Collaboration tools

Lecture 9 - Programming for Data Science
Dr. Alan M. Lewis

Programming Module Outline

- Reading, understanding and manipulating data (Kevin)
- Modelling and visualising data (Kevin)
- Functions and Flow
- **Tools for programming**
- High Performance Computing

Today's Lecture

1. Google Drive
2. Introducing git
3. Bash (briefly)
4. Using git locally
5. Using git remotely

Google Drive

The simplest collaboration tools are in Google Suite:

- Docs
- Slides
- Colab

Google Drive

The simplest collaboration tools are in Google Suite:

- Docs
- Slides
- Colab

Good for: Reports, Presentations

Worse for: Code, handling many files.

Git outline

Git is a version control system.

Basic Features:

- Creates a “repository” (repo)
- Tracks changes in repository
- Takes snapshots (commits) of the repo which can be recovered

Git outline

Git is a version control system.

Basic Features:

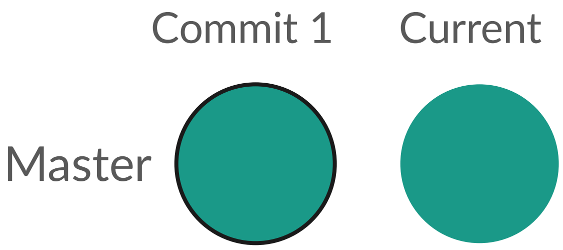
- Creates a “repository” (repo)
- Tracks changes in repository
- Takes snapshots (commits) of the repo which can be recovered

Advanced Features:

- Allows separate “branches”, with changes kept separate.
- Allows collaboration, uploading and downloading changes.
- Merges independent changes efficiently.

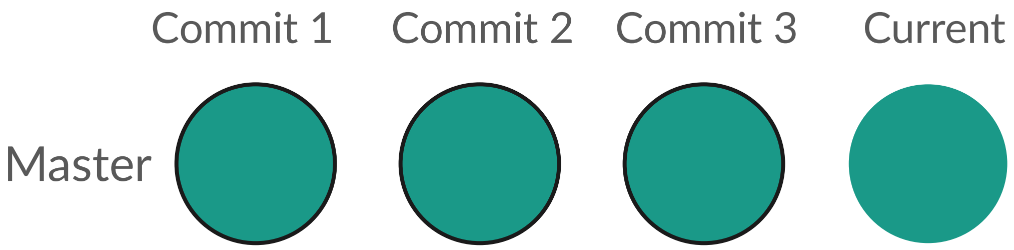
Git in pictures

Git is a version control system.



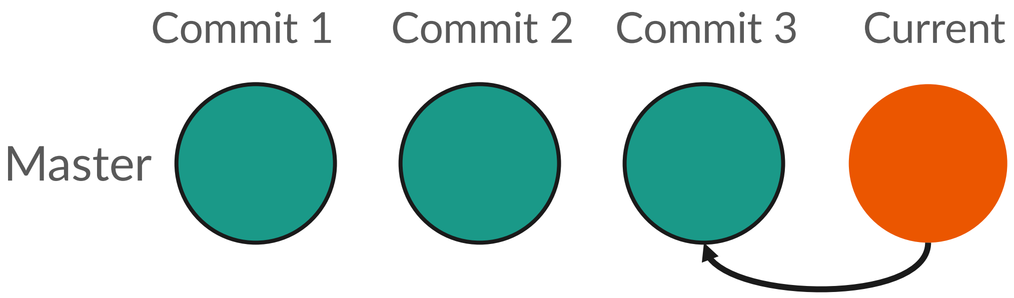
Git in pictures

Git is a version control system.



