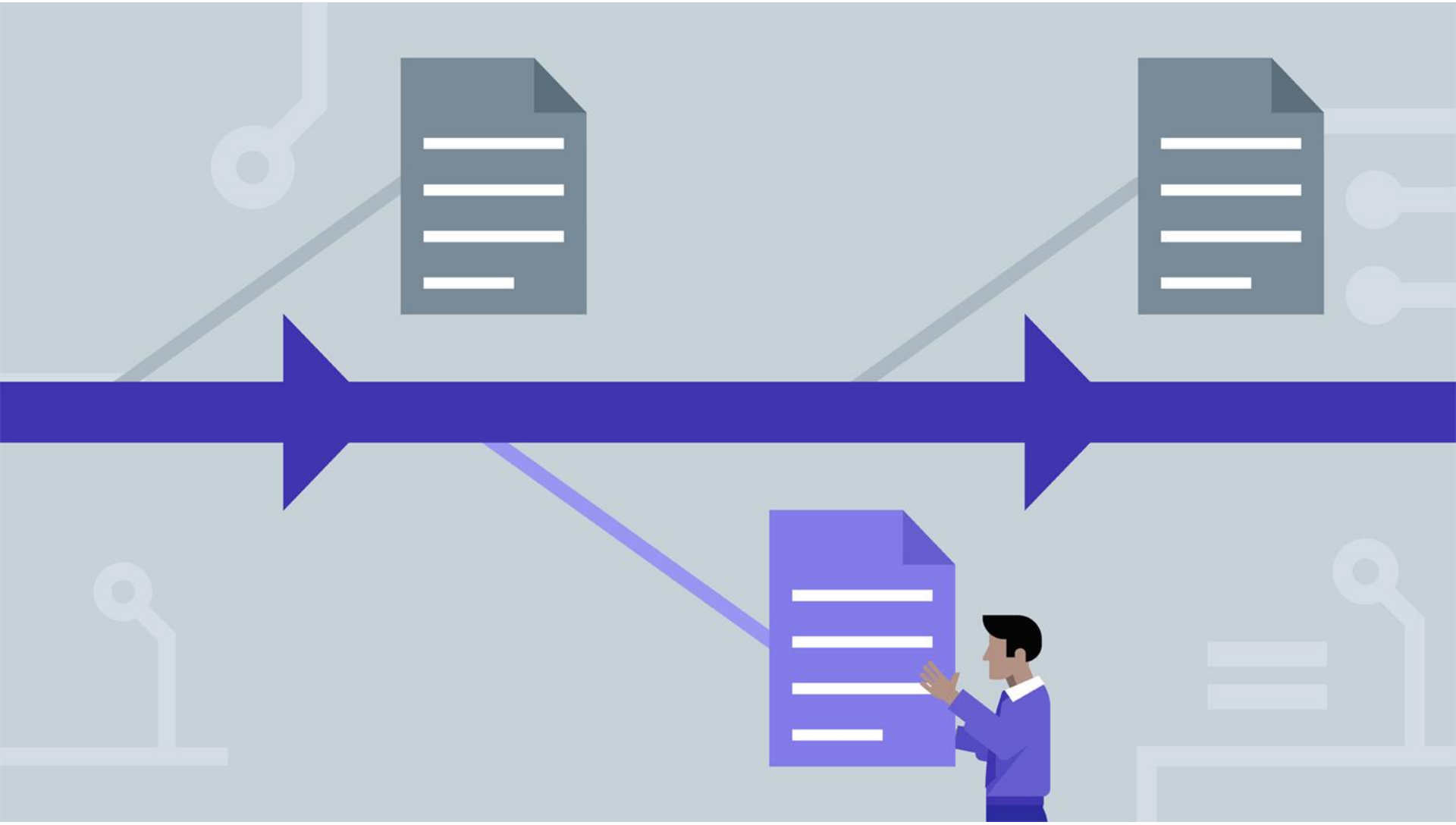
# Git Rekt

## Collaborating on Code

# Version Control



# Version Control

- Deal with software changes
- Track who did what
- Find a particular change (version)
- Manage multiple releases

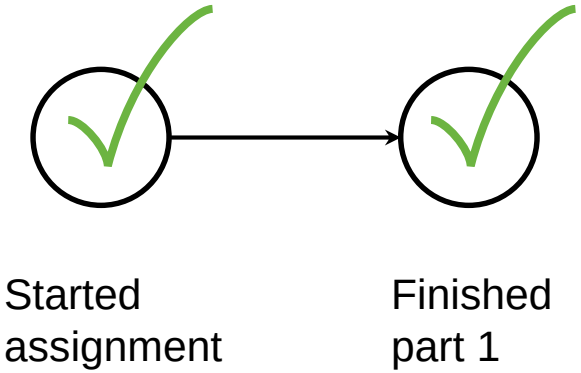
# What is Git?

Git is a *distributed* version control system

Stores changes to files over time

Tracking complex changes across many files

# Changes are Discrete

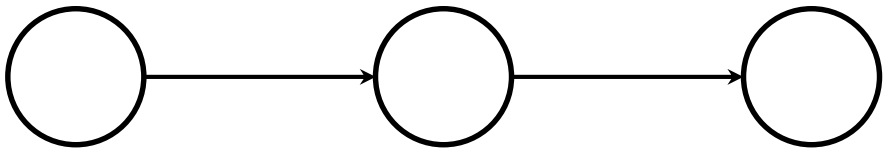


# Change Control

Someone else changed something.
What did they do?

Everything's suddenly broken.
What changed?

# Commit



# Staging: ”add”

Sometimes we want to choose which changes we actually want to commit

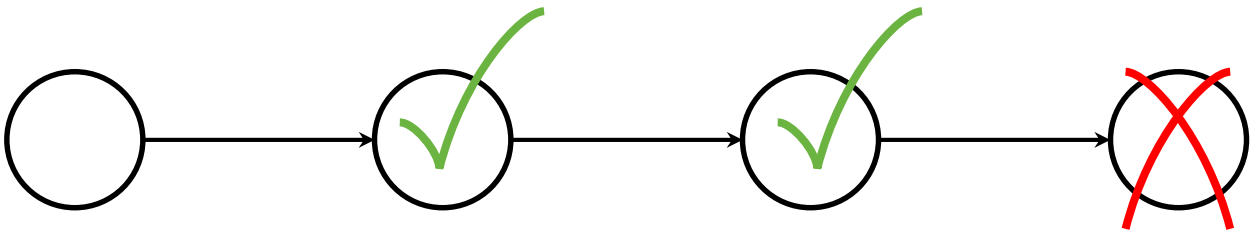
Before we can commit, we need to “add” the files we’re interested in to the “staging area”

# Reversion

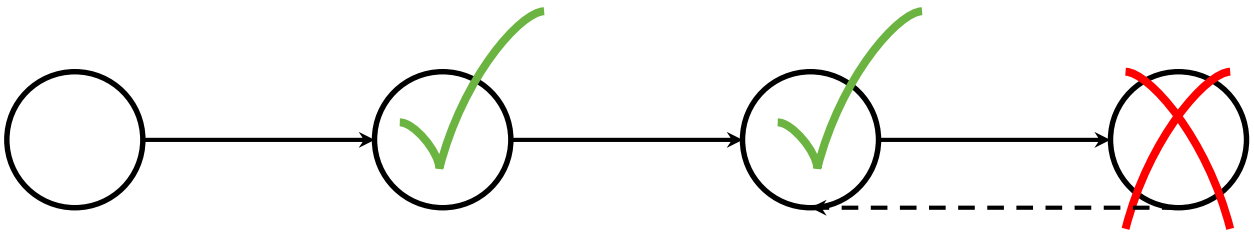
What if the change I’m about to make turns out not be any good?

