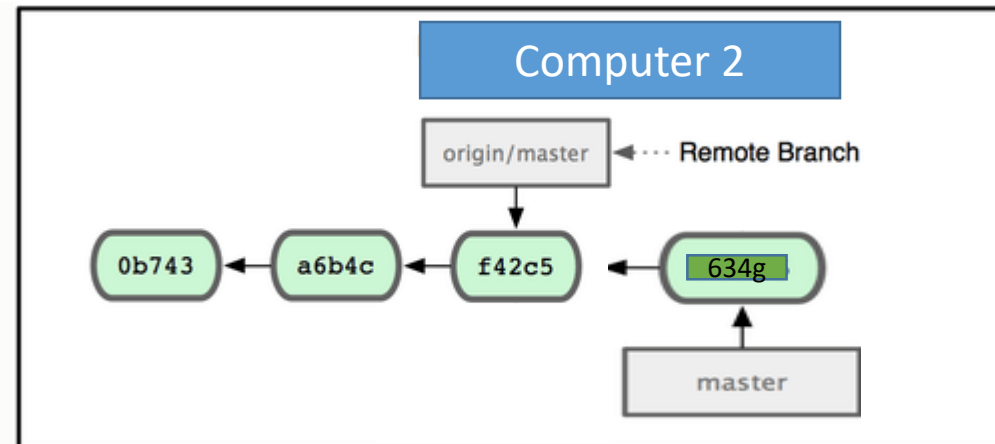
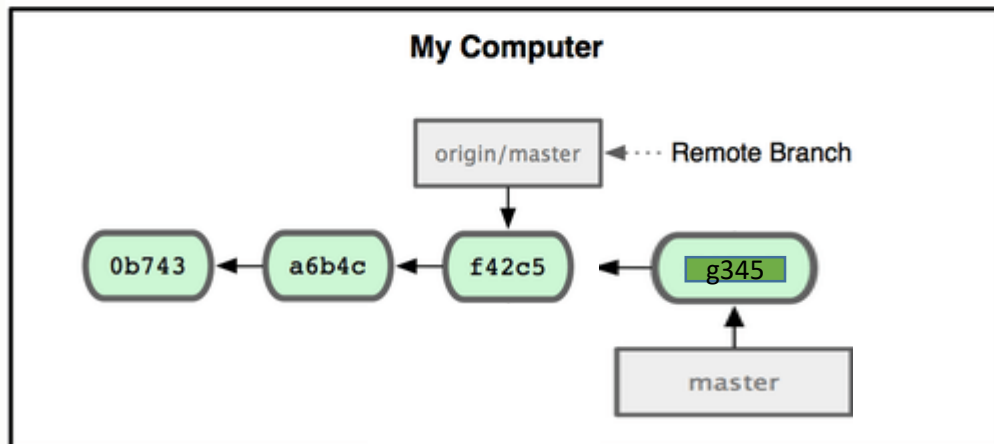