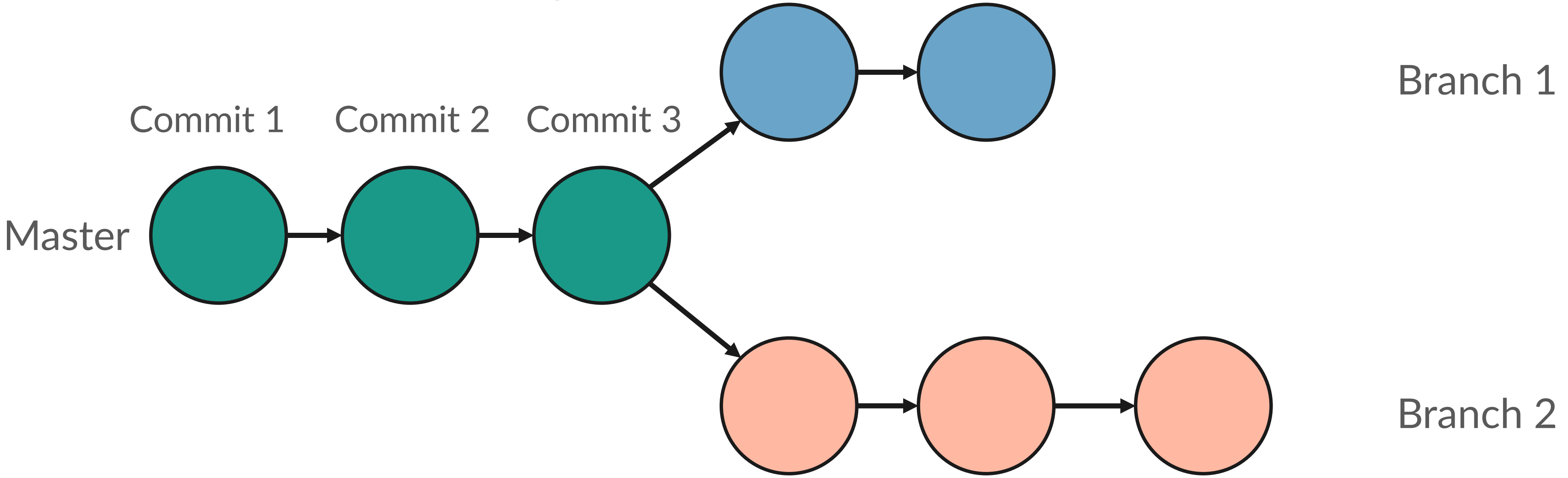
Git in pictures

Git is a version control system.