How can I undo those changes?

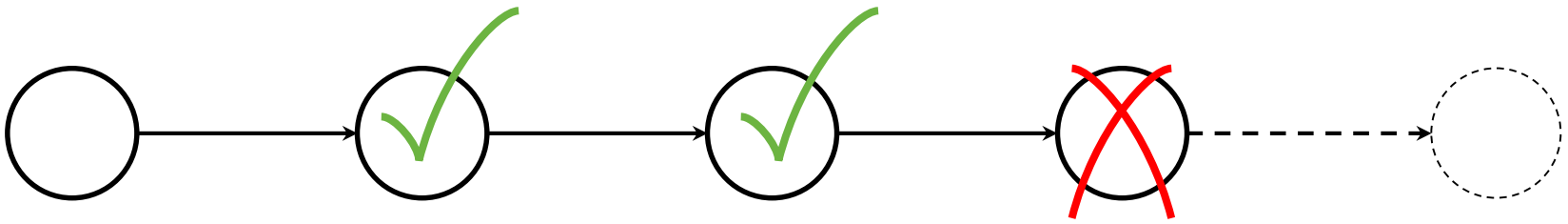
# Reversion



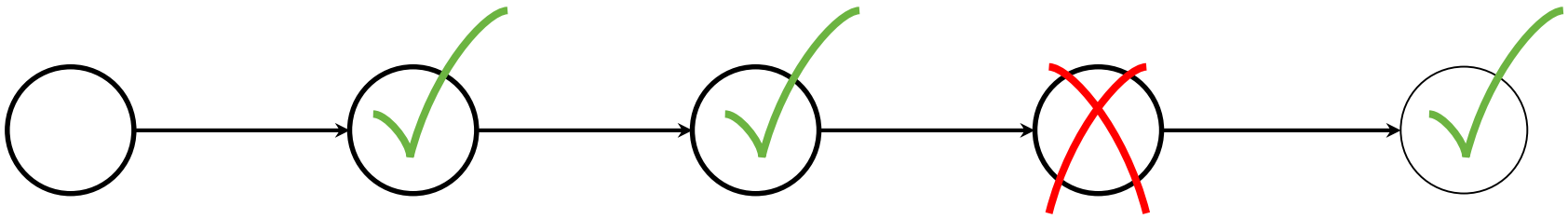
# Reversion



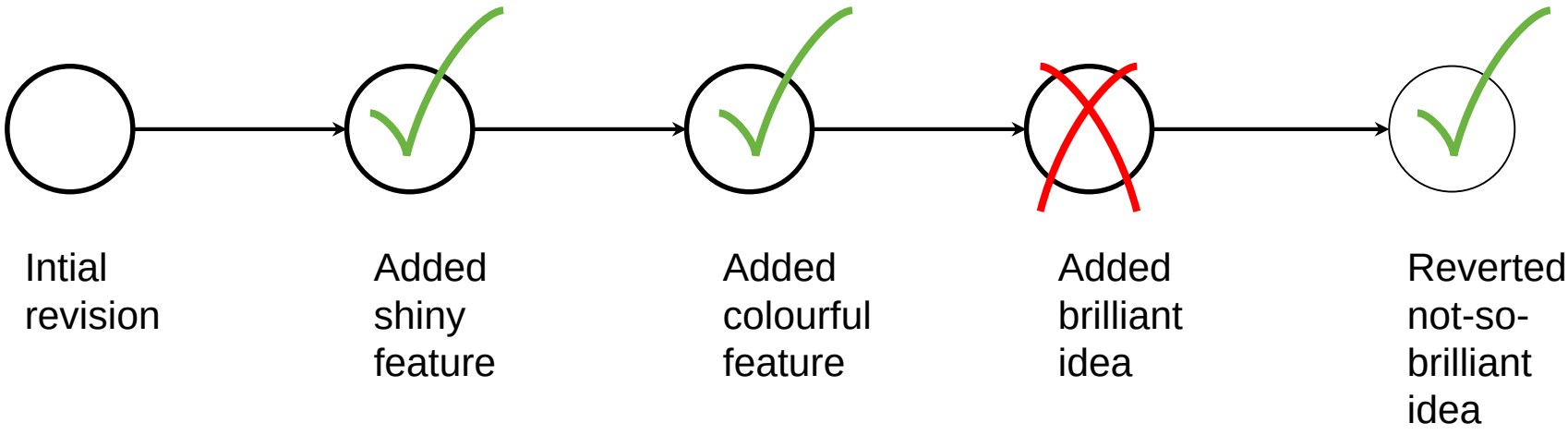
# Reversion



# Reversion

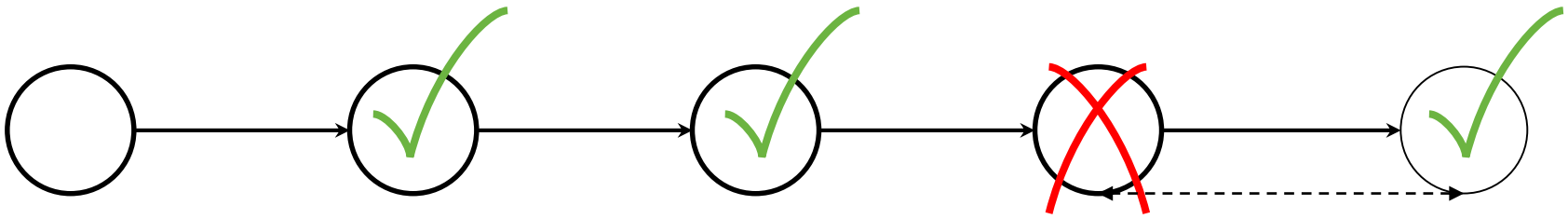


# History




# Change Log

# Diff



What is the difference
between these revisions?

# Diff

A diff between two versions of a program will contain line-by-line changes to many files