`git clone schacon@git.ourcompany.com:project.git`

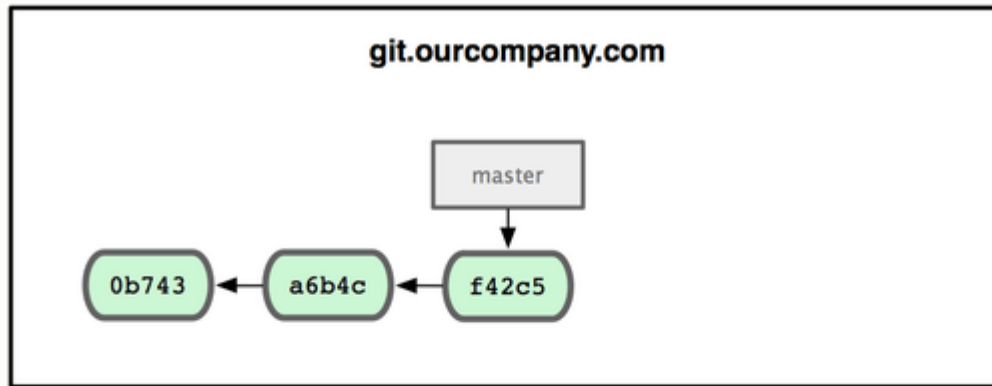




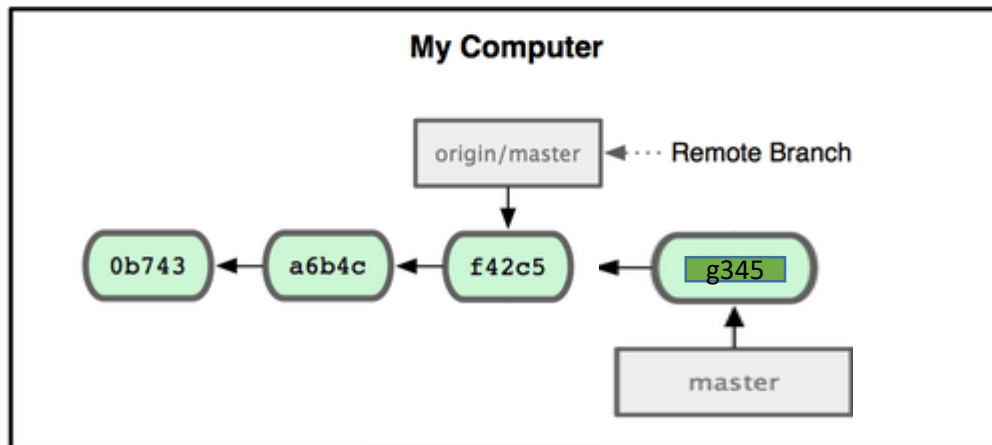
`git clone schacon@git.ourcompany.com:project.git`



Git push origin master



`git clone schacon@git.ourcompany.com:project.git`

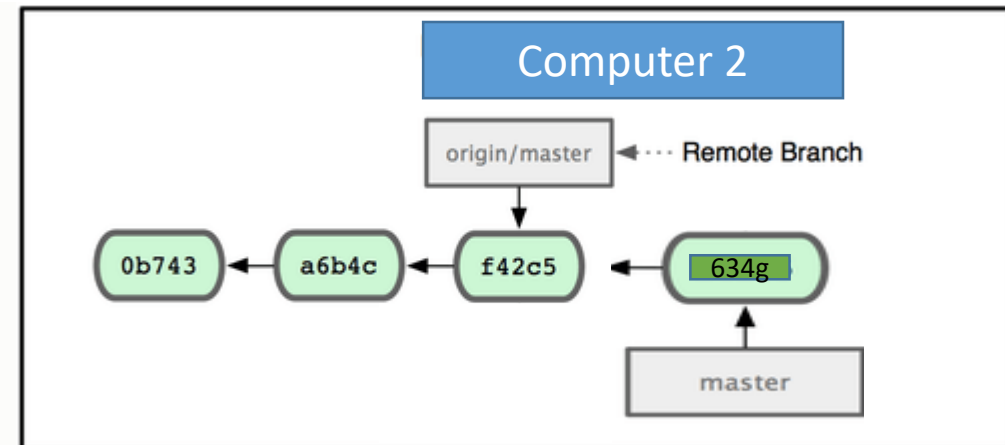


Git push – what happens:

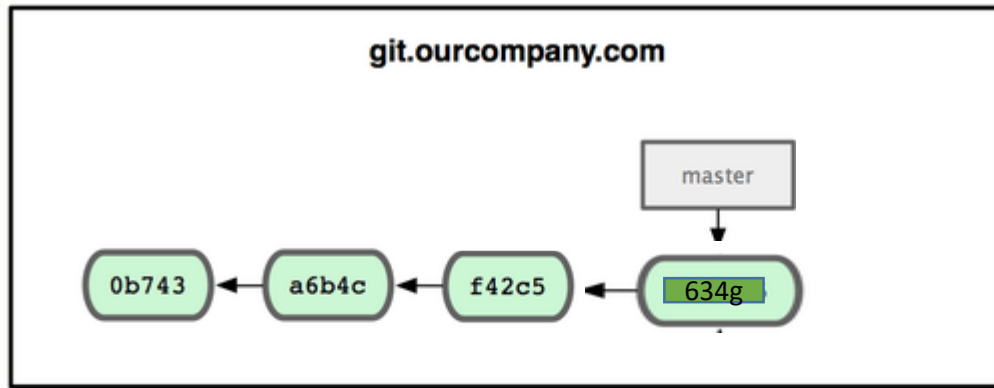
Computer 2: I want to push

Server: I've got master at f42c5

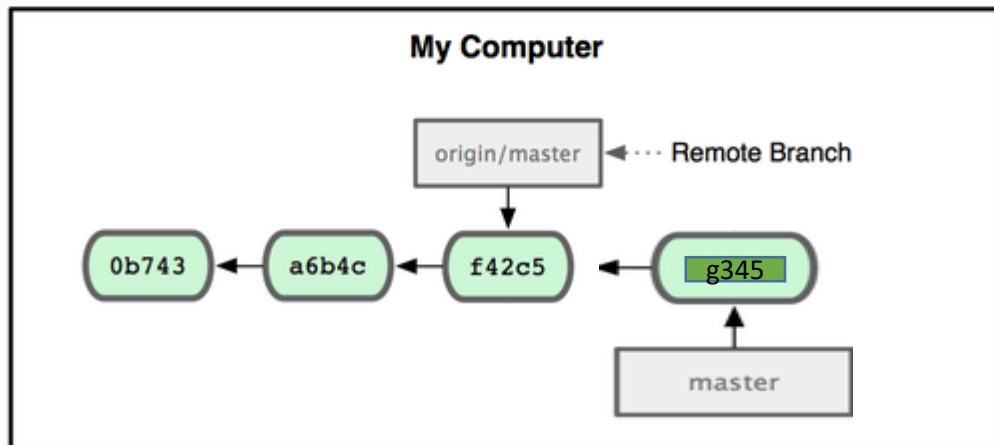
Computer 2: OK, I can see that, here's the difference



Git push origin master



↓
`git clone schacon@git.ourcompany.com:project.git`

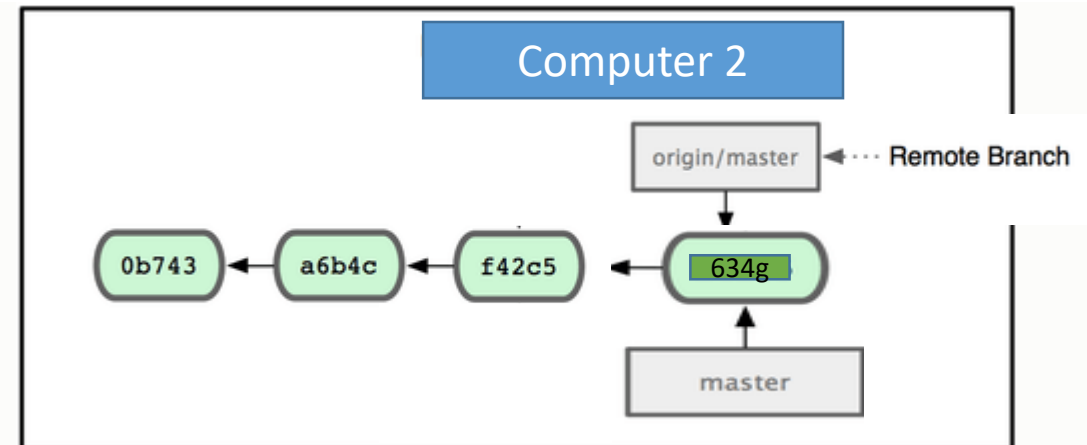


Git push – what happens:

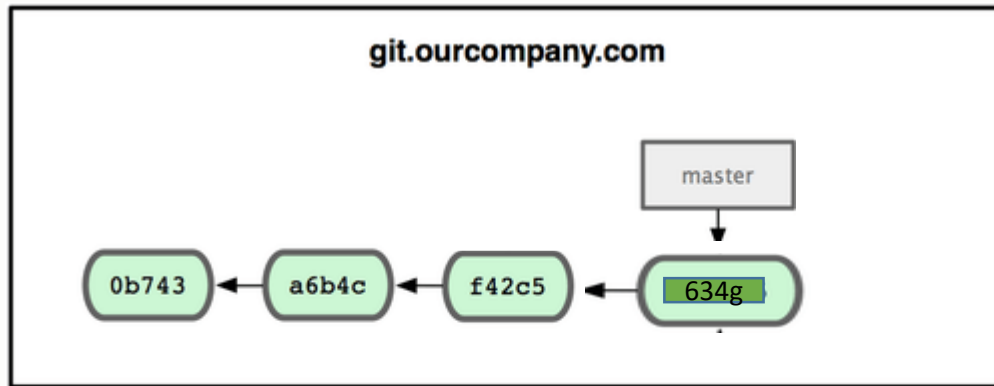
Computer 2: I want to push

Server: I've got master at f42c5

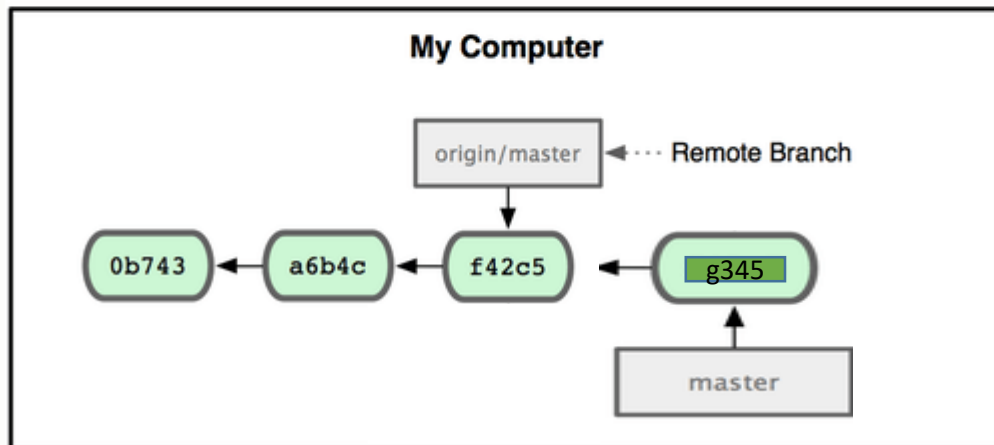
Computer 2: OK, I can see that, here's the difference



