Part 3

Branch and merge

Concept: branch

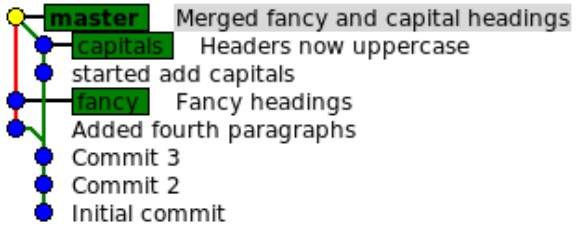
Separate, safe place to work

Free to experiment

Branches are cheap

- Just a pointer to a commt
- Cheap to create
- Cheap to switch between

Merge changes back in when you're ready



All repos come with a `master` branch

Workflow: feature branch

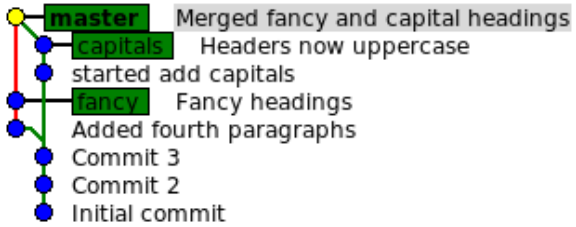
`master` is always working

- Mature state only

Generally, **one branch per feature**

When feature done

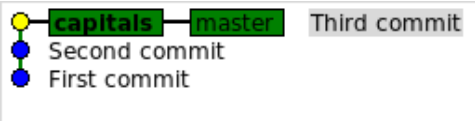
1. Merge into `master`
2. Delete the feature branch



Capitals branch

Let's work on the "feature" of changing headers to capital letters

`$ git branch capitals` : creates a new branch



- `$ git status`
- `$ git branch`
- refresh `gitk`
- `$ git checkout capitals`
- `$ git status`
- `$ git branch`

In **just one** text file

1. change title to uppercase
2. change "commit 1" line to uppercase

- `$ git status`
- `$ git commit -am "Started adding capitals"`
- `$ git status`
- refresh `gitk`

Branches isolate change

- `$ git status`

Look at your text file

- `$ git checkout master`
- `$ git status`

Look at the file again

- `$ git checkout capitals`
- `$ git status`

Look at the file again

(`git gui` also switches branches)

Git keeps you safe

In capitals

1. Make all-caps all the titles and "commit" headers in both files
2. Save the file, don't `git add` or `git commit`
3. `$ git checkout master`

```
$ git commit -am 'Headers now uppercase'
```

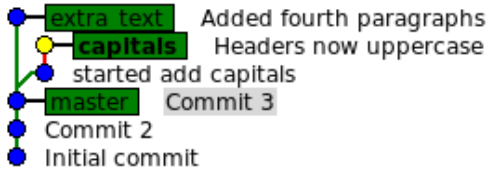
Many branches!

In master

- `$ git checkout master`
- Refresh `gitk`
- `gitk` > File > List references

Want to add extra paragraphs to both text files

- We'll create a new branch to do that
- `$ git branch extra_text`
- `$ git checkout extra_text`



Add a new paragraph to each text file, `add` and `commit`

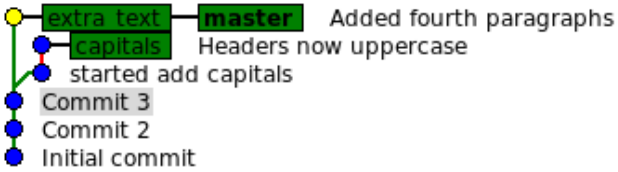
Refresh `gitk`

Merging (the very easy part)

Bring the changes from *another* branch into *this* branch

To incorporate `extra_text` branch into `master`

1. `$ git checkout master`
2. `$ git merge extra_text`



Fast-forward merge: just slide the `master` branch pointer along

Don't need the `extra_text` branch any more

- `$ git branch -d extra_text`
- `$ git branch`
- Refresh `gitk`

Merging (the not-so-easy part)

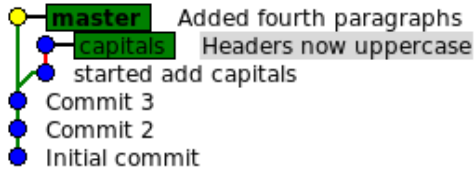
To incorporate `capitals` branch into `master`

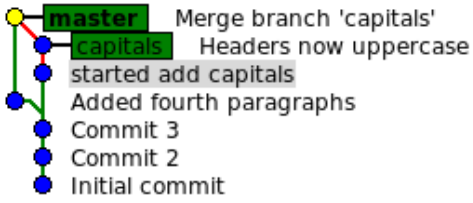
Can't fast forward

Create a *new* **merge** commit

1. `$ git checkout master`
2. `$ git merge capitals -m "Merging capitals as headers"`

Refresh `gitk`





Sensible merging

Look at one of the text files

```
# COMMIT 3

I try in vain to be persuaded that the pole ...

# Commit 4

Its productions and features may be without example...
```

More log fun

Refresh the view in `gitk`

```
$ git log --graph --oneline
```

```
$ git log
```

```
commit 94f64e551debca70bfee7a3e4e1331e936c5f627 (HEAD -> master)
Merge: 0d8d3b5 c65cfd3
```

`master` and `capitals`

Commit with two parents

- `HEAD^1` is first parent ("mother")
- `HEAD^2` is second parent ("father")
- `HEAD^2^1` is second parent's first parent ("father's mother")
- `HEAD~` always tracks first parent
- `HEAD^2^1^1` == `HEAD^2~~` == `HEAD^2~2`

Seeing changes

```
$ git diff HEAD^1
```

```
$ git diff HEAD^2
```

Also in `gitk`

Making life difficult

First, undo this merge

- `$ git reset HEAD^1 --hard`

Create a branch `fancy`

- `$ git branch fancy`
- `$ git checkout fancy`

In `fancy`, in **one file**, use fancy words for the headings: `# Most fancy commit line 1`

Add and commit the changes

- `$ git commit -am "Fancy headings"`

Merge `fancy` into `master`

- `$ git checkout master`
- `$ git merge fancy`

Don't delete `fancy` yet

Merging (the hard part)

When we now merge `capitals` into `master`, what should Git do?

```
$ git checkout master
```

```
$ git merge capitals -m "Merging capitals as headers"
```

```
Auto-merging frankenstein.txt
CONFLICT (content): Merge conflict in frankenstein.txt
Auto-merging carmilla.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Look in the file

```
# FRANKENSTEIN, by Mary Shelly

<<<<<<< HEAD
# Most fancy commit line 1
=======
# COMMIT 1
>>>>>>> capitals
```

Resolving merge conflicts

Option 1

```
$ git merge --abort
```

Chocolate

Resolving merge conflicts

```
# FRANKENSTEIN, by Mary Shelly

<<<<<<< HEAD
# Most fancy commit line 1
=======
# COMMIT 1
>>>>>>> capitals
```

Human intervention needed to solve these problems

```
# Most fancy COMMIT line 1
```

- Remove the conflict markers as well
- `add` files when all resolved
- `$ git commit -m "Merging capitals as headers"` when done

Look into tools like *KDiff3* and *Meld* for GUIs

End of part 3

- Branches
- Merge
- Merge conflict

Take a break, get some coffee