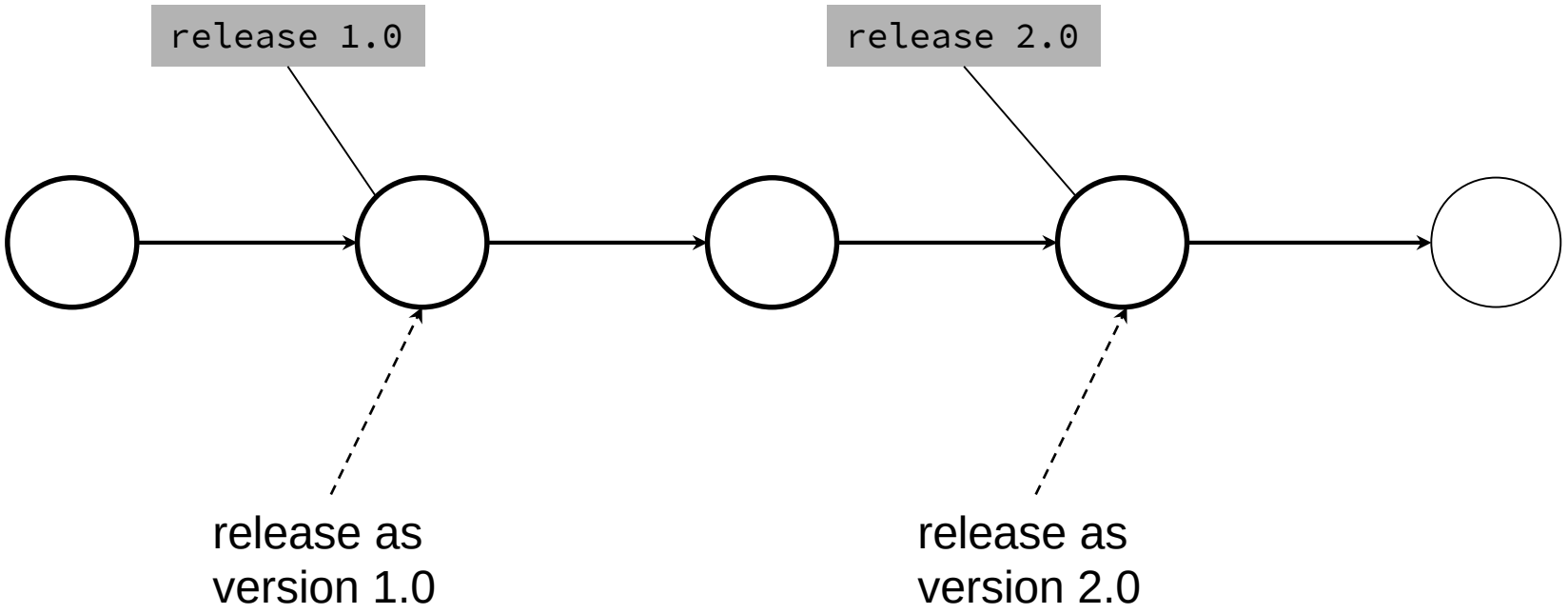
37 hellodeco/src/main/java/hellodeco/HelloDeco.java — View

```
...  ...  @@ -1,34 +1,37 @@
1    1    package hellodeco;
2    2
3    3    public class HelloDeco {
     4    +    /*
     5    +     * DO NOT MODIFY THIS CLASS
     6    +     */
4    7
5    8        public static void main(String[] args) {
6         -        /*
7         -         * DO NOT MODIFY THIS METHOD.
8         -         */
9         -
10   9            System.out.println("Studio 1 Collaborators:");
11        -        Studio1CollaboratorsList collaboratorsList1 = new Studio1CollaboratorsList();
12        -        collaboratorsList1.addCollaborators();
13        -        collaboratorsList1.printCollaborators();
     10   +        DECO2800CollaboratorsList studio1 = new Studio1Collaborators();
     11   +        studio1.addCollaborators();
     12   +        studio1.printCollaborators();
14   13           System.out.println();
15   14
16   15           System.out.println("Studio 2 Collaborators:");
17        -        Studio2CollaboratorsList collaboratorsList2 = new Studio2CollaboratorsList();
18        -        collaboratorsList2.addCollaborators();
19        -        collaboratorsList2.printCollaborators();
     16   +        DECO2800CollaboratorsList studio2 = new Studio2Collaborators();
     17   +        studio2.addCollaborators();
     18   +        studio2.printCollaborators();
20   19           System.out.println();
```

# Tagging

Which revision did I sell to Alistair, again?

