Git

Presented by Ed Armstrong
Presented for SHARCNET
October 2020

Git

- Open Source
- Distributed
- Revision Control

training.sharcnet.ca
youtube.sharcnet.ca

(https://training.sharcnet.ca/course/view.php?id=48)

What is Revision Control?

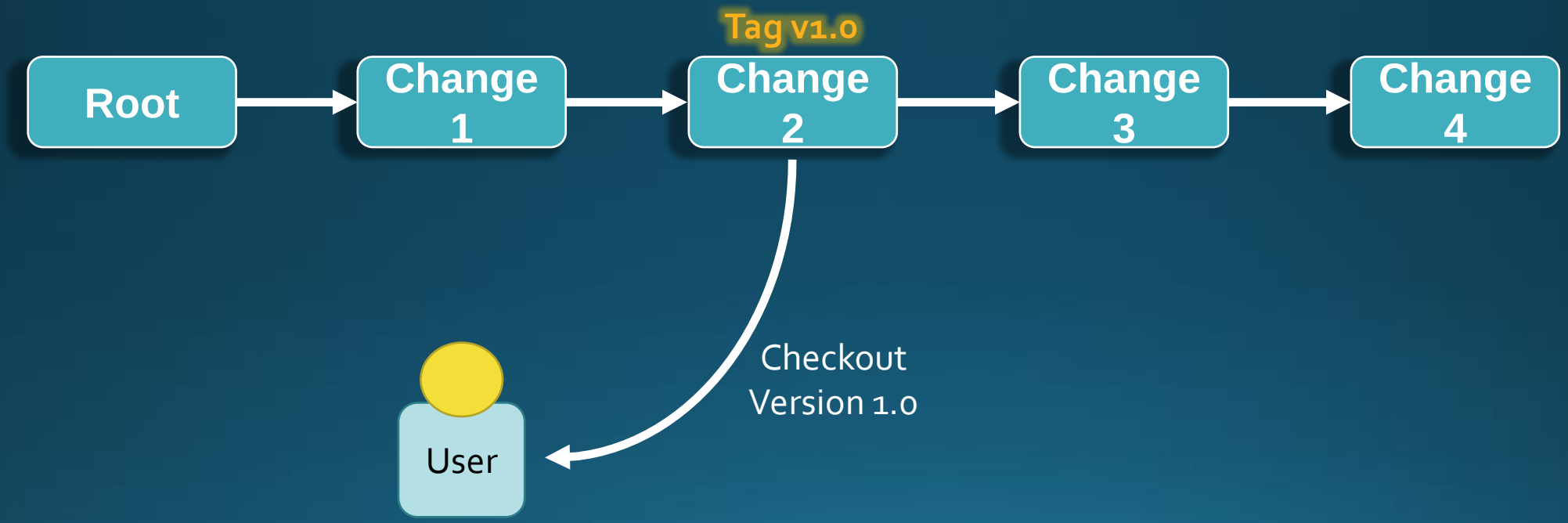
Undo changes to files

Work on multiple branches simultaneously

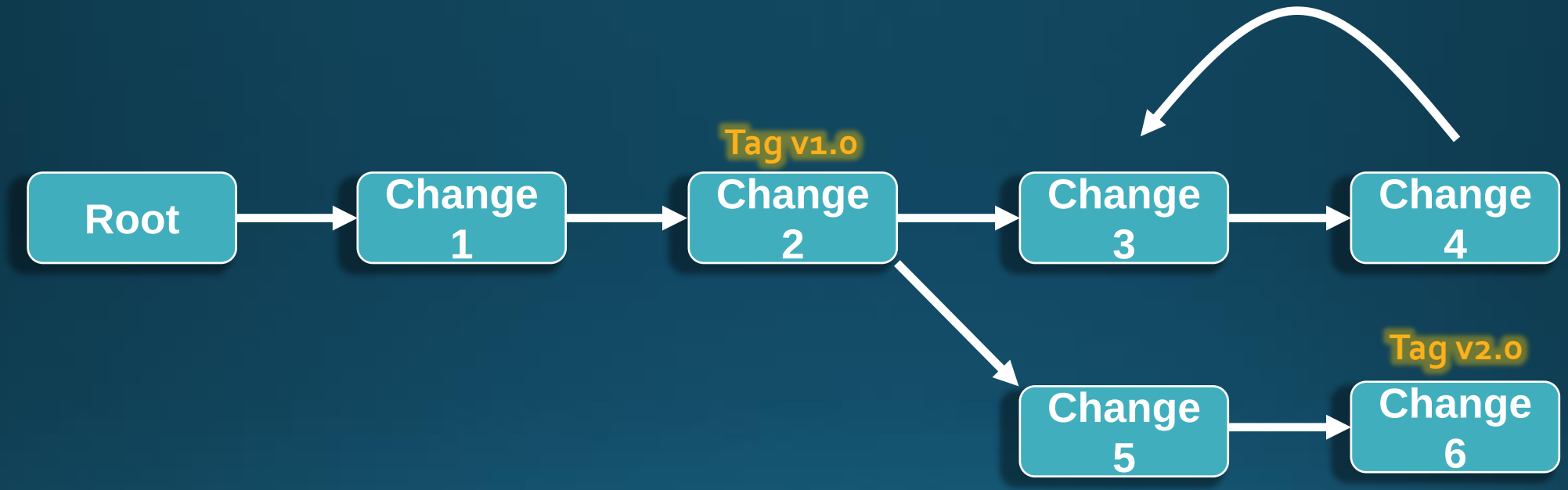
Manage multiple developers

Automatically backup code base

Revision Control



Revision Control



The Core

- Init
- Add
- Commit
- Log
- Status

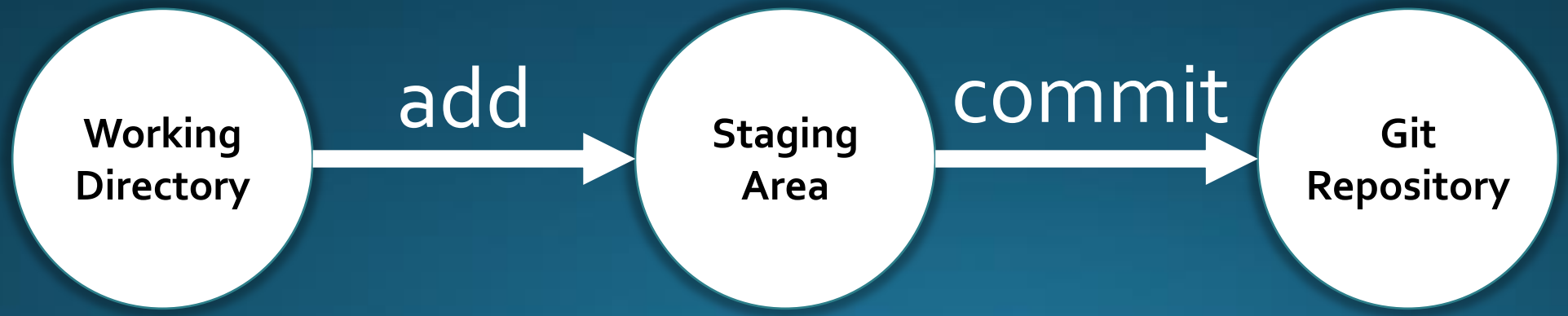
```
git <command> [options]
```

Terminology

1. The Working Directory
2. The Staging Area
3. The Git Repository

Terminology

1. The Working Directory
2. The Staging Area
3. The Git Repository



Init

```
git init [options] [path]


git init <path>
git init --bare <path>
```

Add

```
git add [options] [path]


git add <single file>
git add <directory>
git add .
git add *
```

Commit

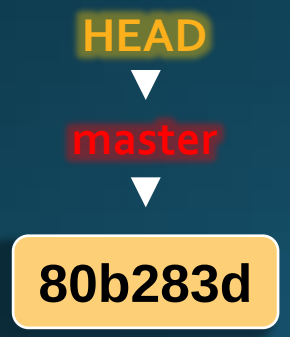
```
git commit [options]


git commit -a
git commit -m "message"
git commit -am "message"
```

Log / Status

```
git log [options] [revison-range] [path]
git status [options] [--] [path]


git log --oneline
git log --oneline --all
git log --graph --oneline
git status
```

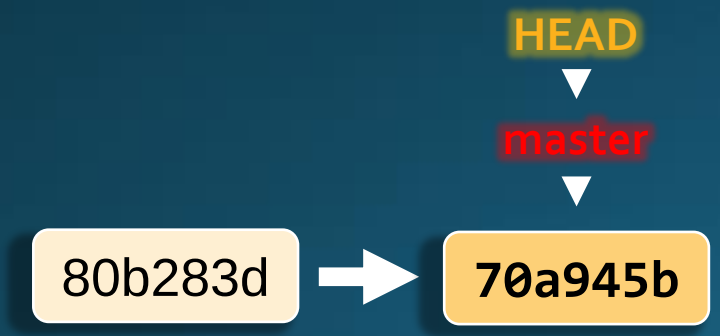
Head Node

```
> git add .
> git commit -m"initial commit"
> git log --oneline
80b283d (HEAD -> master) initial commit
```



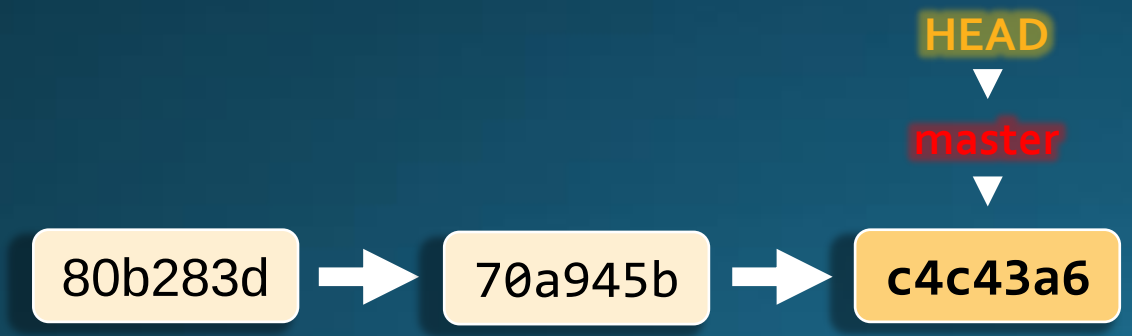
Head Node

```
> git add .
> git commit -m"added file"
> git log --oneline
70a945b (HEAD -> master) added file
80b283d initial commit
```