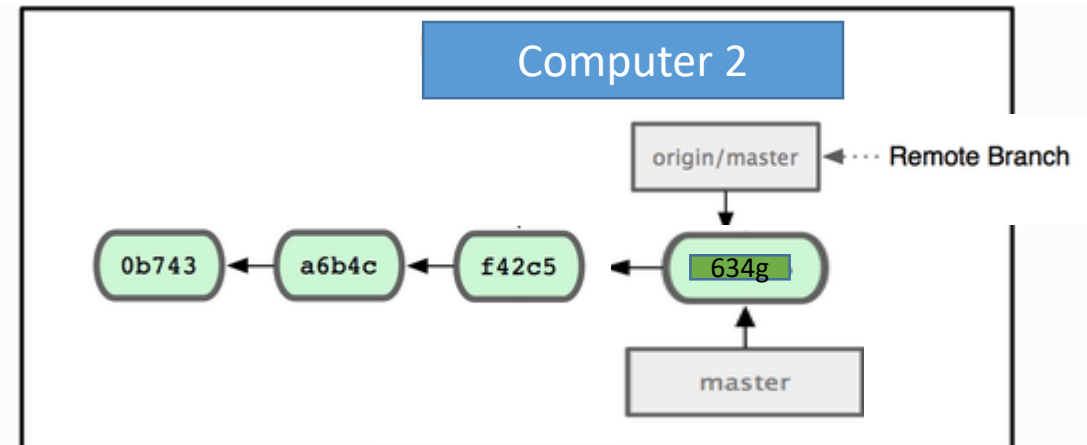
Git push



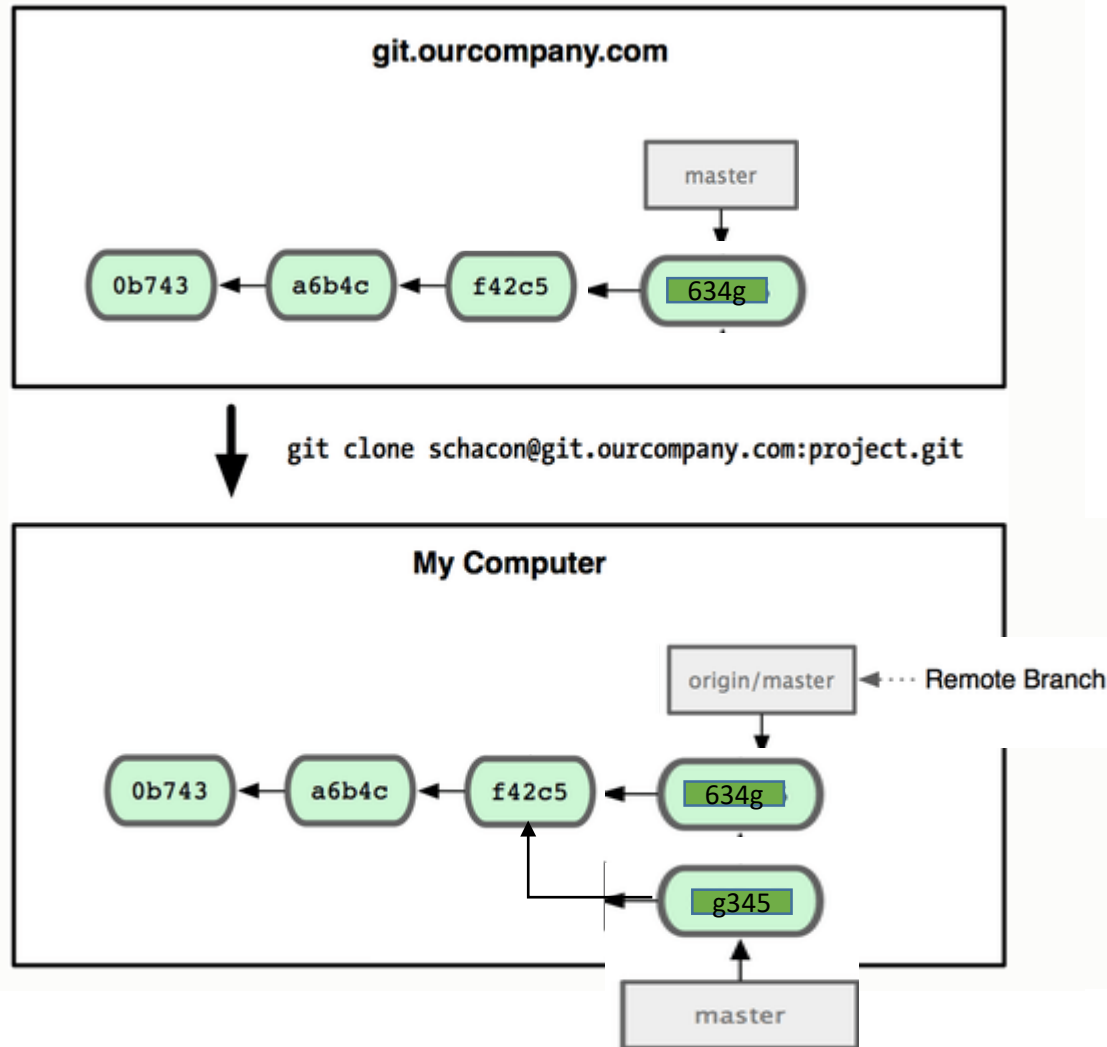
`git clone schacon@git.ourcompany.com:project.git`