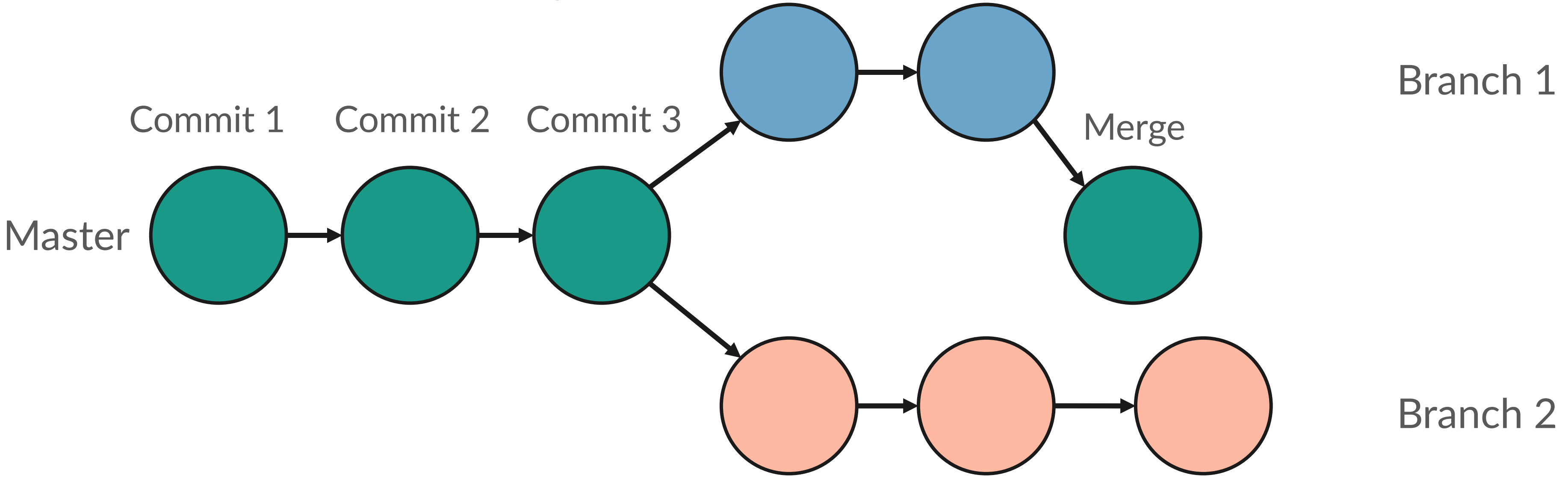
Git in pictures

Git is a version control system.



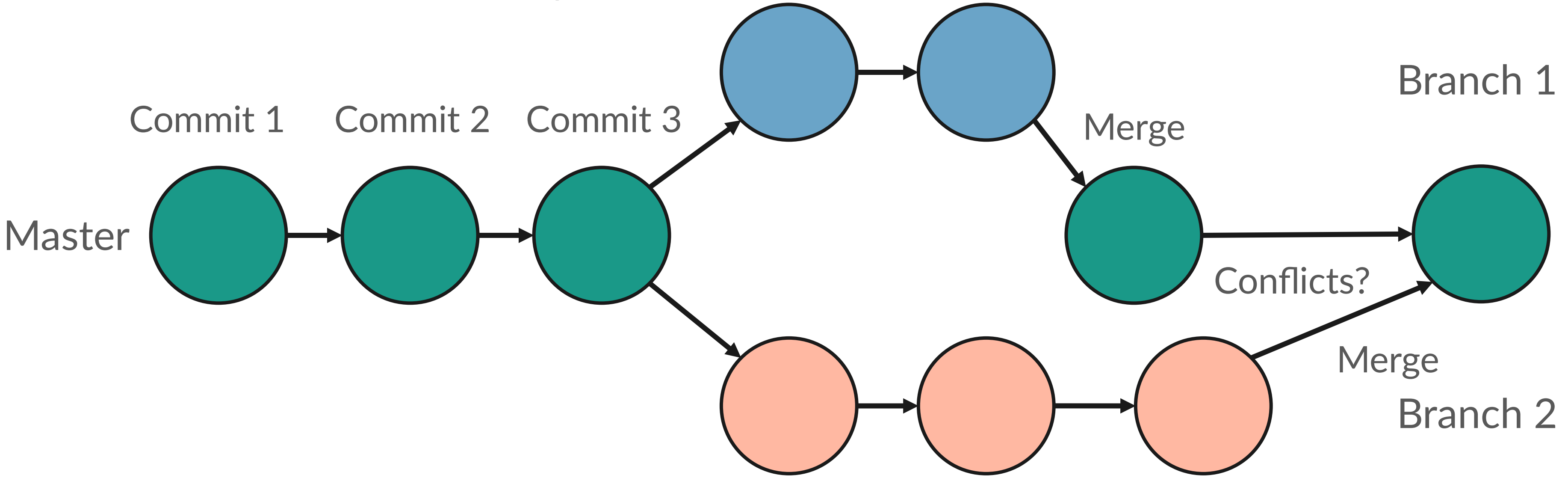
Git in pictures

Git is a version control system.



Git in pictures

Git is a version control system.



Bash (briefly)

Bash is a “shell”. It interprets text instructions and runs programs.

Bash (briefly)

Bash is a “shell”. It interprets text instructions and runs programs.