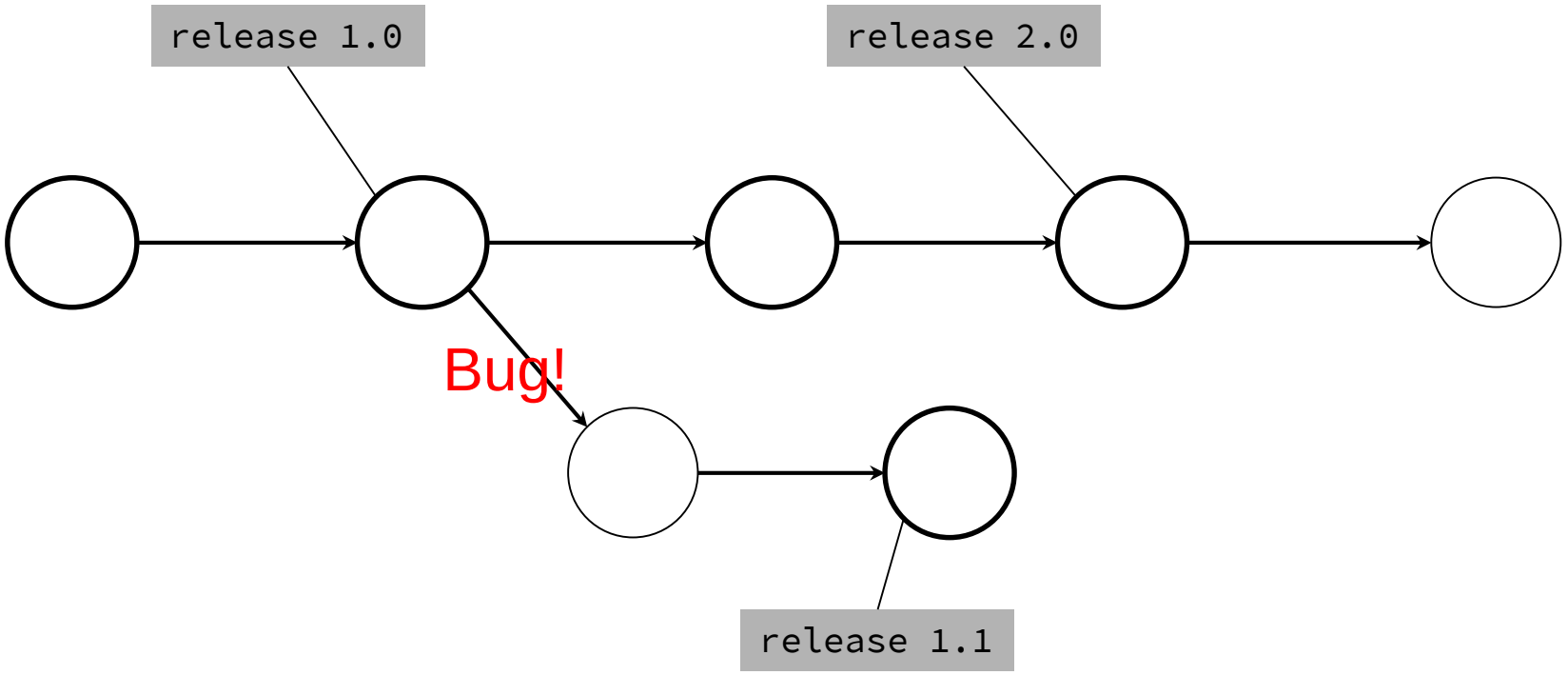
# Tagging


release 1.0
release 2.0
release as
version 1.0
release as
version 2.0

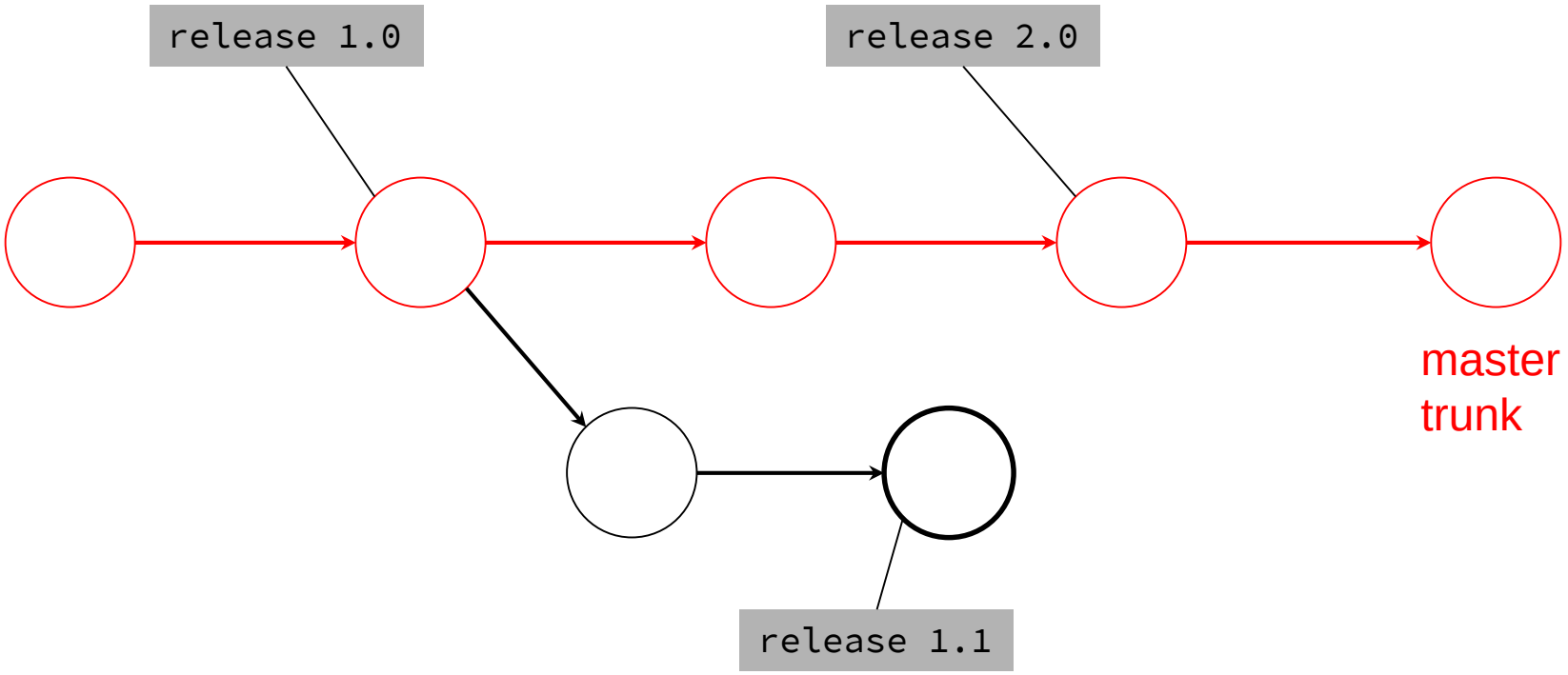
# Branching

I have to maintain the version I sold to Alistair as well as developing the new feature to sell to Barbara