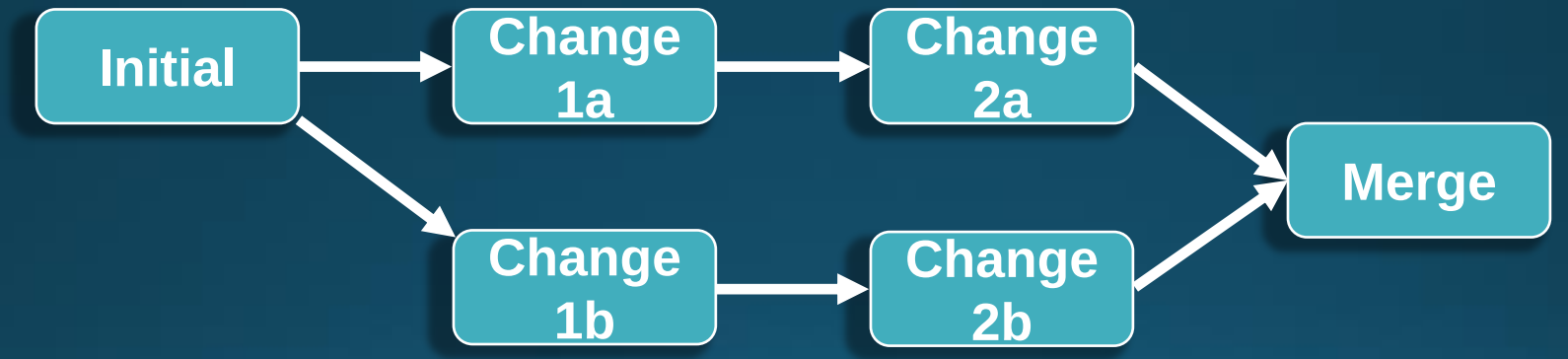
Head Node

```
> git add .
> git commit -m"added file"
> git log --oneline
c4c43a6 (HEAD -> master) changed file
70a945b added count
80b283d initial commit
```



Branching

Branching



Branching

Situation: We want to edit some files but keep them isolated until we are satisfied with them.

Use: `git branch <branch_name>`

`git checkout <branch_name>`

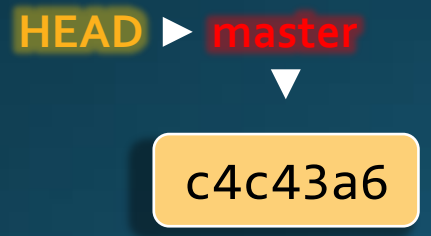
`git merge <branch_name>`

Result: We create a new branch and update it while continuing to update the master branch as well. In the end we will merge both branches together.

Branching

create the repository

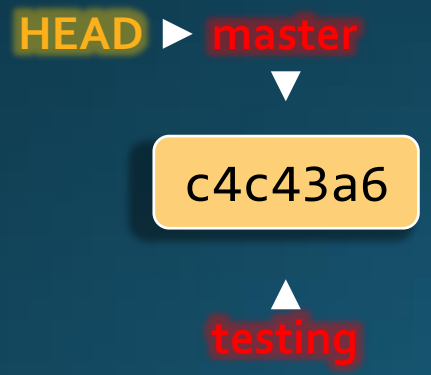
```
> git add helloworld.c Makefile assets.txt
> git commit -m"initial"
```



Branching

create the repository

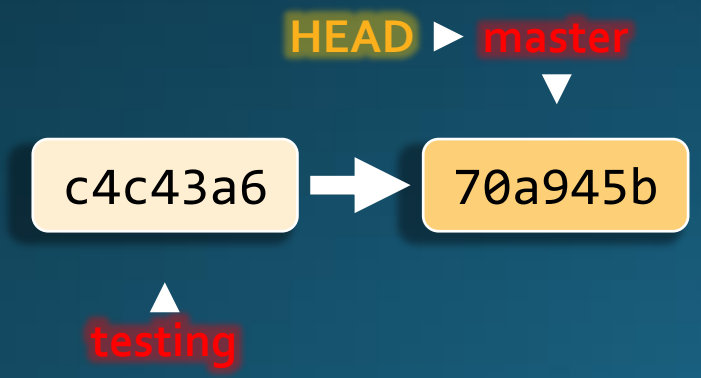
```
> git add helloworld.c Makefile assets.txt
> git commit -m"initial"
> git branch testing
```



Branching

add to master branch

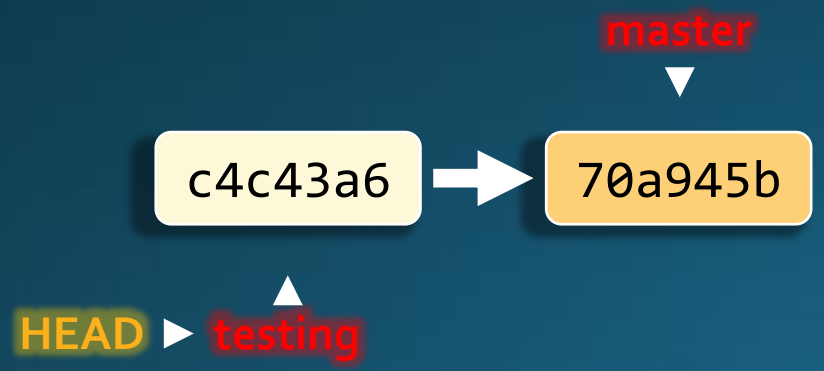
```
> git add library.c
> git commit -m"added library file"
```



Branching

switch branches

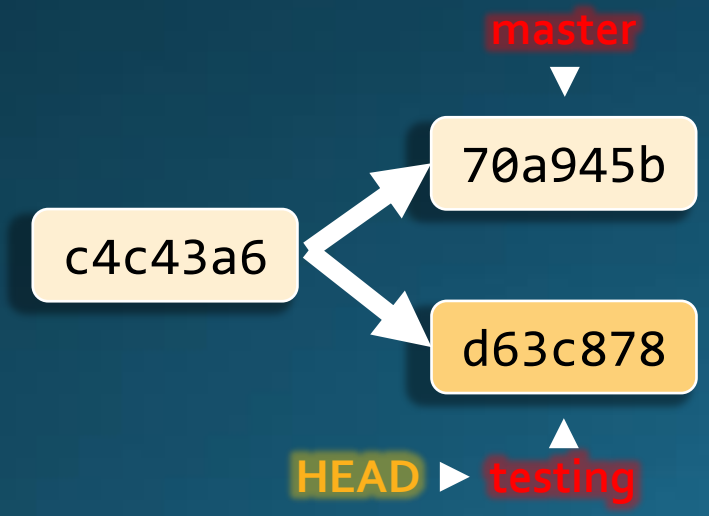
```
> ls
Makefile  assets.txt  helloworld.c  library.c

> git checkout testing
> ls
Makefile  assets.txt  helloworld.c
```



Branching

add to testing branch

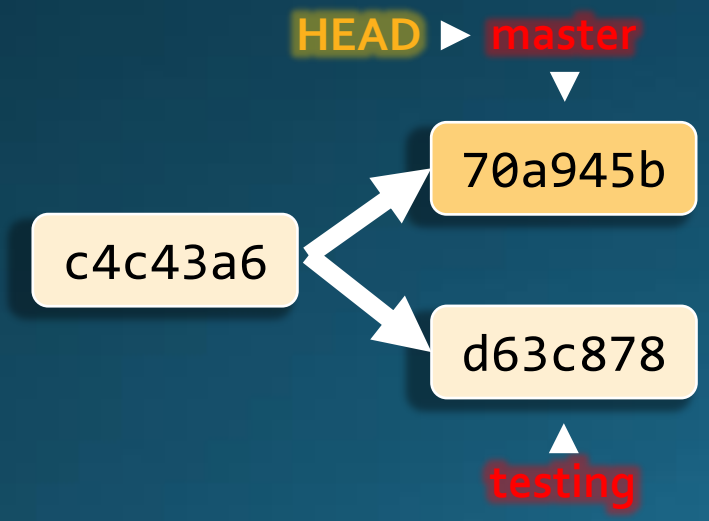
```
> git add testing.c
> git commit -m"added testing file"
git log --oneline --all
d63c878 (HEAD -> testing) added testing file
70a945b (master) added library file
c4c43a6 init
```



Branching

checkout master branch

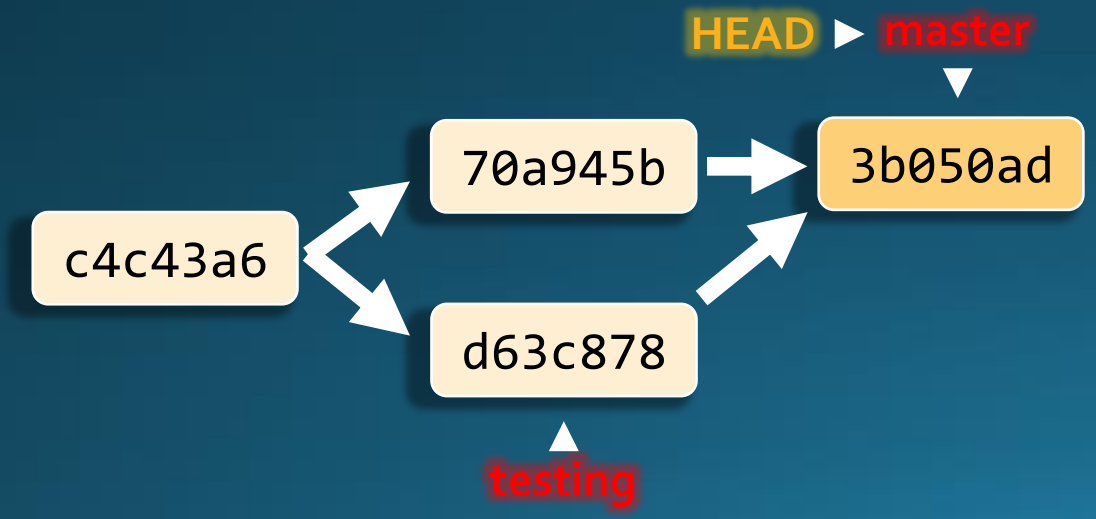
```
> git checkout master
> ls
> Makefile  assets.txt  helloworld.c  library.c
> git log --oneline --all
d63c878 (testing) added testing file
70a945b (HEAD -> master) added library file
c4c43a6 init
```