It comes with a collection of tiny programs for manipulating files.

```
# Change Directory
cd Desktop

# List files in directory
ls

# Move, copy and delete
files (add -r for folders)
mv file1 Folder/file1
cp file1 Folder/file1
rm file1
```

Bash (briefly)

Bash is a "shell". It interprets text instructions and runs programs.

It comes with a collection of tiny programs for manipulating files.

We need to know this because git is controlled by the bash shell.

```
# Change Directory
cd Desktop

# List files in directory
ls

# Move, copy and delete
files (add -r for folders)
mv file1 Folder/file1
cp file1 Folder/file1
rm file1
```

Bash (briefly)

Bash is a “shell”. It interprets text instructions and runs programs.

It comes with a collection of tiny programs for manipulating files.

We need to know this because git is controlled by the bash shell.

We will talk about bash more when we talk about HPC.

```
# Change Directory
cd Desktop

# List files in directory
ls

# Move, copy and delete
files (add -r for folders)
mv file1 Folder/file1
cp file1 Folder/file1
rm file1
```

Bash (briefly)

Paths can be relative (to where we are now) or absolute.

There are some useful shortcuts when using paths.

```
# Relative Paths

cd ~ # Home directory
cd - # Previous directory
cd . # This directory
cd .. # One directory up
cd / # Root directory
```

Bash (briefly)

Paths can be relative (to where we are now) or absolute.

There are some useful shortcuts when using paths.

```
# Relative Paths

cd ~ # Home directory
cd - # Previous directory
cd . # This directory
cd .. # One directory up
cd / # Root directory

# Wildcards
ls * # List everything
ls *.py # list everything
      ending in .py
```

Bash (briefly)

We can run python scripts from a bash terminal.

We need to be careful about where we put files referred to in the script.

```
python script.py
```

Using Git locally - getting started

Create a repository	`git init -b main`
Add files to track	`git add filename`
Show which files are tracked/ changed/untracked	`git status`
Save a snapshot	`git commit -a`

Aside: vi text editor

Git commit messages are created using a text editor which runs in the terminal called vi. Key things to know are:

Press “Insert” or “i” to insert text.
Press “Esc” to stop inserting text.
Type “:w” to save.
Type “:x” to save and exit.
Type “:q!” to exit without saving.

Using Git locally - making progress

Show differences since last commit	`git diff`
Remove file from repo (this deletes the file!)	`git rm`
List available branches	`git branch`
Switch to existing branch	`git checkout branch-name`
Create new branch	`git checkout -b branch-name`

Using Git locally - when things go wrong

Go back to last commit	`git stash`
Show old commits	`git log`
Go back to a specific commit (Dangerous, use with care. You will lose data.)	`git reset --hard/soft/mixed commit-id (commit-id found in git log)`
Undo a specific commit	`git revert commit-id`

Using Git locally - merging branches

Merge another branch into the current branch	`git merge branchname`
Check if there are unresolved conflicts	`git diff --name-only --diff-filter=U`
Remember to commit after resolving conflicts!	`git commit -a`
Delete branch	`git branch -d branchname`

Using Git locally - merging branches

Sometimes merges cannot be done automatically due to conflicts.

Conflicts must be resolved manually.

Once conflicts have been resolved, delete the lines containing >>, << and ==.

```
<<<<<<< HEAD

Version of code from this branch

=======

Version of code from the other branch being merged in.

>>>>>>> merge_test
```

Using Git remotely

Set up remote location (origin)	`git remote add origin https://github.com/???/???`
Connect local and remote branch	`git push -u origin branchname`
Update remote branch after that	`git push`
Update local branch from remote	`git pull`

Using Git remotely - using an existing repo

Create a local copy of the repository.

```
git clone
https://github.com/???/???
```

Create a new branch of the remote repository, if allowed.

```
git push -u origin branchname
git push
```

In bigger projects, you can't push directly into the main branch. Instead, you need to open a "pull request".

Today's Lecture

1. Google Drive
2. Introducing git
3. Bash (briefly)
4. Using git locally
5. Using git remotely