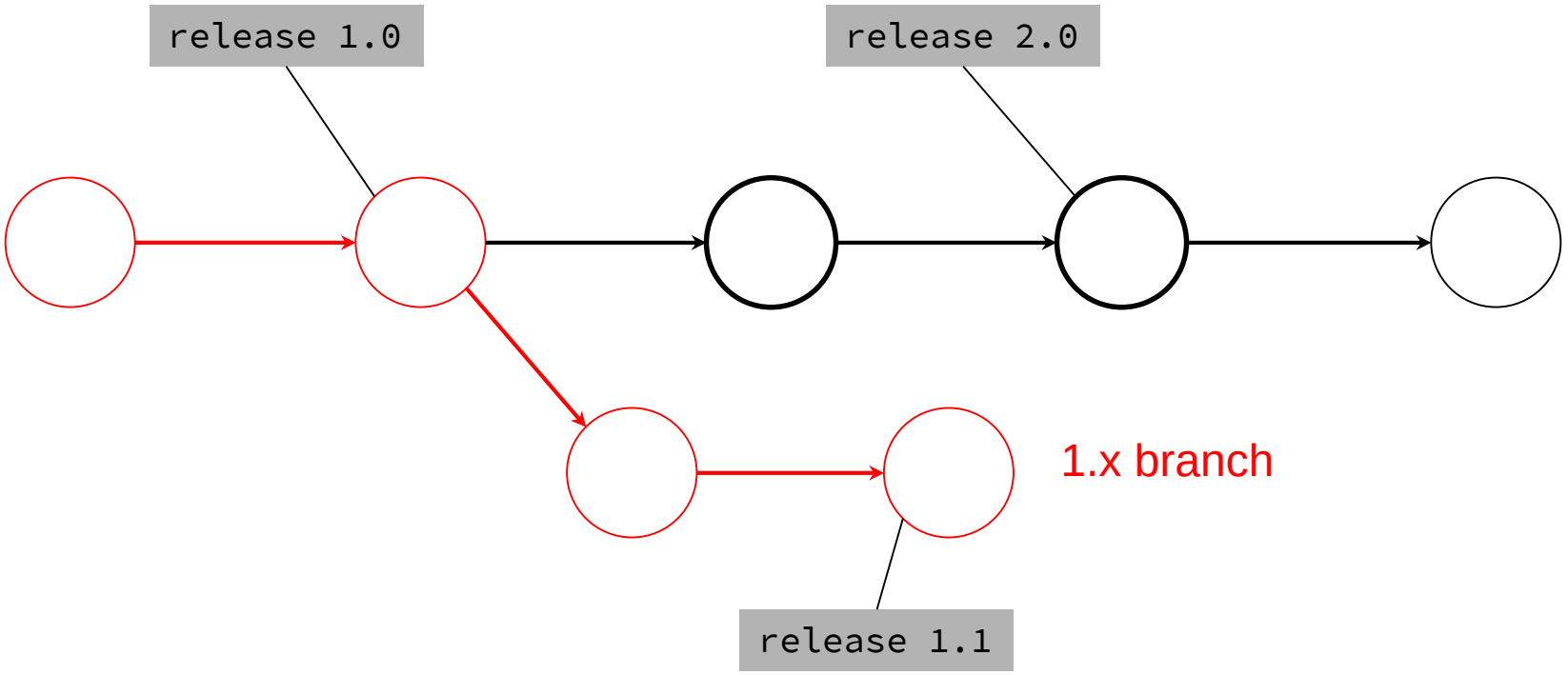
# Branching



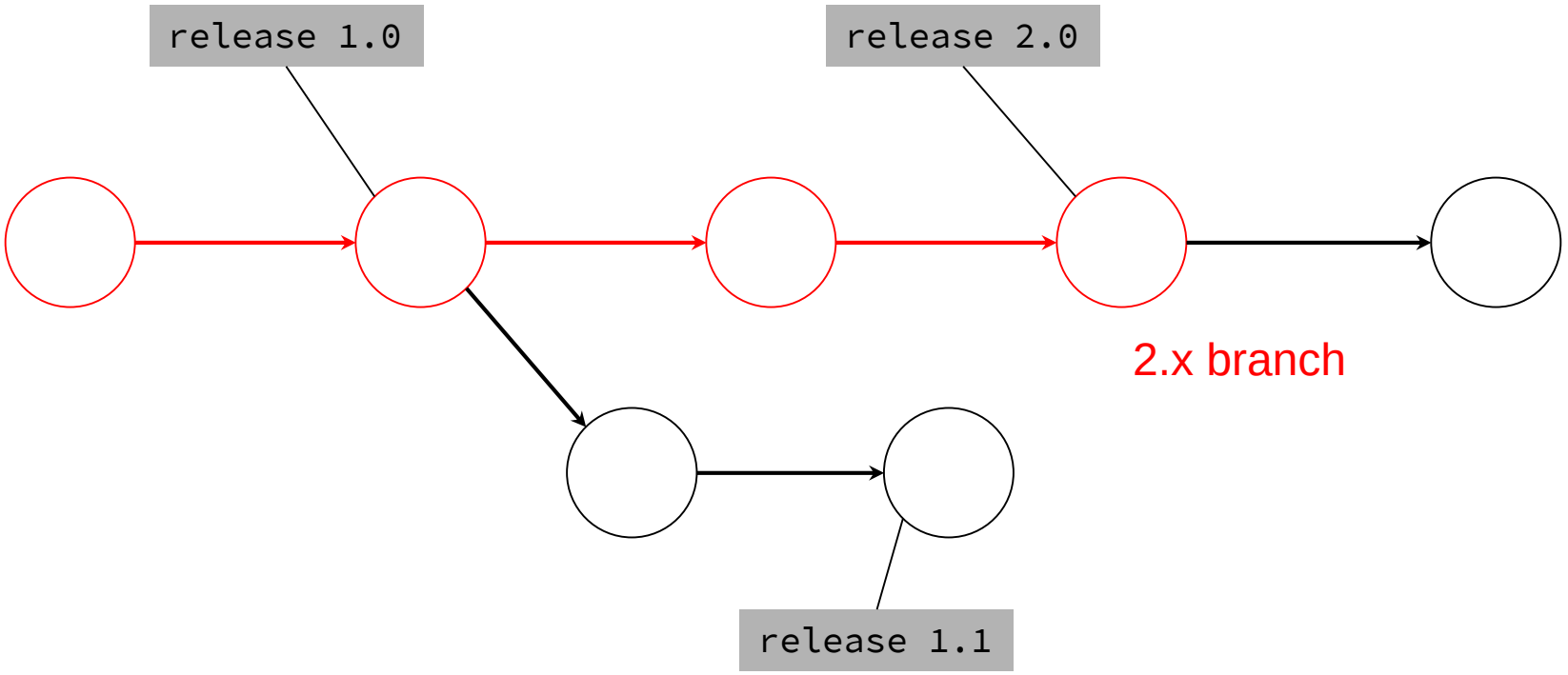
# Branching



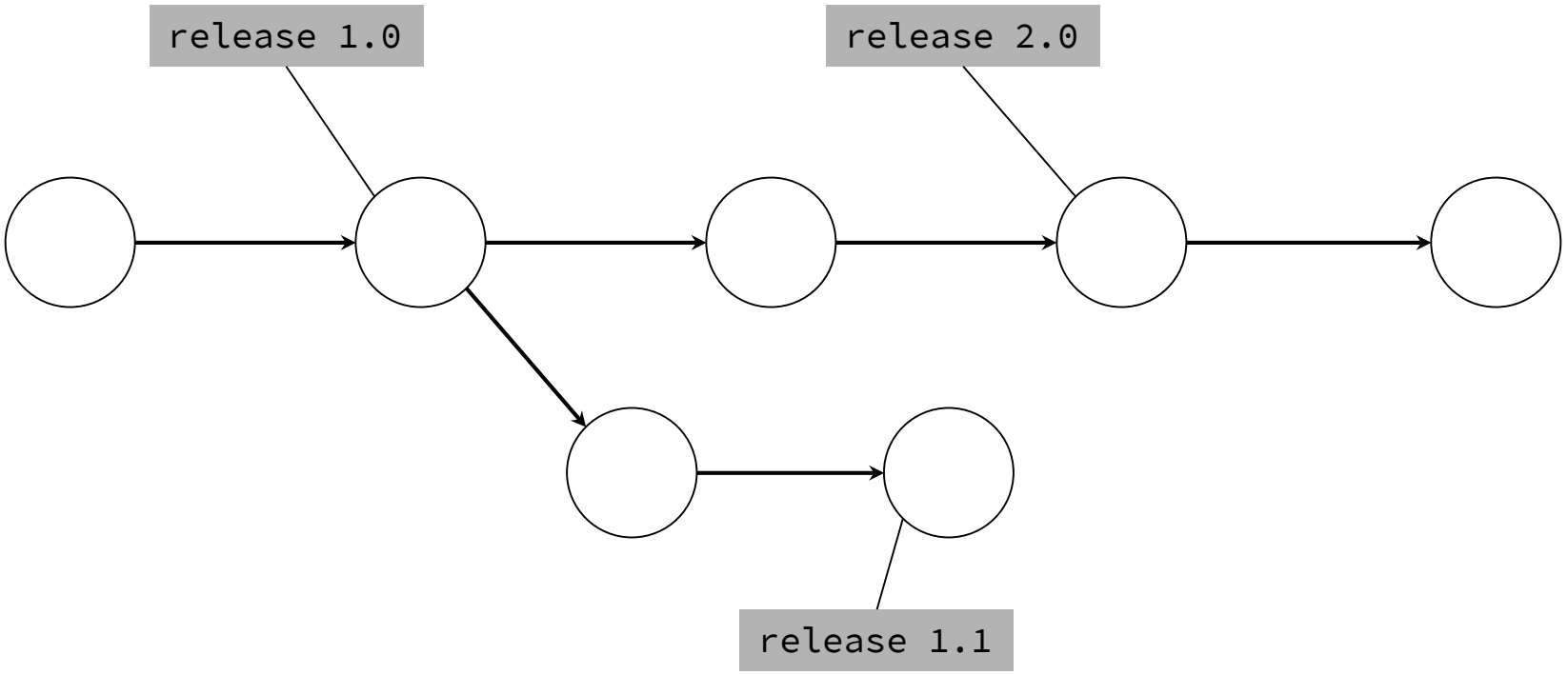
# Branching



# Branching



# Branching



# Git Commands for Branching

‘branch’ lists the available branches, and can be used with -d to delete branches

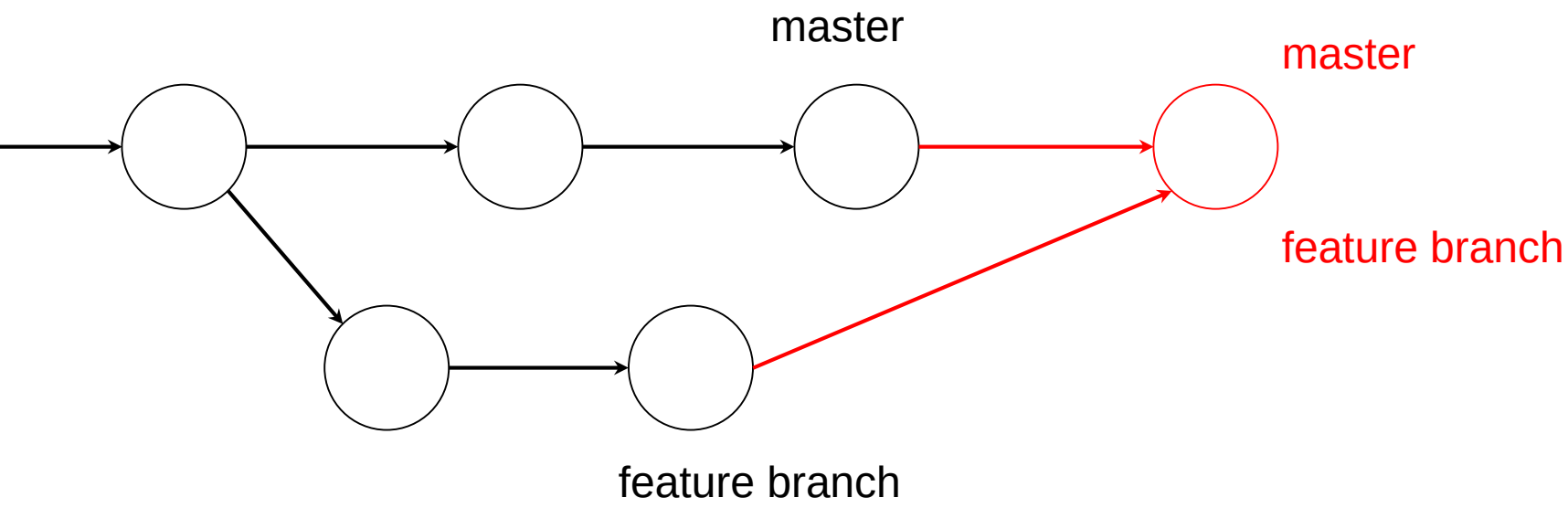
‘checkout’ switches the working copy to a specified branch.