Branching

merge the branches

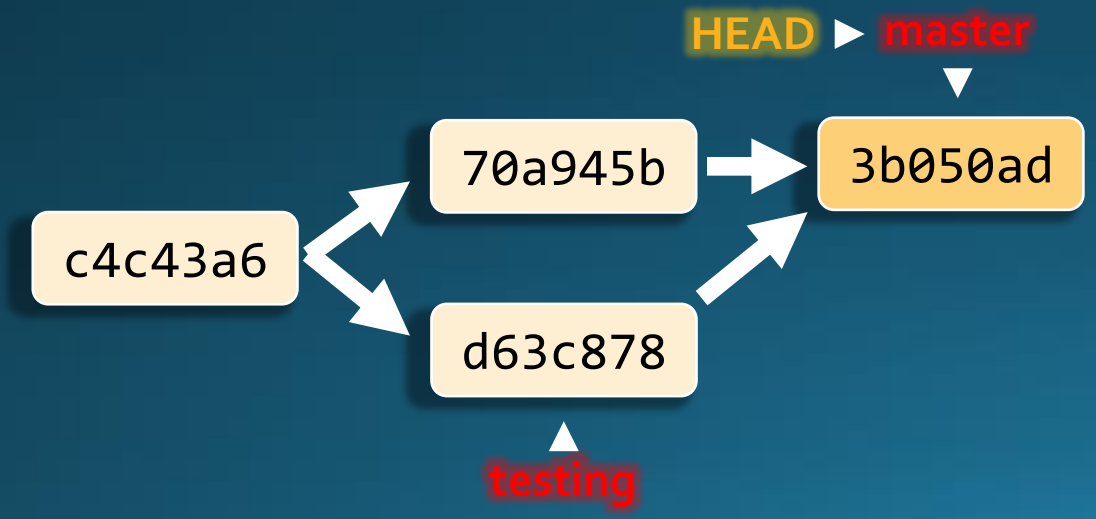
```
> git merge testing
Merge made by the 'recursive' strategy.
 testing.c | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testing.c
```



Branching

merge the branches

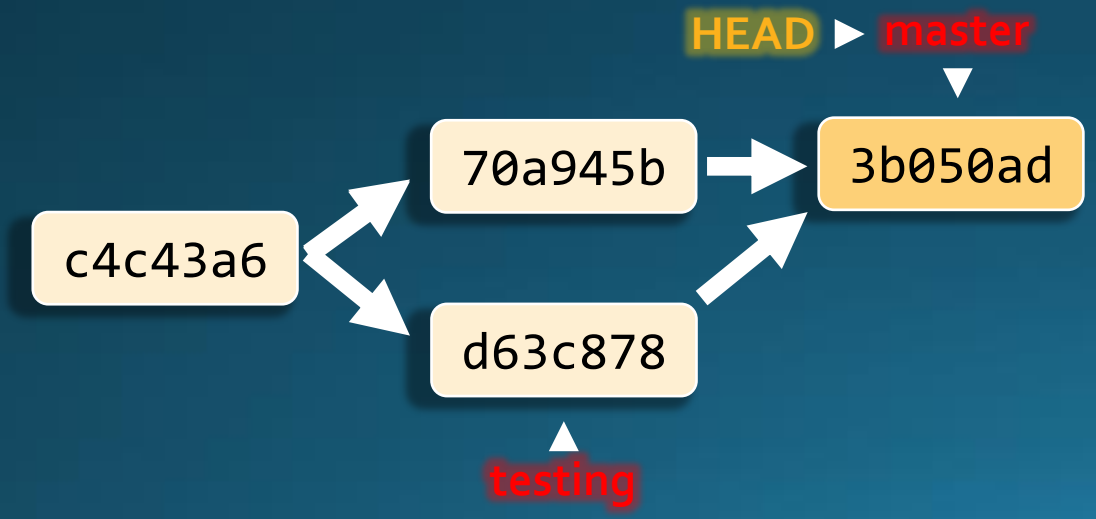
```
> ls
> Makefile  assets.txt  helloworld.c  library.c  testing.c
```



Branching

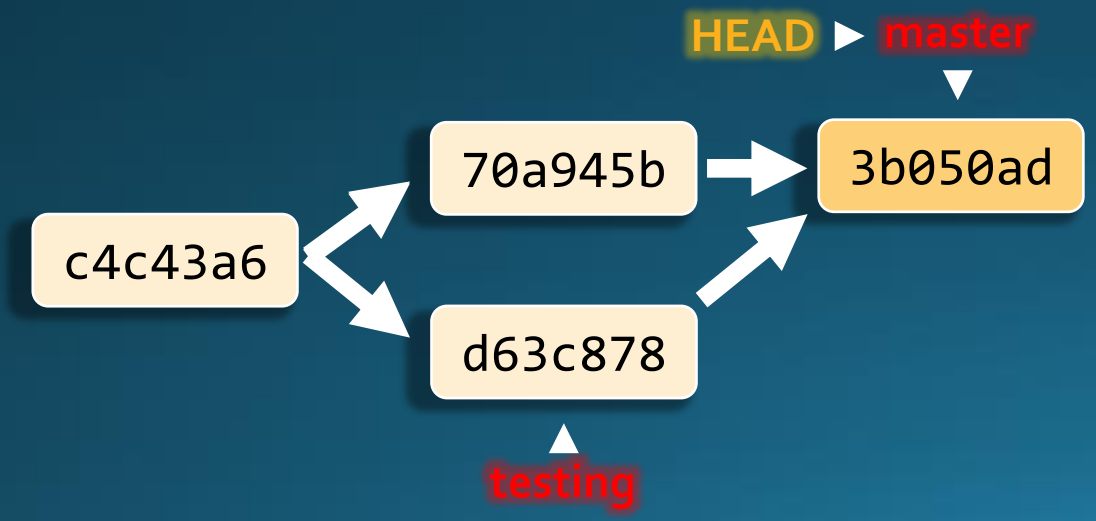
view the graph

```
> git log --oneline --all --graph
*   3b050ad (HEAD -> master) Merge branch 'testing'
|\
| * d63c878 (testing) added testing file
* | 70a945b added library file
|/
* c4c43a6 init
```



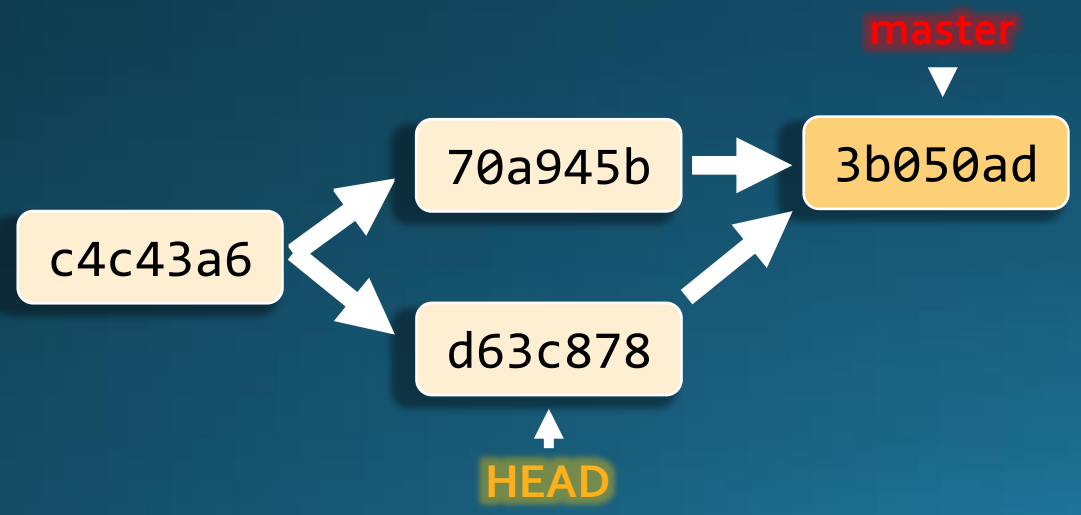
Delete Branch

```
> git branch -d testing
Deleted branch testing (was d63c878)
> git log --oneline
3b050ad (HEAD -> master) Merge branch 'testing'
d63c878 added testing file
70a945b added library file
c4c43a6 init
```



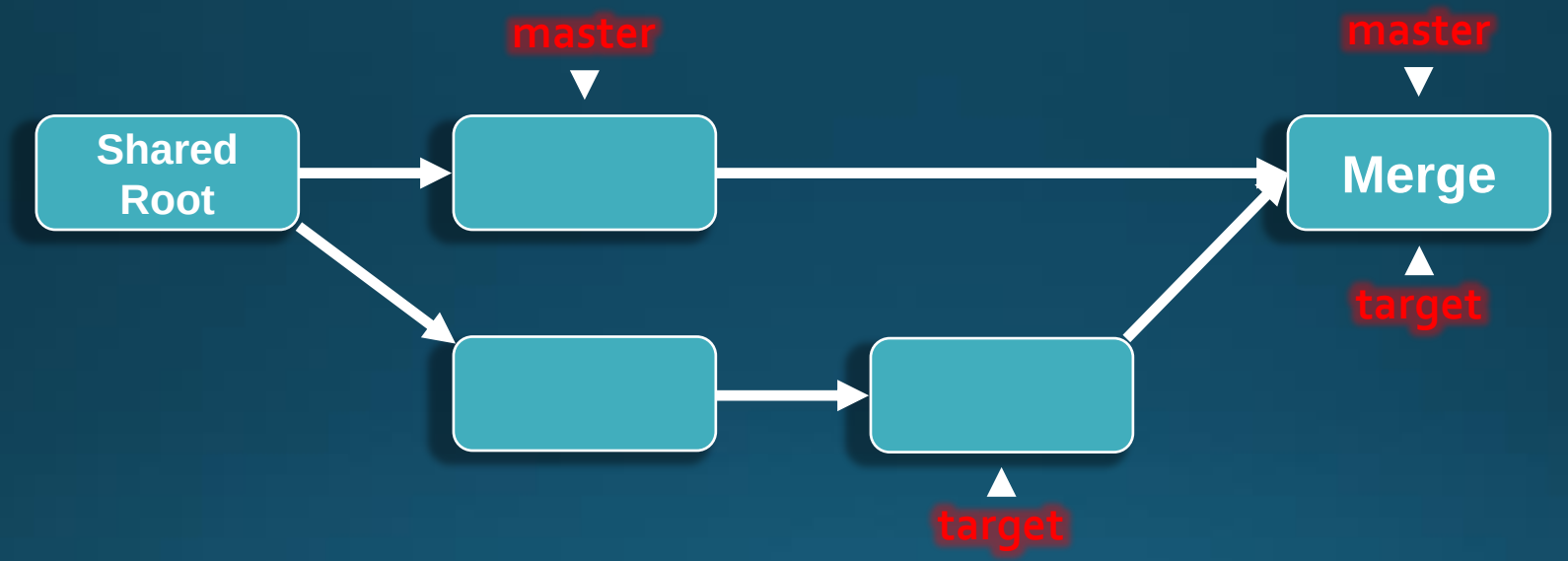
Delete Branch

```
> git checkout d63c
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again.
```