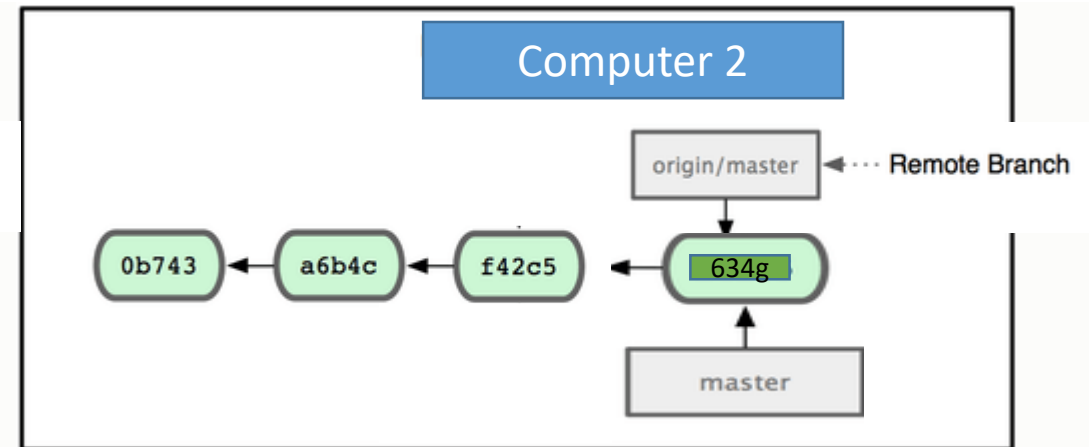
Git push – what happens:
My Computer: I want to push
Server: I've got master at 634g
Computer 2: Doh



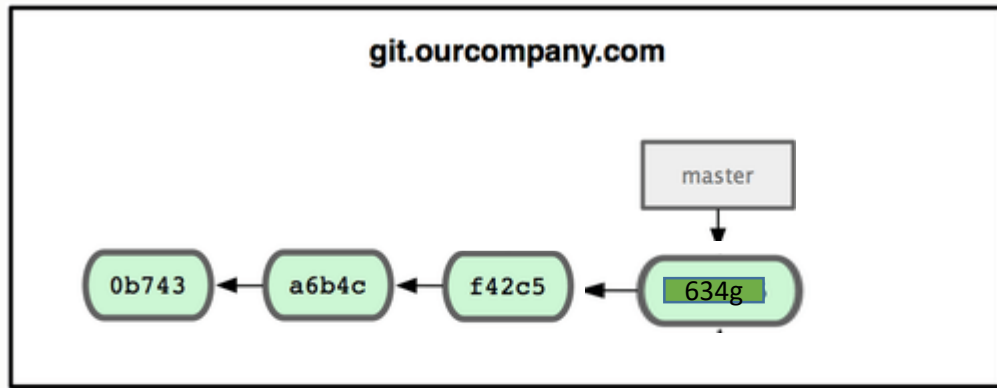
Git fetch



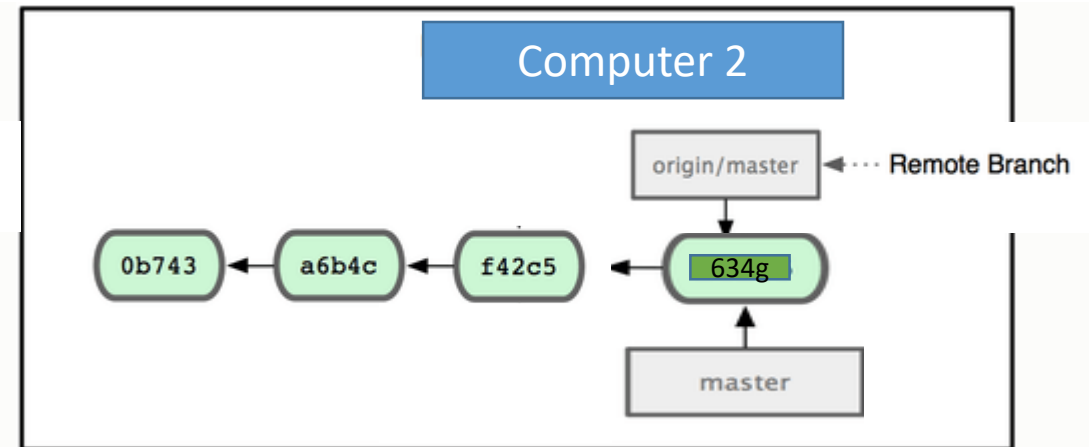
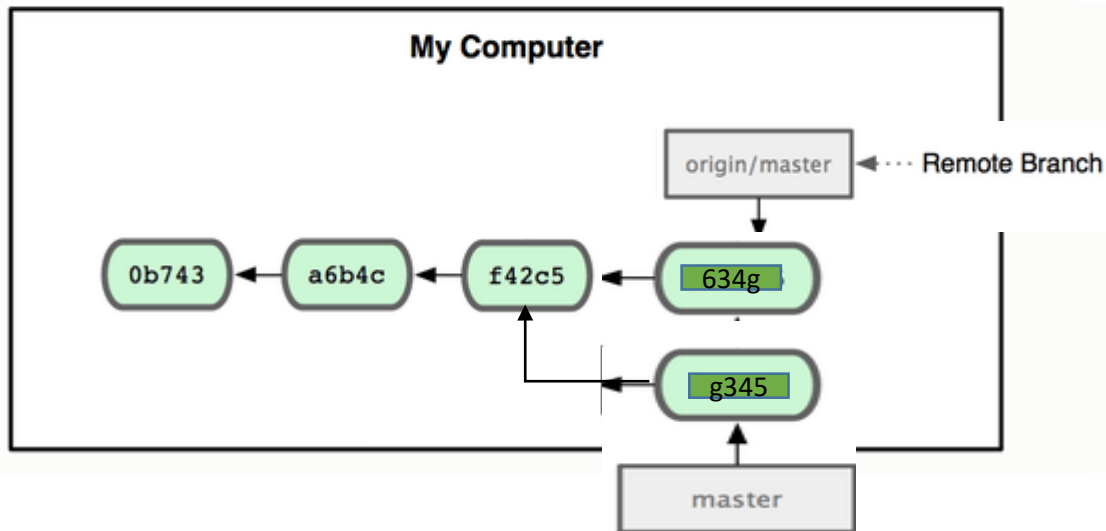
Git push – what happens:
My Computer: I want to push
Server: I've got master at 634g
Computer 2: Doh