Create a new branch using ‘checkout -b’

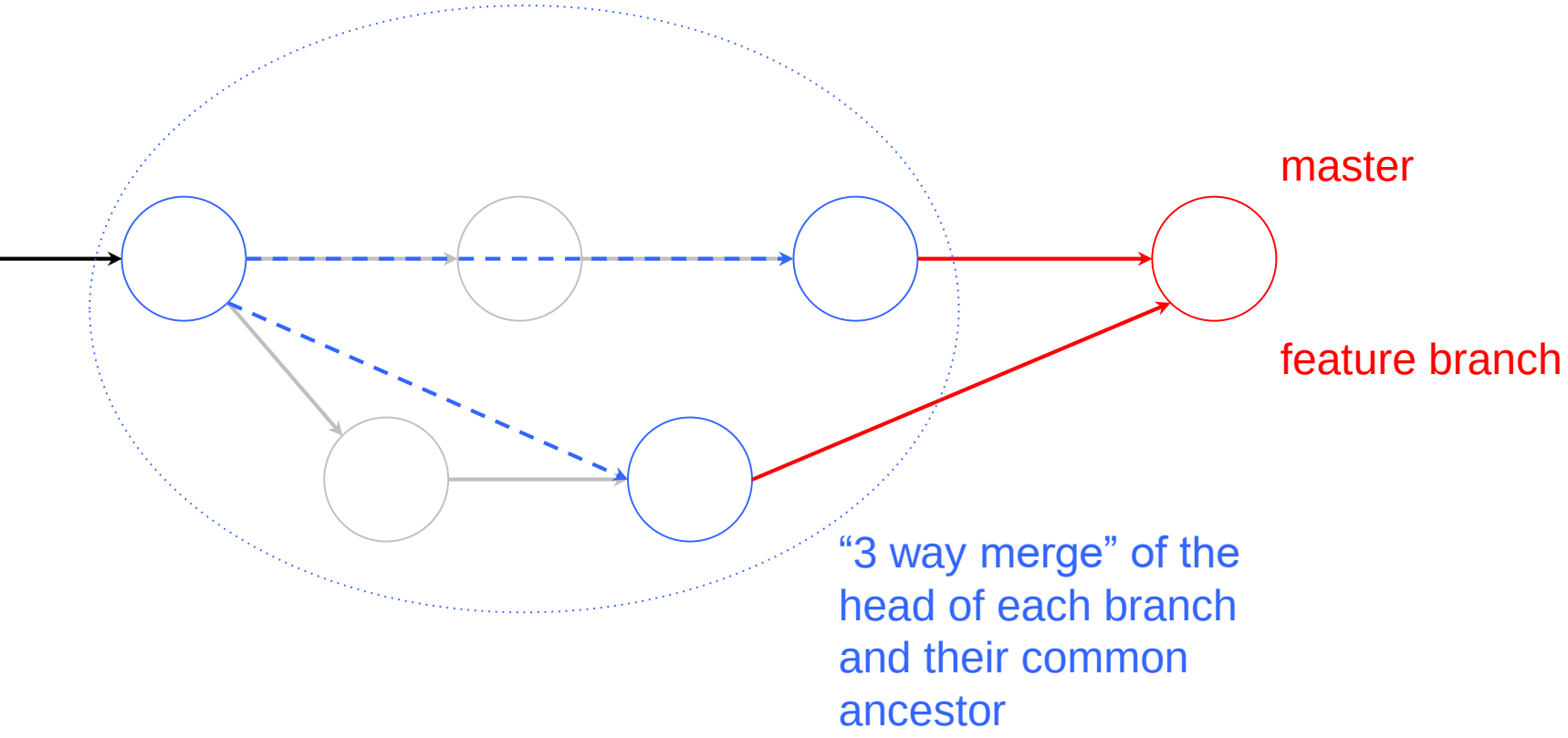
# Merging

Now Alistair wants some of the stuff
I did for Barbara too

# Merging



# Merging


master
feature branch
“3 way merge” of the head of each branch and their common ancestor

# Merging

Sometimes requires
manual intervention

This is more likely, the more changes have taken place since the branches were last merged

# Local Version Control

Revision history is on
your local machine

Lose it, and you’re toast

# Centralised Version Control

Revision history is on a
central server

You “check out” the current working set
(or a past snapshot)

