Git 1

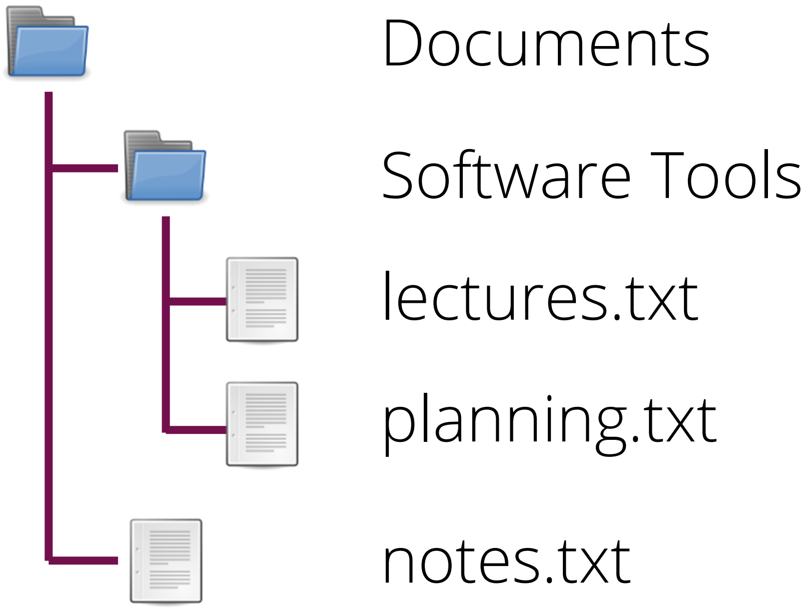
COMS10012 Software Tools

Git

File system

Organises files, but

- versioning?
- working together?



Documents

Software Tools

lectures.txt

planning.txt

notes.txt

Git

working copy

repository

complete history
of your project

Installation

alpine: **`sudo apk add git`** to install.

Syntax: **`git COMMAND [OPTIONS]`**

`git help COMMAND` shows information, but on alpine you need to install **`git-doc`** first.

Setup

In git, commits are tagged with a name and email address. Set these up with e.g.

```
git config --global user.name "David"

git config --global user.email "-"
```

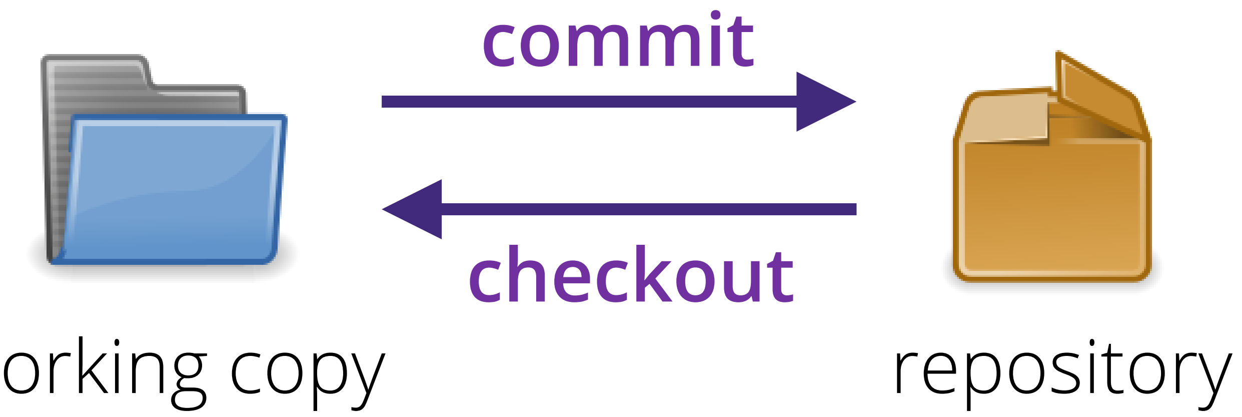
(These are for your projects – git itself does not need an "account" and does not send you mail.)

Initialise a repository

```
alpine310:~/tutorial$ git init
Initialized empty Git repository in
/home/vagrant/tutorial/.git/
alpine310:~/tutorial$ ls -a
.     ..     .git
```

Git

Basic git workflow: code, commit, repeat.