Merging

Merging



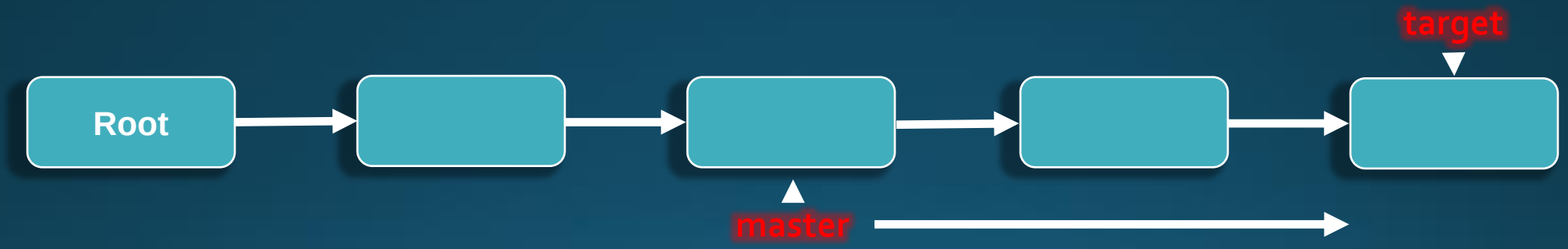
Merging Checklist

1. Checkout the branch you want to merge into, use git status to ensure the HEAD is pointing at the DESTINATION branch

2. Fetch any remote changes (next section)

3. Execute 'git merge target' providing the name of the SOURCE branch.

Fast Forward



Conflicts

1. Git fails during the merge due to both branches having edits on the same file.

2. Git fails to start due to changes in the working directory, or staging area.

Merging

Situation: You have two branches that you want to merge. Each branch has made edits to the same file.

Use:
```
git branch <branch_name>
git checkout <branch_name>
git merge <branch_name>
```

Result: We edit the file and recommit the merge.

How we got there?

1. Create a new repository
2. Add a single file to the MAIN branch
3. Create and checkout a TARGET branch
4. Append a line to the file
5. Checkout the MAIN branch
6. Append a line to the file
7. Attempt to MERGE

How do we fix it?

1. Manually edit the file
2. Abort the merge
3. Specify which branch to keep

Merge File Formatting

```
> cat merge.txt
this has no conflicts
<<<<<<< HEAD
the master branch conflicted text
=======
the target branch conflicted text
>>>>>>> target
this has no conflicts
```