“SVN is bad” – Leggy, 2013

# Distributed Version Control

You have a revision history on your local machine

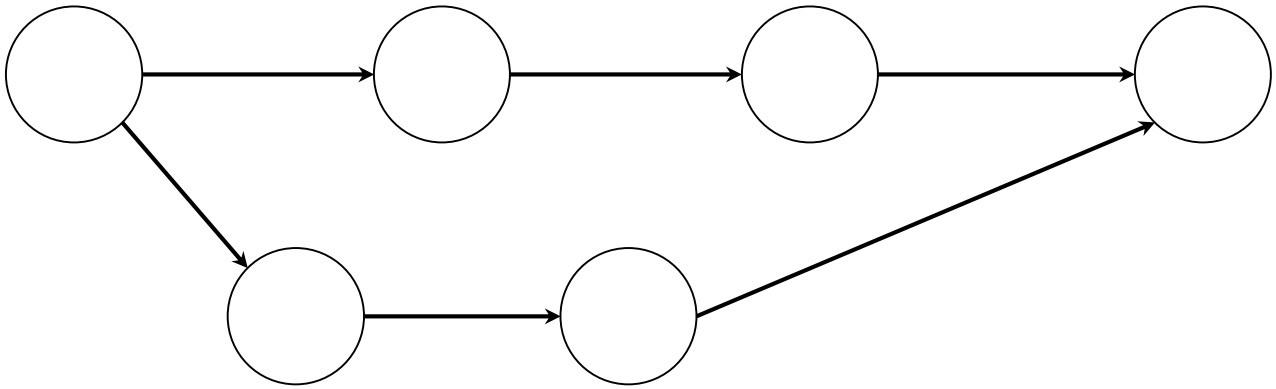
So does the server

So does every developer

You push and pull commits to each other to keep your repositories synchronised

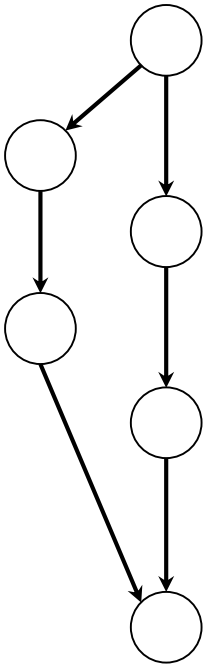
"Git is good" – Leggy, 2013

# History is a directed acyclic graph

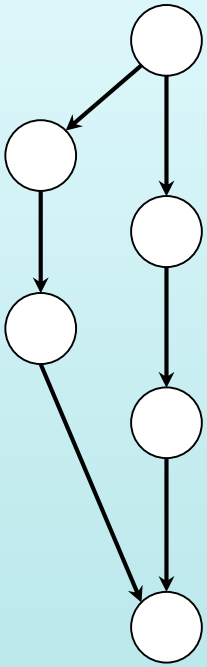


# History is a directed acyclic graph

local



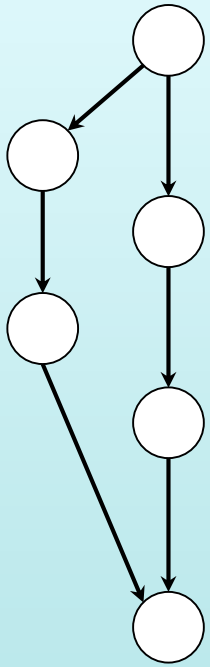
remote



# Clone

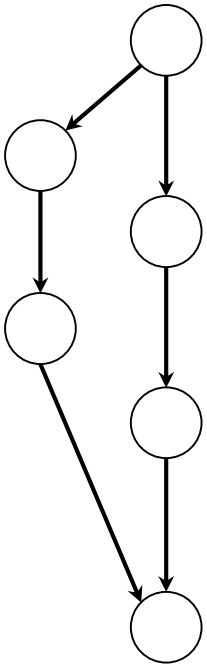
local

remote

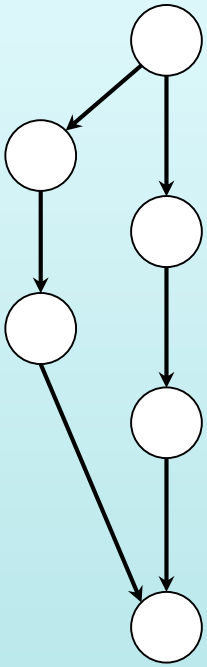


# Clone

local



remote

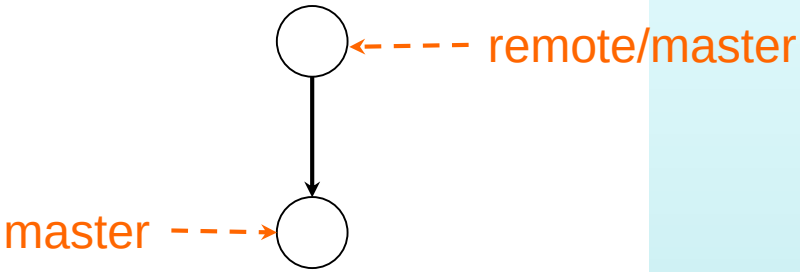


# Push

# Push

local