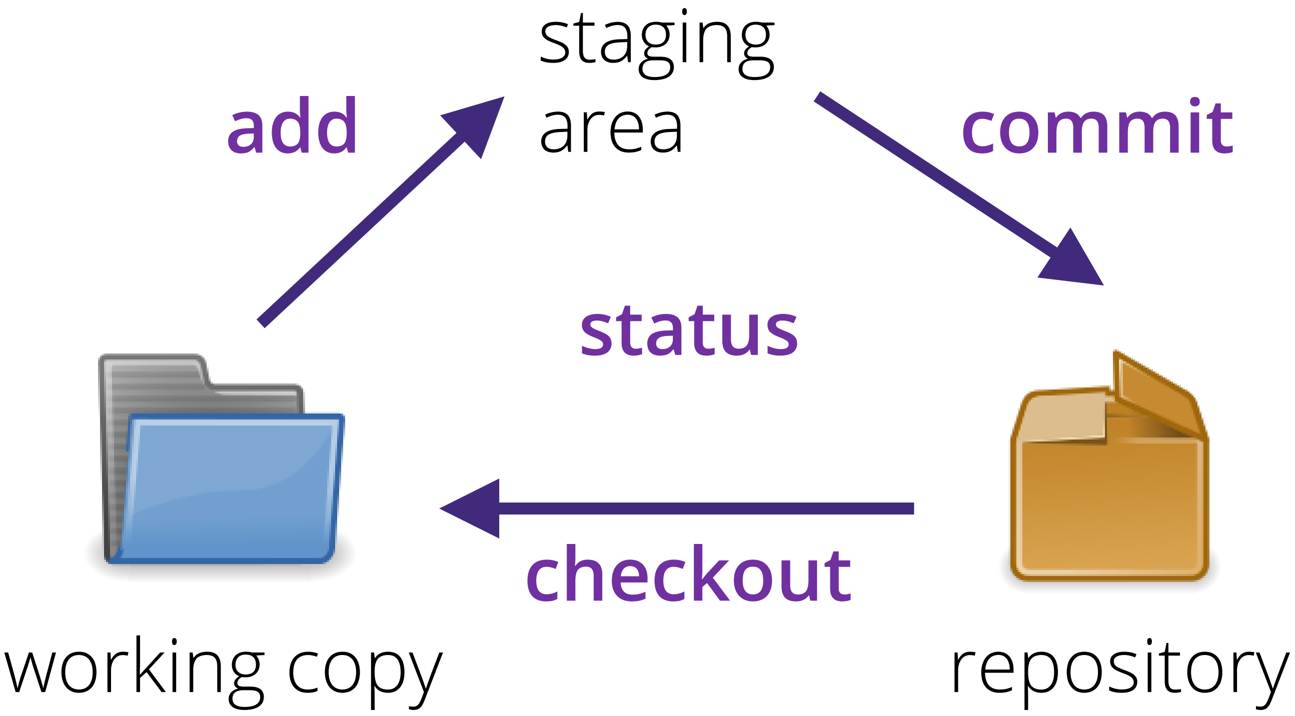
commit

checkout

working copy

repository

Actually …


staging
area
add
commit
status
checkout
working copy
repository

git status

```
alpine310:~/tutorial$ git status
On branch master
No commits yet
nothing to commit (create/copy files and use
"git add" to track)
```

create a file

Create a file, mine is called main.c :

```
alpine310:~/tutorial$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what
   will be committed)
        main.c
nothing added to commit but untracked files
present (use "git add" to track)
```

add / stage the file

```
alpine310:~/tutorial$ git add main.c
alpine310:~/tutorial$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.c
```

commit

```
$ git commit -m "Added a file"

[master (root-commit) 4a3add1] Added a file
 1 file changed, 0 insertions(+),
 0 deletions(-)
 create mode 100644 main.c
```

commits

Every commit needs a message.

If you leave off the –m option, git opens an editor for you … by default, vim. Use ":q" then enter to get out again.

commit (message) style

generally:

- lots of small commits, not a few big ones
- descriptive commit messages
- short first line (max 60-80 characters)

commit messages

```
$ git commit -m "Fix graphics bug

Fix bug #320 where graphics flicker on a
high-DPI screen on mac."
```

alternative:

```
$ git commit -m "first line" -m "longer text"
```

another edit

Update main.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:   main.c
```

another commit

```
$ git add .
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   main.c

$ git commit -m "main function"
[master 69f83ec] main function
 1 file changed, 3 insertions(+)
```

log: commit history

```
$ git log
commit 69f83ec5... (HEAD -> master)
Author: David <->
Date:   Fri Jul 24 09:17:22 2020 +0000

    main function

commit 4a3add1a61b693228b126c3a758f20b0007a39df
Author: David <->
Date:   Fri Jul 24 08:52:24 2020 +0000

    Added a file
```

checkout: revert to a commit

revert edits to a file:

```
$ git checkout -- FILE
```

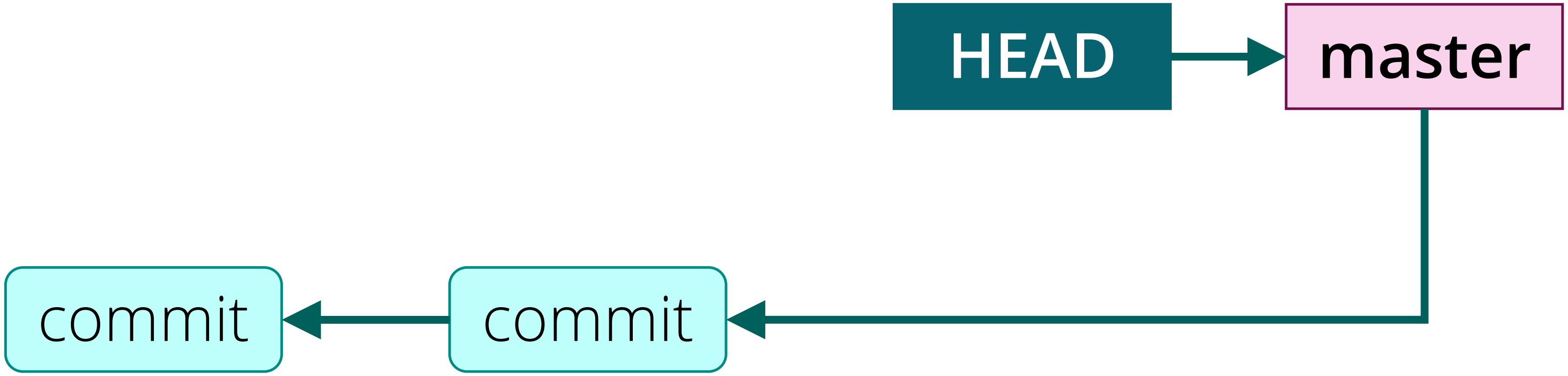
revert working directory to a commit:

```
$ git checkout ID
You are in 'detached HEAD' state ...
```

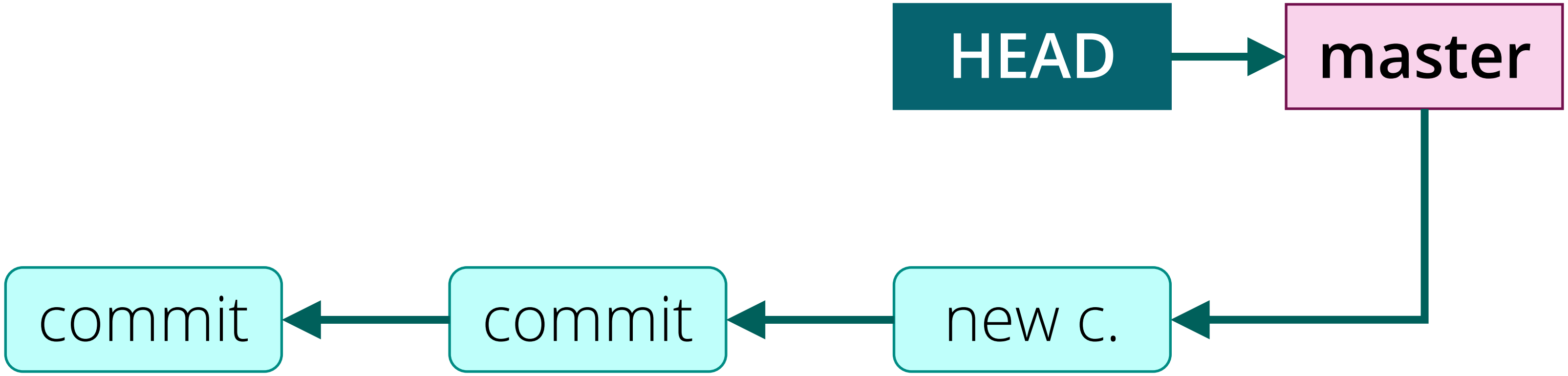
either **`git checkout master`** to go back to latest, or:

```
git stash; git checkout master; git stash pop
```

commit graph



commit graph


HEAD

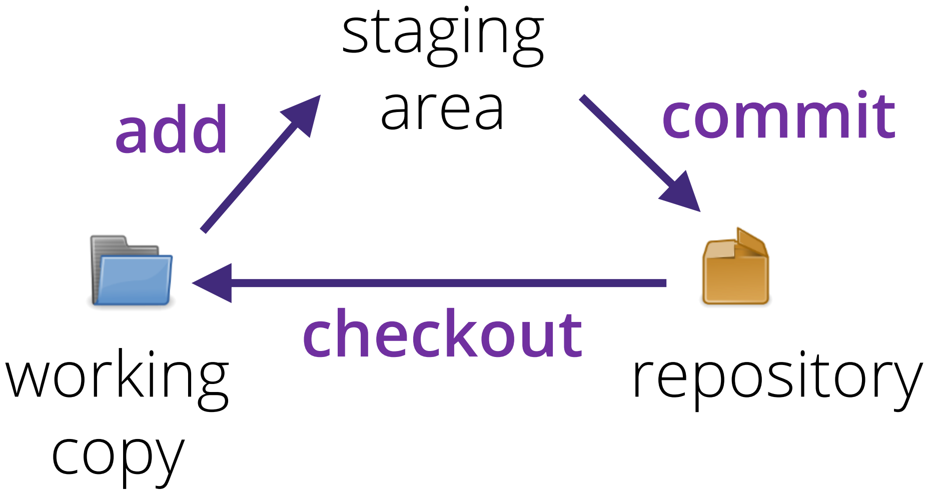
master

commit

commit

new c.

diagram


staging area

add

commit

working copy

checkout

repository

init

status

log