Remotes

Remote

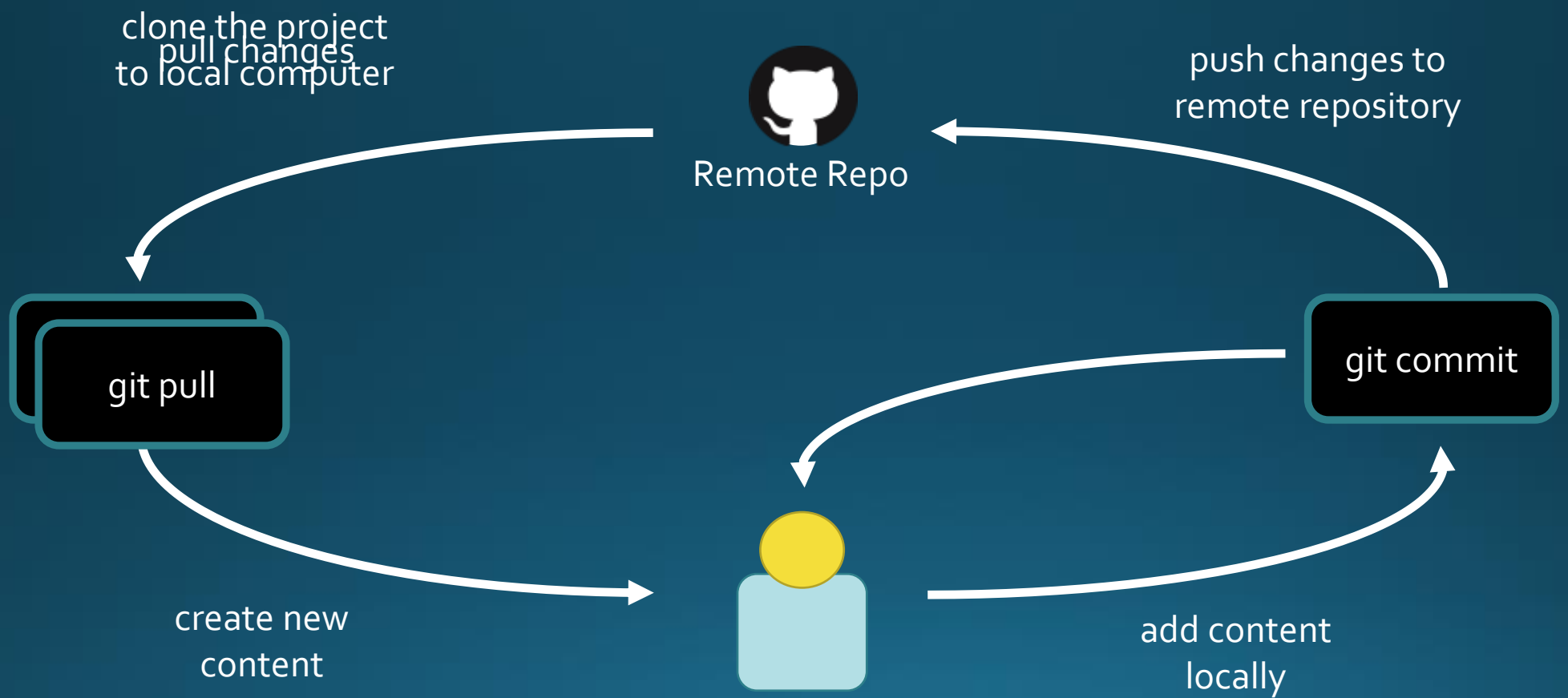
Source Forge

GitLab

GitHub

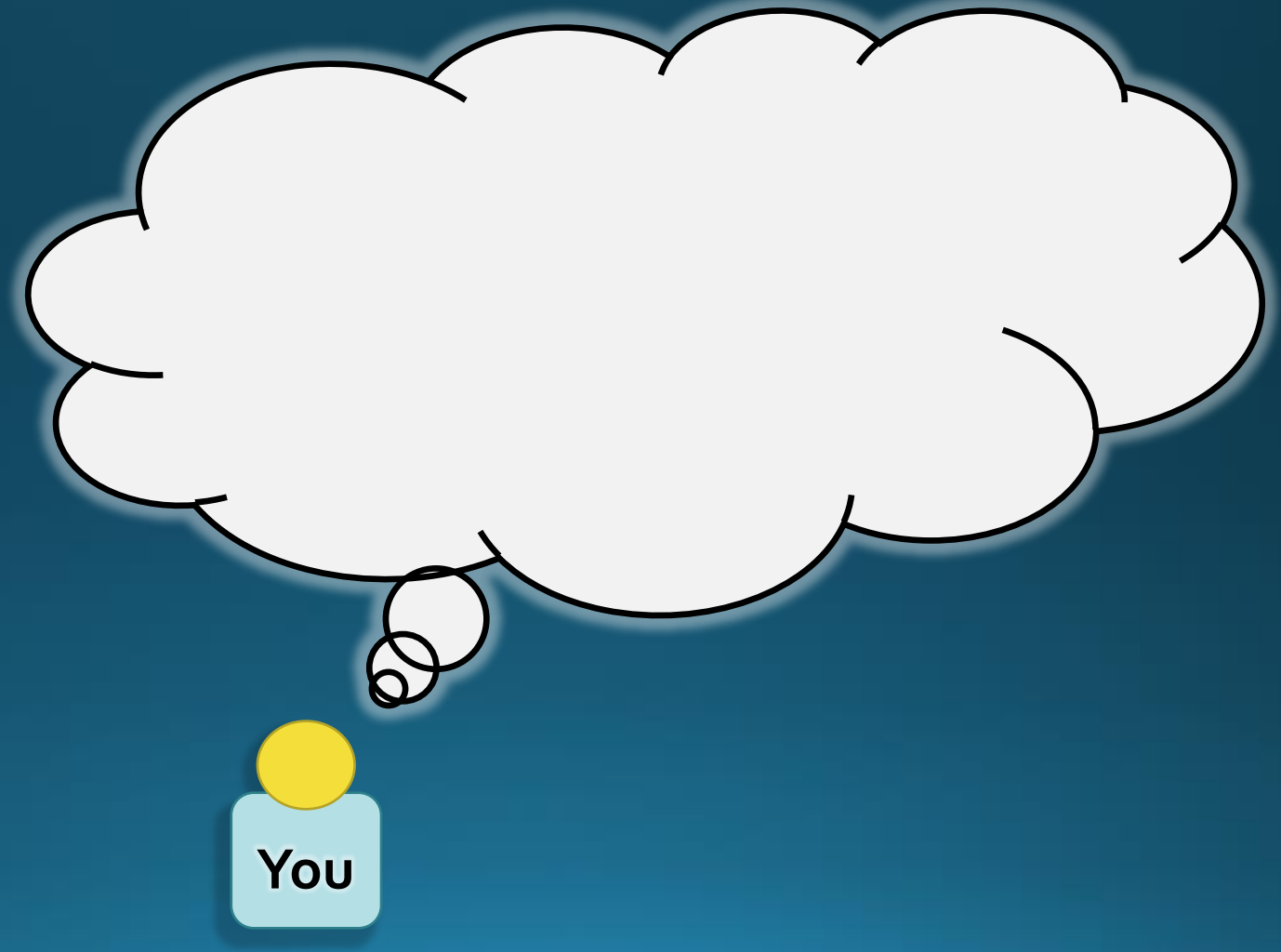
git.sharcnet.ca

Remote

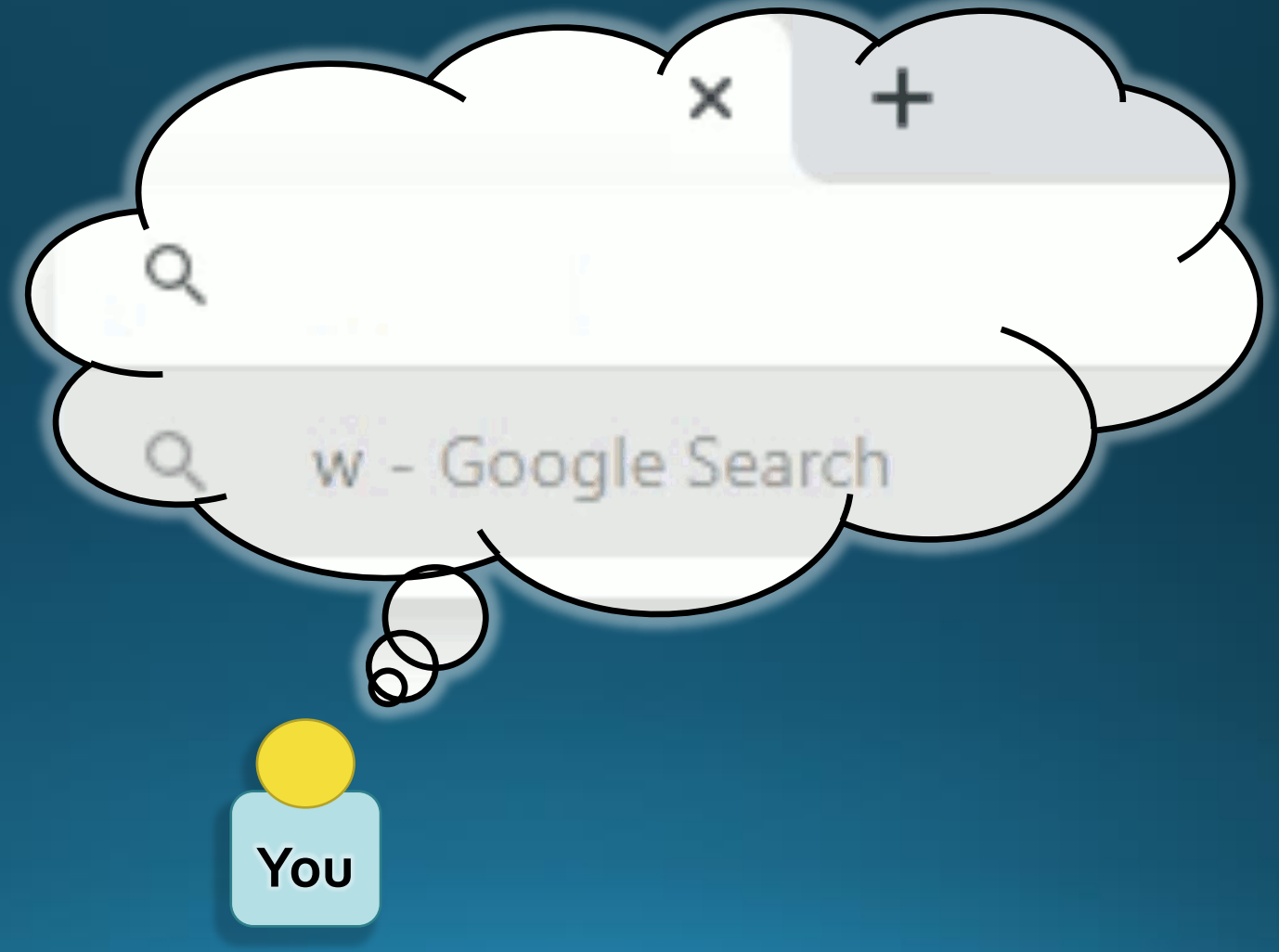

clone the project
pull changes
to local computer
Remote Repo
push changes to
remote repository
git pull
git commit
create new
content
add content
locally

Remote Repositories

Remote Repositories



Remote Repositories



Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:Thaerious/my_new_repo.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# my_new_repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:Thaerious/my_new_repo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:Thaerious/my_new_repo.git
git branch -M main
git push -u origin main
```

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:Thaerious/my_new_repo.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

```
> git clone git@github.com:username/my_new_repo.git
```

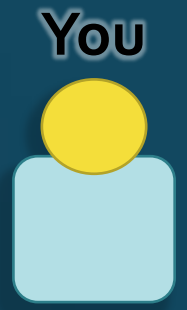
...or create a new repository on the command line

```
echo "# my_new_repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:Thaerious/my_new_repo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:Thaerious/my_new_repo.git
git branch -M main
git push -u origin main
```

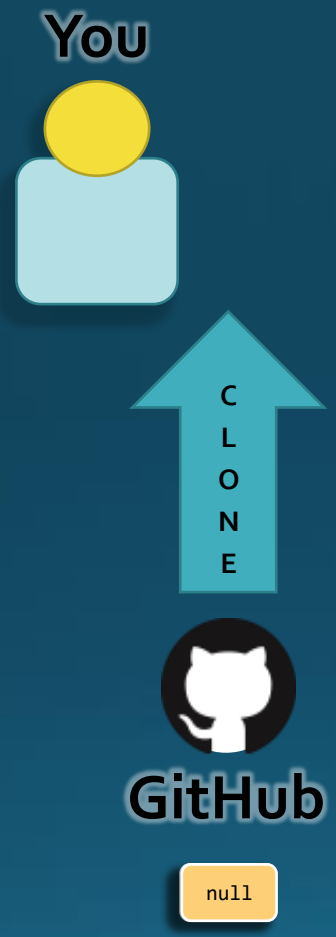
Remote Repositories



null

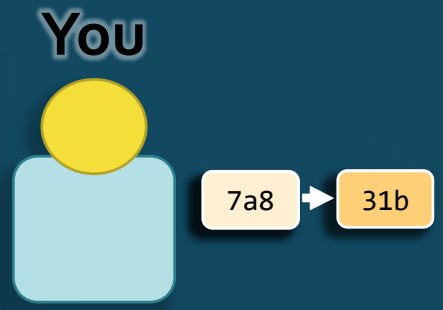
```
> git clone git@github.com:username/my_new_repo.git
```

Remote Repositories



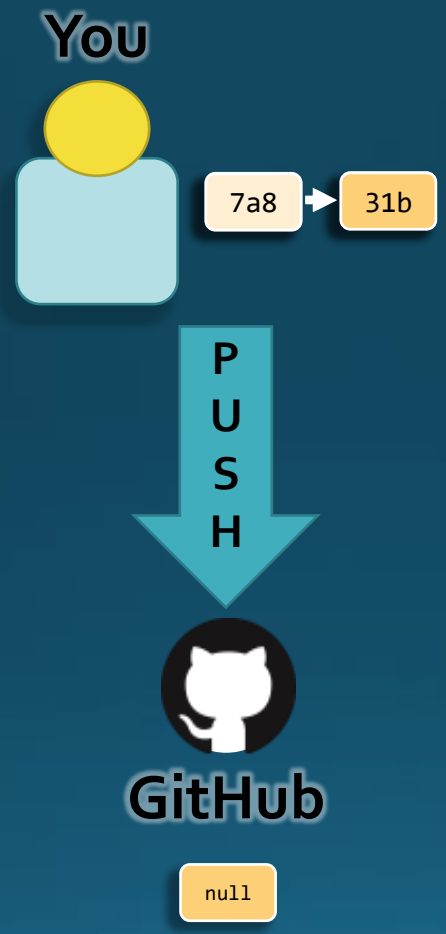