Git push try 2

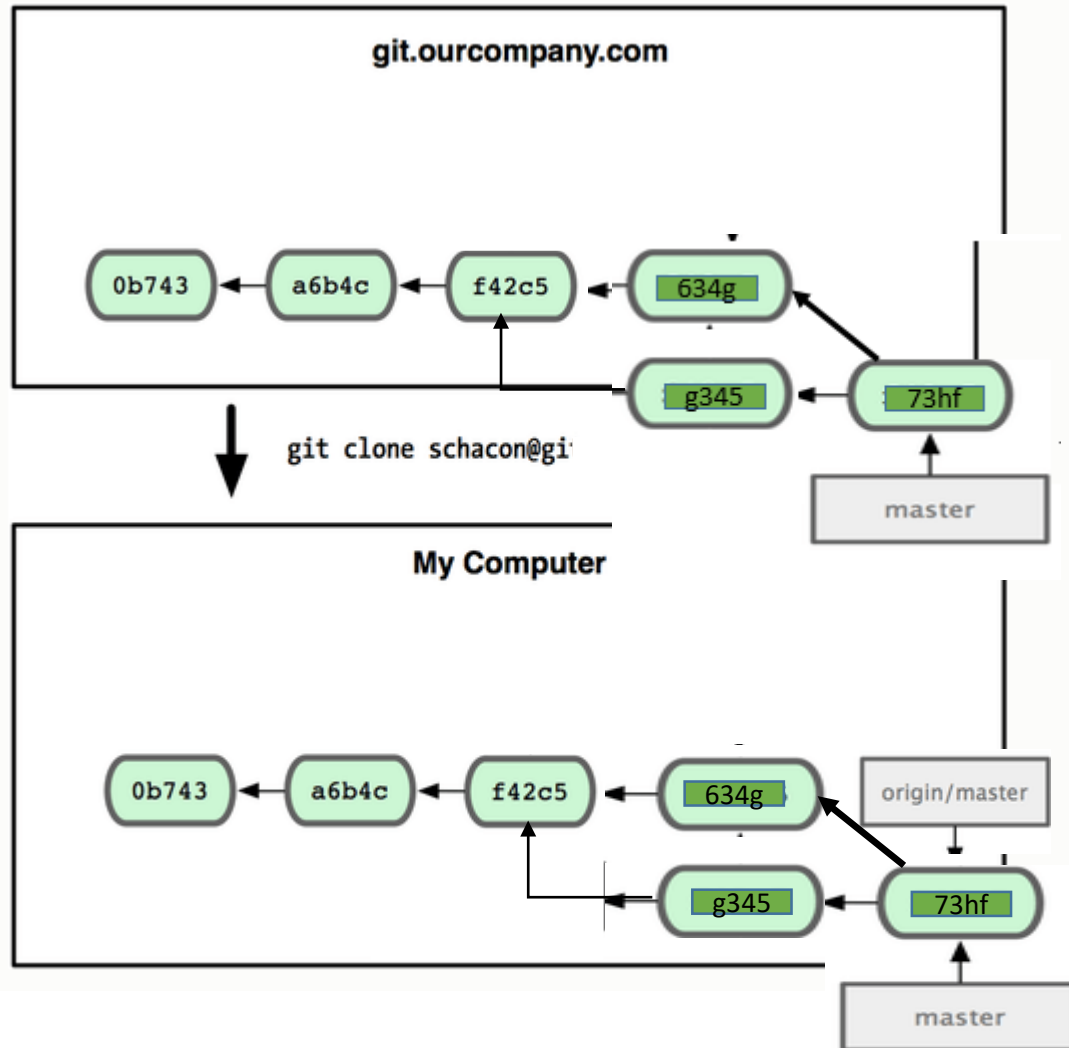


↓
`git clone schacon@git.ourcompany.com:project.git`



Git push – what happens:
My Computer: I want to push
Server: I've got master at 634g
Computer 2: OK, I can see tha
Computer 2: git merge origin/master

Git push try 2

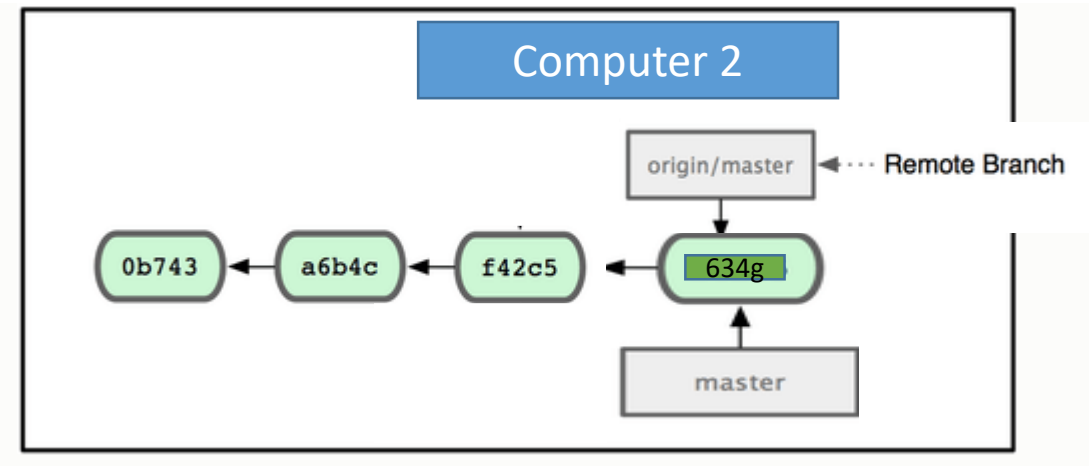


Git push – what happens:

My Computer: I want to push

Server: I've got master at 634g

Computer 2: OK, I can see that, here's the difference



Git push other branches

- You can push any other branch to a server
- `git branch other_branch; checkout other_branch; vim task.c; git commit`
- `git push origin other_branch`
- Other computer types `git fetch`, will get this new branch
- Can type `git checkout other_branch` to use it

Git pull

- Does git fetch and git merge automatically.
- May give too many errors if large time between merges
- You can just stick to git fetch, merge and push

Overall, main git commands

- git init
- git clone
- git add
- git status
- git commit
- git branch
- git checkout
- git merge
- git push
- git fetch
- (git pull)

Git advice

- Commit early and often
- Commit useful messages
- Use branches to experiment (you don't have to push branches, but you should still commit when working on a branch)
- Fetch/merge/push regularly (with code you're happy with)
 - Avoids hassle if you're up to date

Using git

- Follow instructions on git tutorial on Canvas
- (you first need to create a repo for everyone to use, then all group members can clone and use as in this lecture)

Additional resources

- <https://git-scm.com/book/en/v1/Getting-Started>