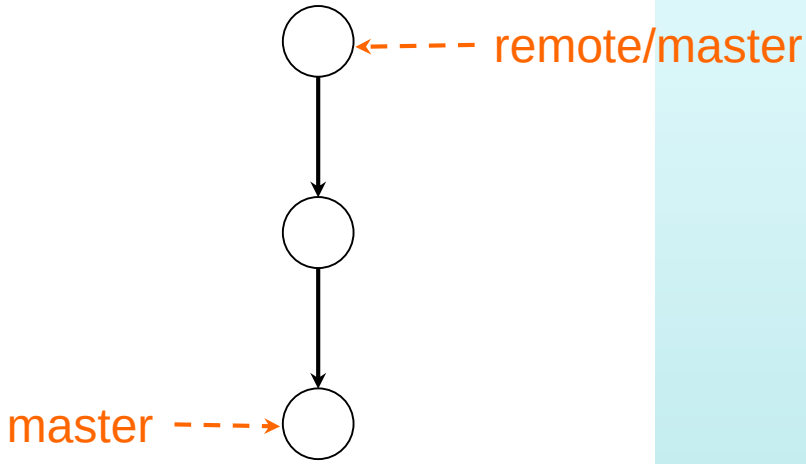
remote




# Push

local

remote




# Push

local

remote

# Fetch

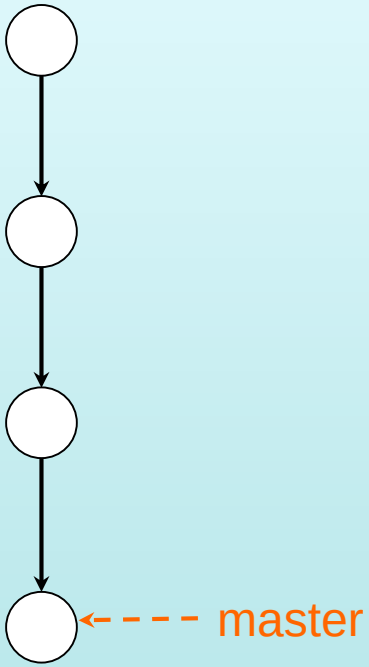
local

remote

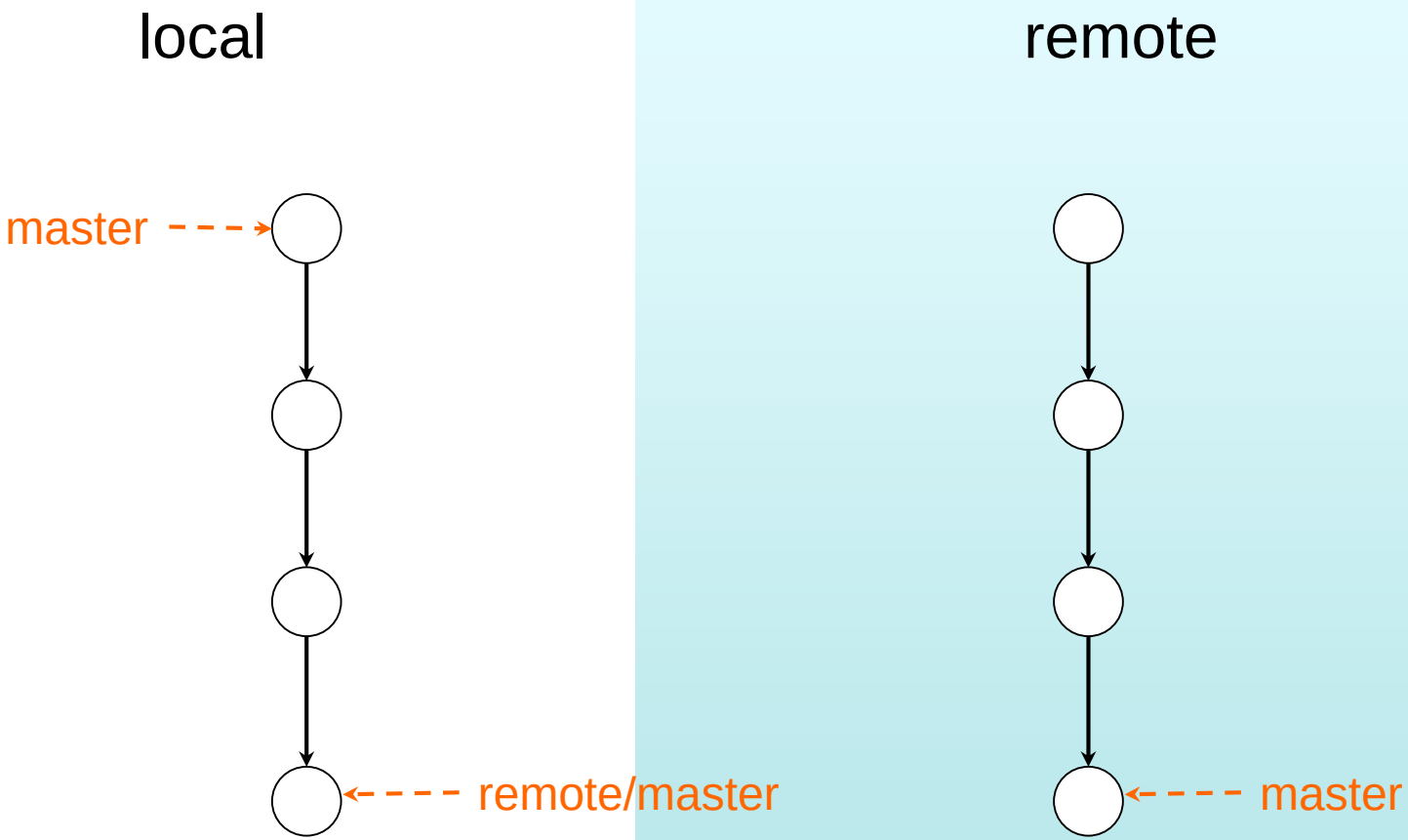

# Fetch

local

remote

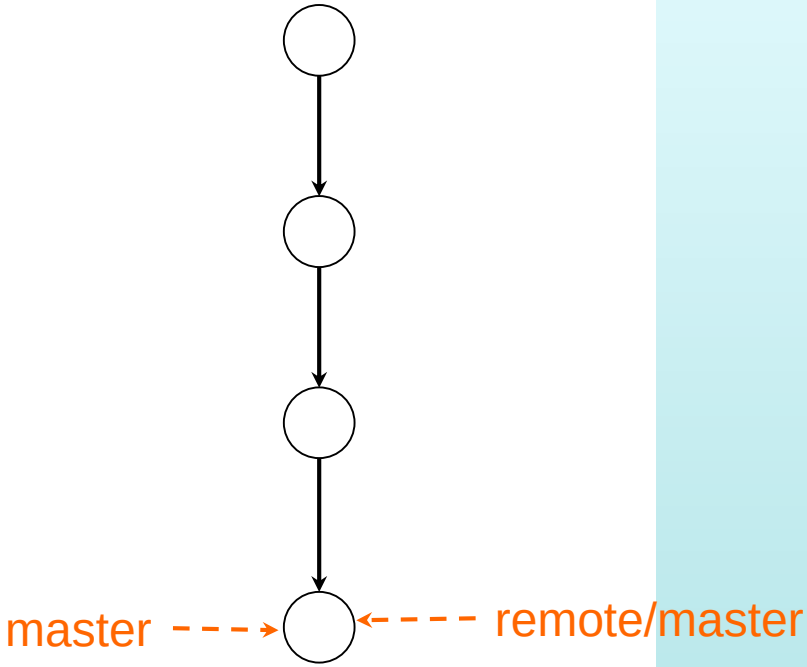
# Fetch

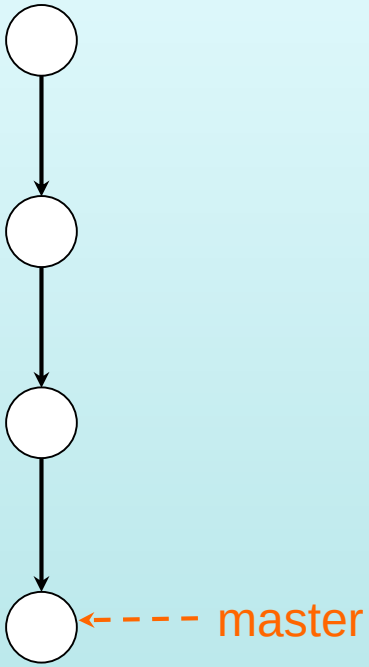


# Merge

local

remote

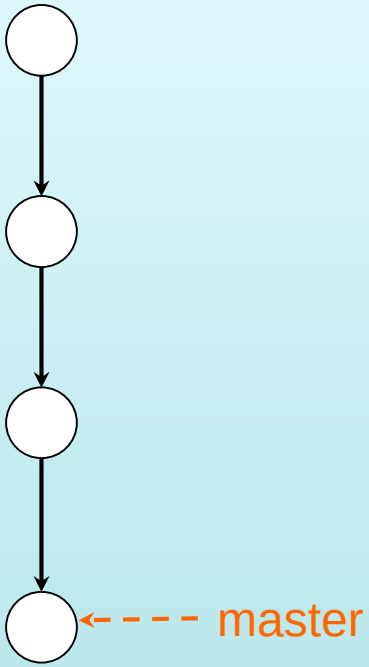


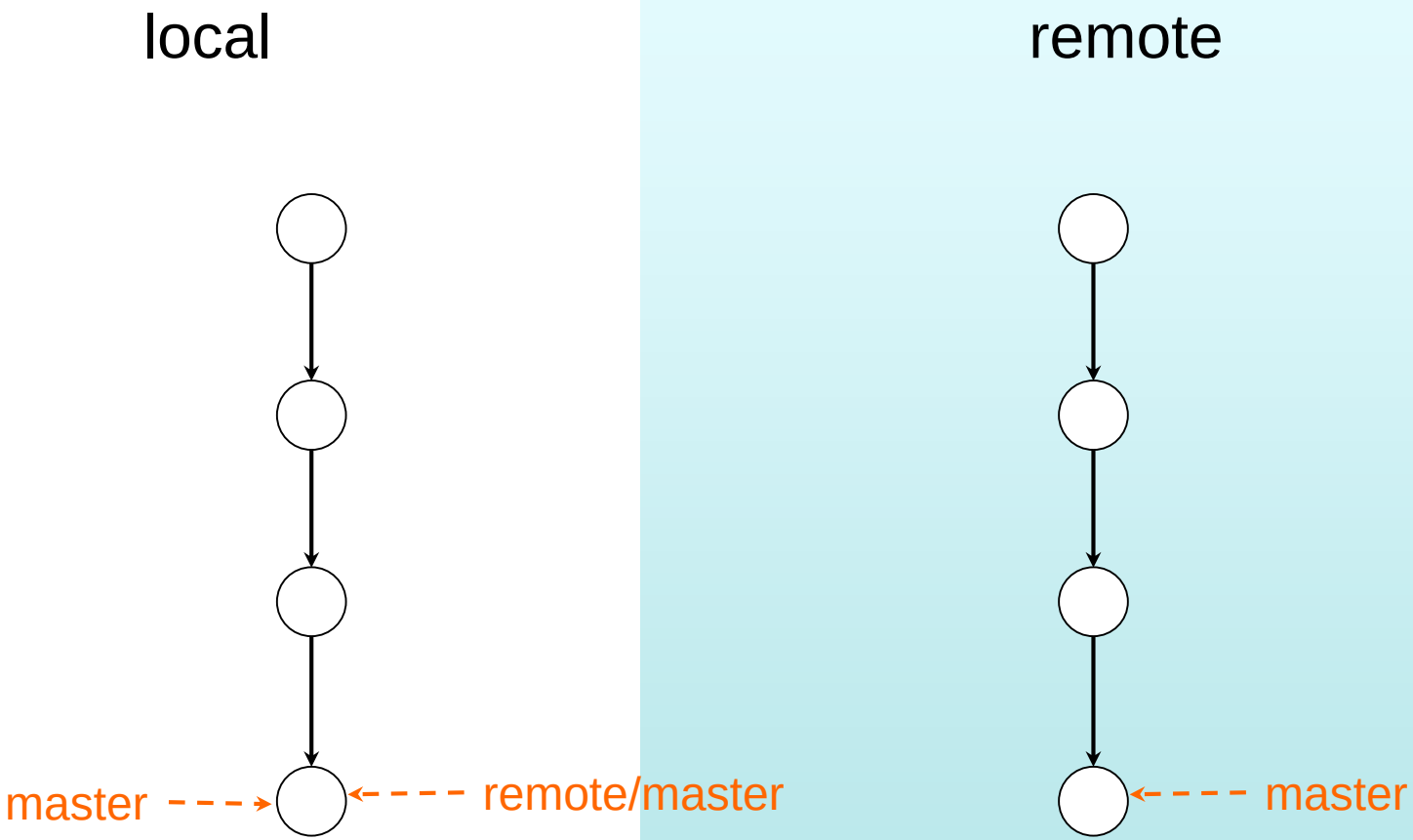
# Pull = fetch + merge

local

remote

# Pull = fetch + merge



# Tutorials

- Pre-requisite
  - Download and install git
    - https://git-scm.com/download/
  - Complete GIT100x
    - Link on BlackBoard
- Git exercise in tutorial
  - Tutors available to help
- Homework
  - Complete GIT200x & GIT300x