```
> git clone git@github.com:username/my_new_repo.git
```

Remote Repositories



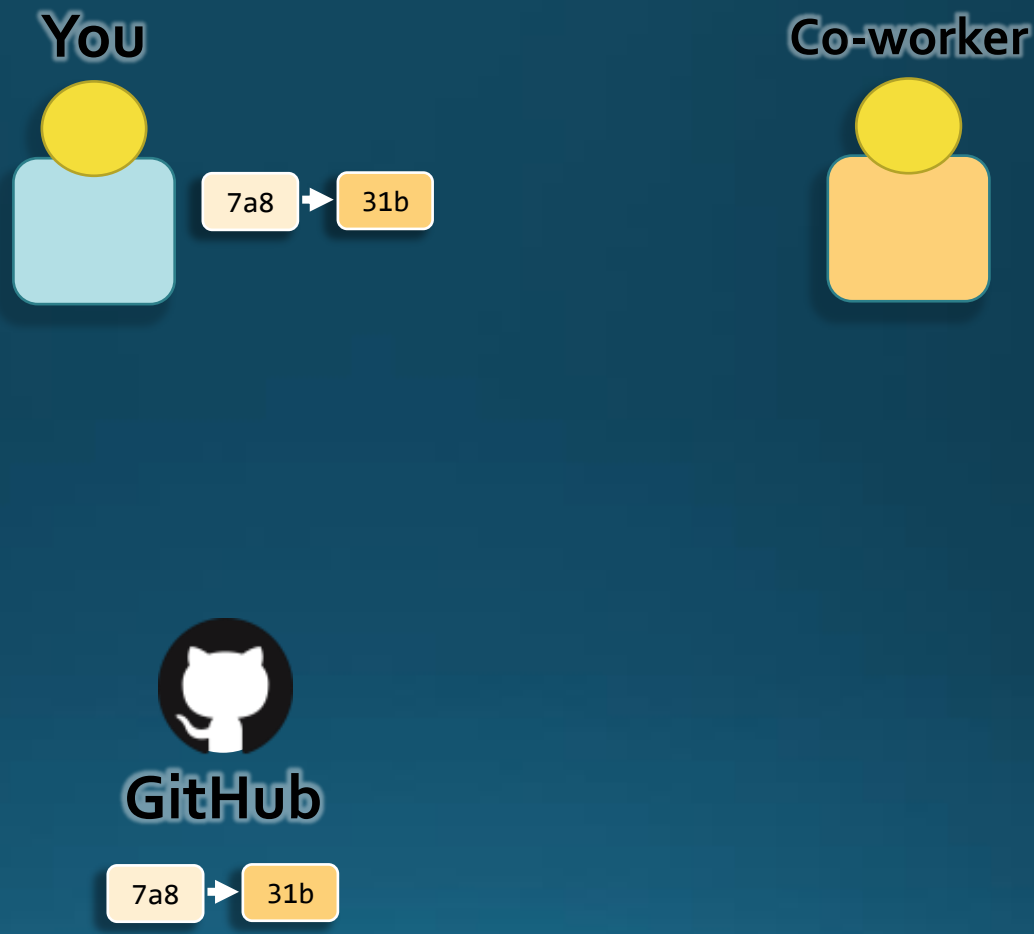

```
> git add *; git commit -m"first"
```

Remote Repositories



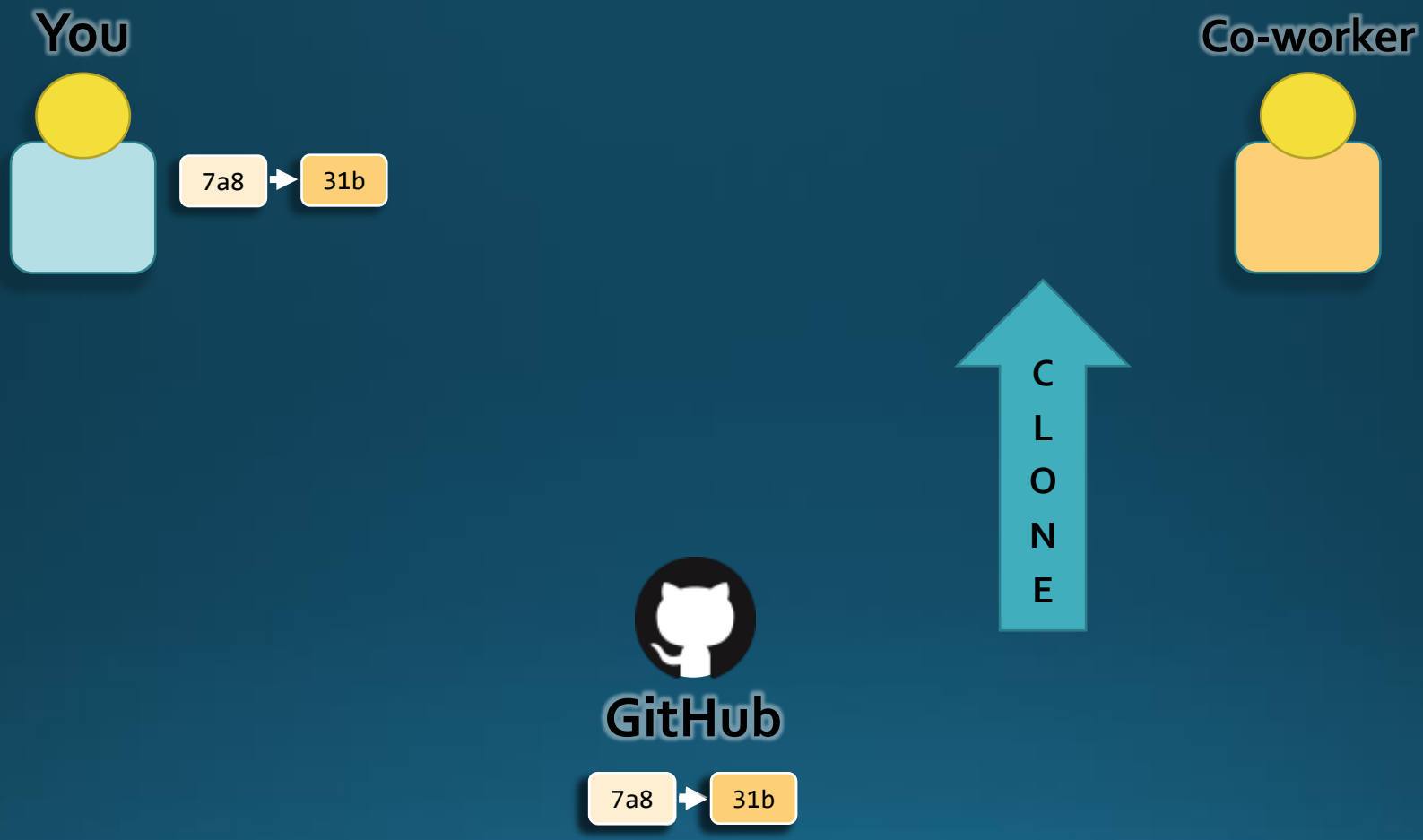

```
> git push
```

Remote Repositories



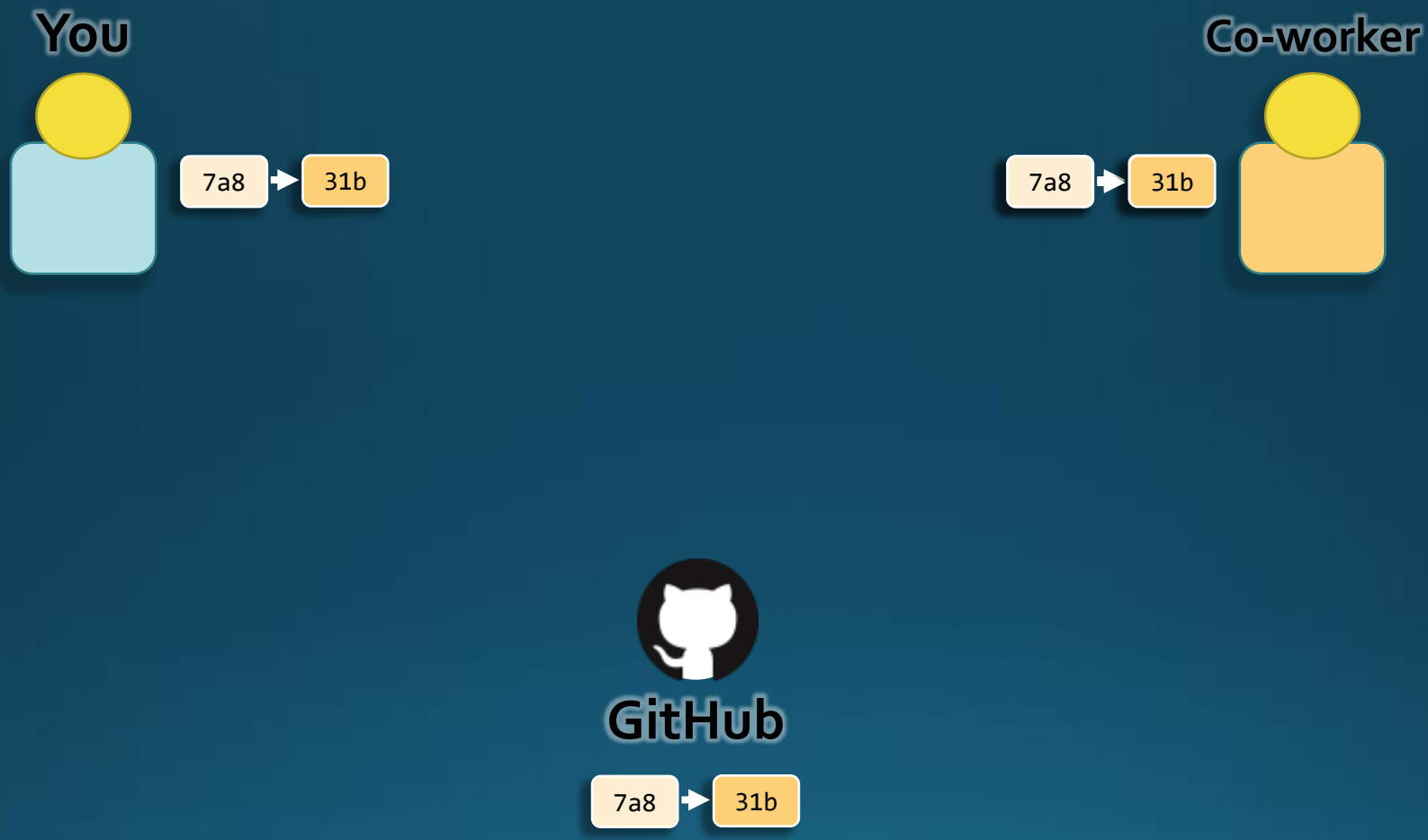
>

Remote Repositories



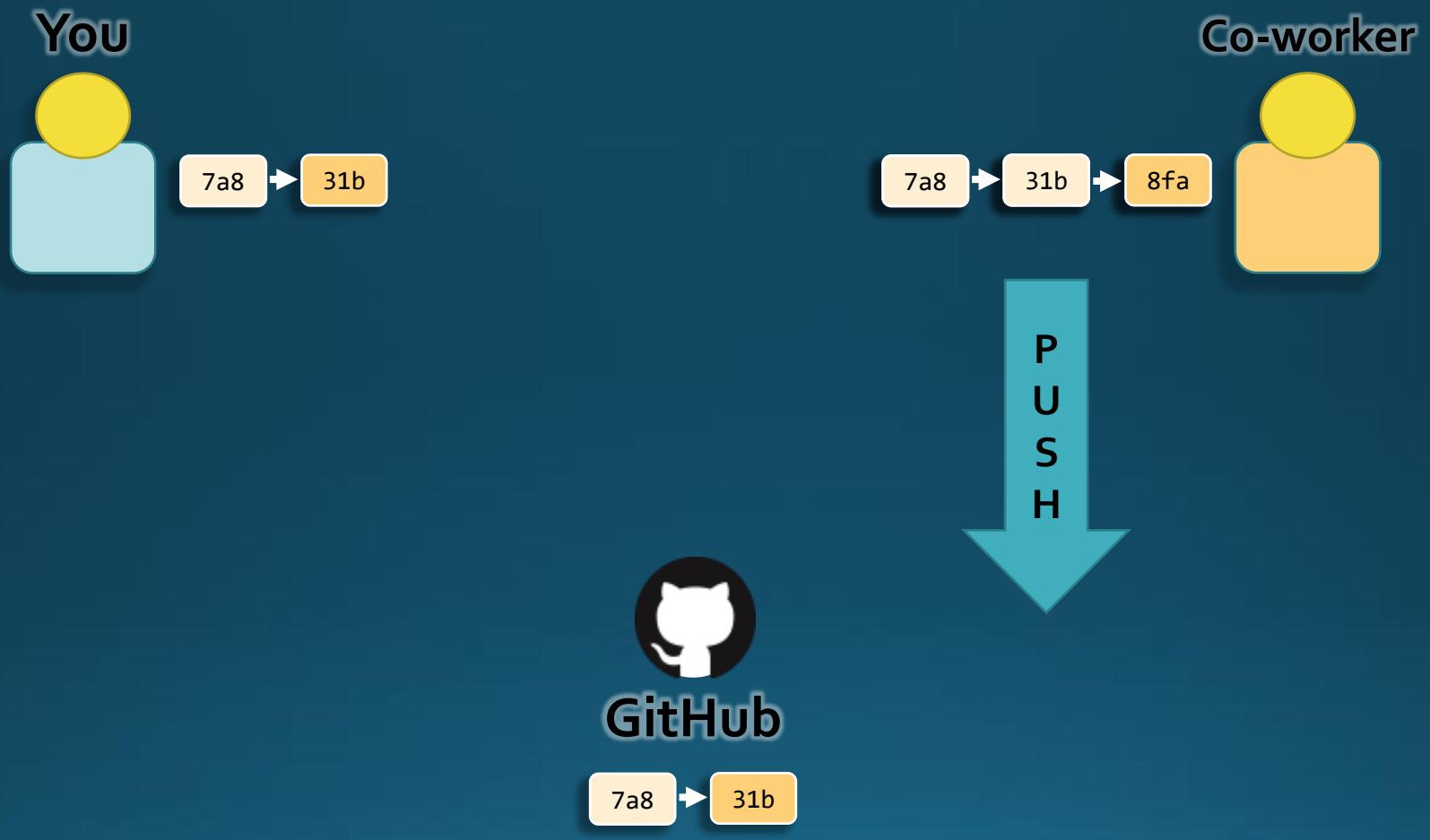

```
> git clone git@github.com:username/my_new_repo.git
```

Remote Repositories



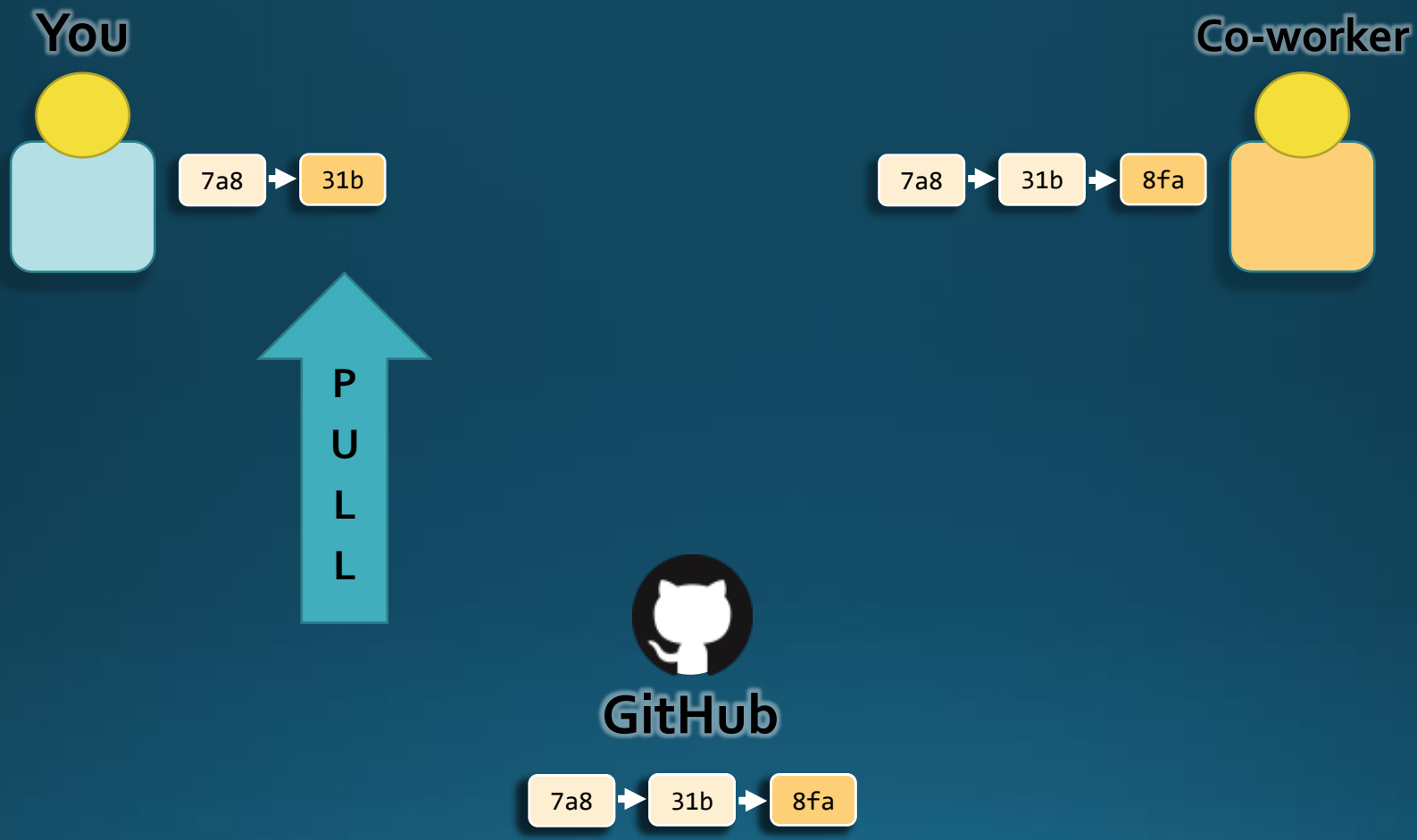

```
> git commit -a -m"made a change"
```

Remote Repositories




```
> git push
```

Remote Repositories




```
> git pull
```

Remote Commands

```
git clone <remote uri>
git pull <remote name>
git push <remote name> <branch> [--all]

git remote -v
git remote add <shortname> <url>
git remote show <remote>
git remote rename <old> <new>
git remote remove <remote>
git fetch <remote>
```

Undo

Undoing Uncommitted Changes

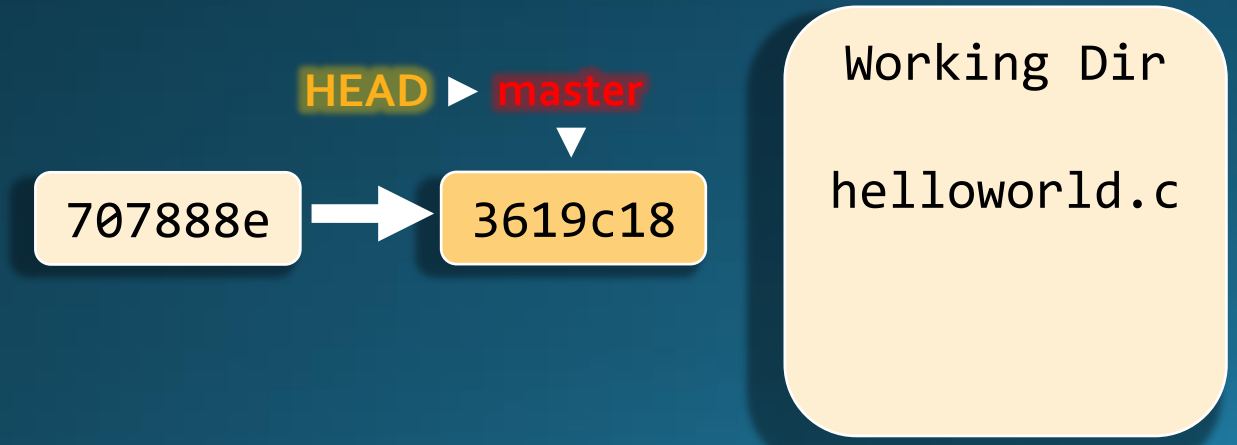
Situation: You have edited your files and saved the changes, but you haven't committed them. Now you change your mind and no longer want to keep the changes.

Use: `git checkout -- <filename>`

Result: Git alters the files in the working directory to match the last commit.

Checkout

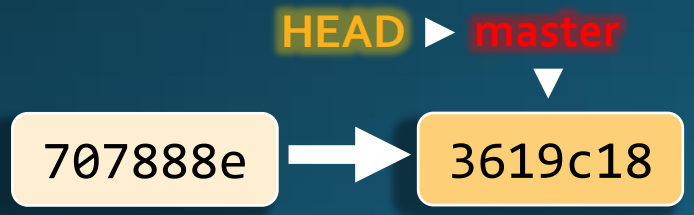
```
> vim helloworld.c
> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   helloworld.c

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



Checkout

```
> vim helloworld.c
> git status
On branch master
Changes not staged for
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   helloworld.c

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

git checkout <branch>

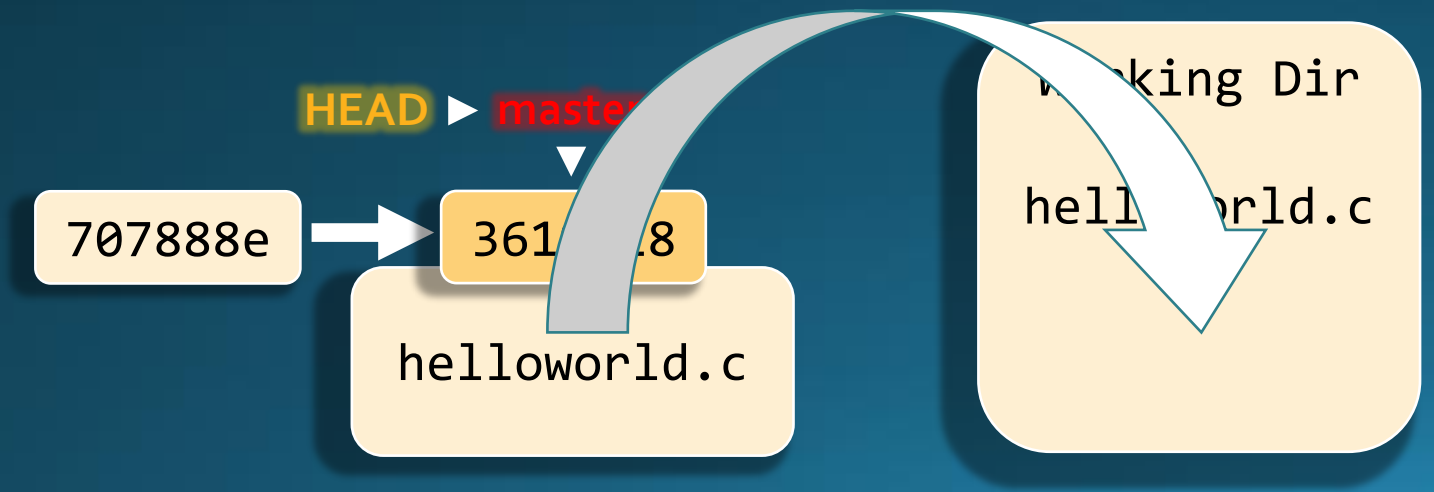


Working Dir

helloworld.c

Checkout

```
> vim helloworld.c
> git checkout -- helloworld.c
```



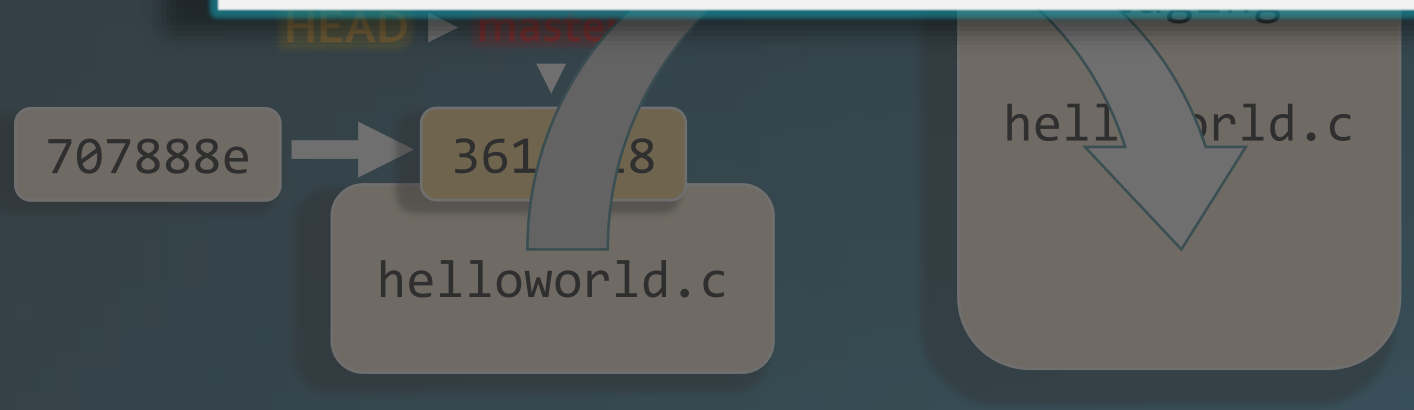
Checkout

```
> vim helloworld.c
> git checkout -- helloworld.c
```

IMPORTANT

It’s important to understand that git checkout is a dangerous command. Any local changes you made are gone.

Git replaces the file with the most recently-committed version. Don’t ever use this command unless you absolutely know that you don’t want those unsaved local changes.



Undoing Things: Unstaging

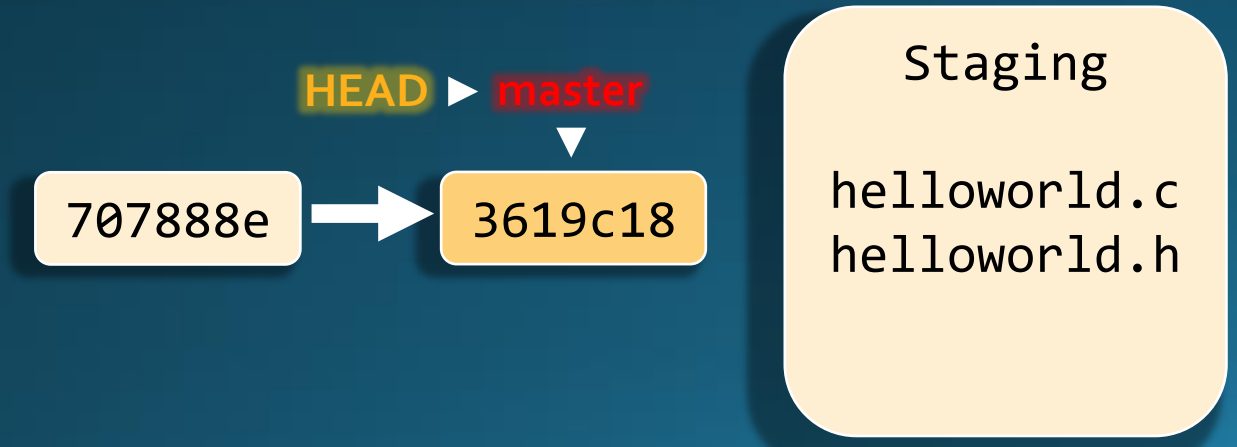
Situation: You change multiple files and want to commit them in two separate commits. You accidently used git 'add *' so you want to remove them from the current commit but keep the changes.

Use: git reset HEAD <file>

Result: Unstage the file(s) to the current commit.

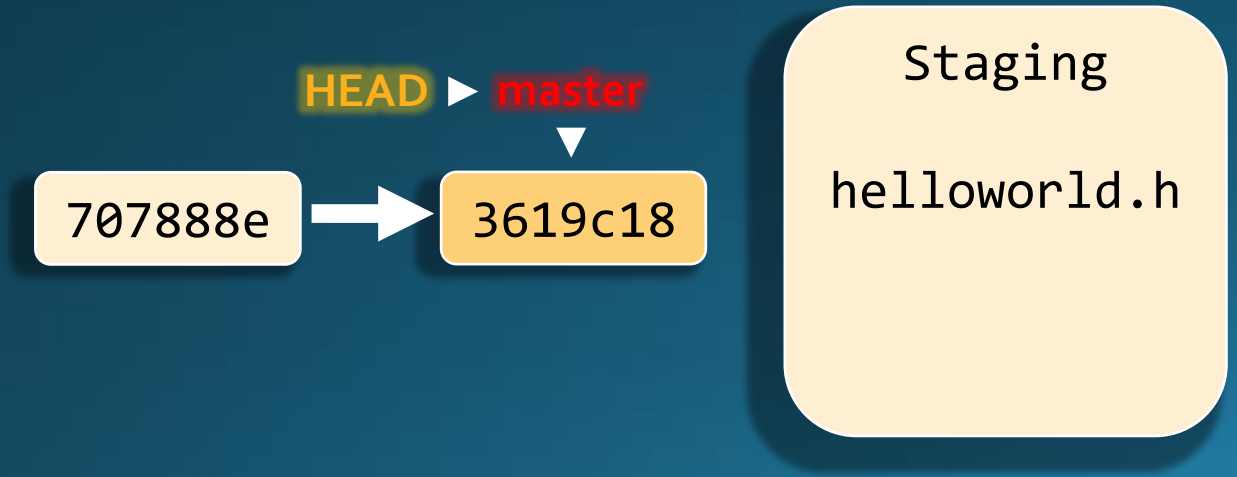
Reset

```
> vim helloworld.c helloworld.h
> git add *
> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   helloworld.c
        modified:   helloworld.h
```



Reset

```
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   helloworld.c
        modified:   helloworld.h

> git reset HEAD helloworld.c
Unstaged changes after reset:
M       helloworld.c
```



Reset

```
> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   helloworld.h

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   helloworld.c
```

Undoing Things: Amend

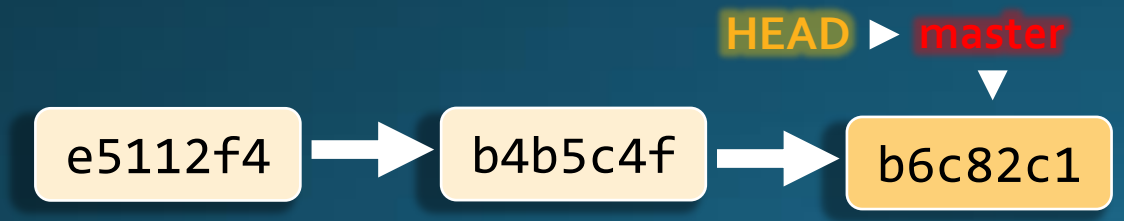
Situation: You have committed your changes too early and want to add to your commit.

Use: `git commit --amend`

Result: You replace the previous commit with the second commit.

Amend

```
> vim helloworld.c
> git add helloworld.c
> git commit -m"Modified .c file"
> git log --oneline
b6c82c1 (HEAD -> master) modified helloworld.c
b4b5c4f (origin/master, origin/HEAD) added hello world header
e5112f4 Added hello world source
```



Amend

```
> vim helloworld.c
> git add helloworld.c
> git commit --amend -m"re-modified .c file"
> git log --oneline
e5d7qdf (HEAD -> master) re-modified .c file
b4b5c4f (origin/master, origin/HEAD) added hello world header
e5112f4 Added hello world source
```