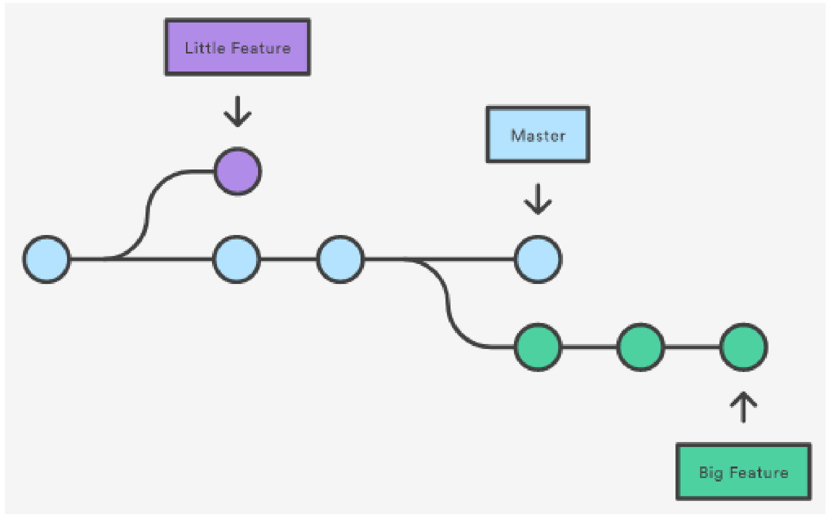
Git core concepts

Contents

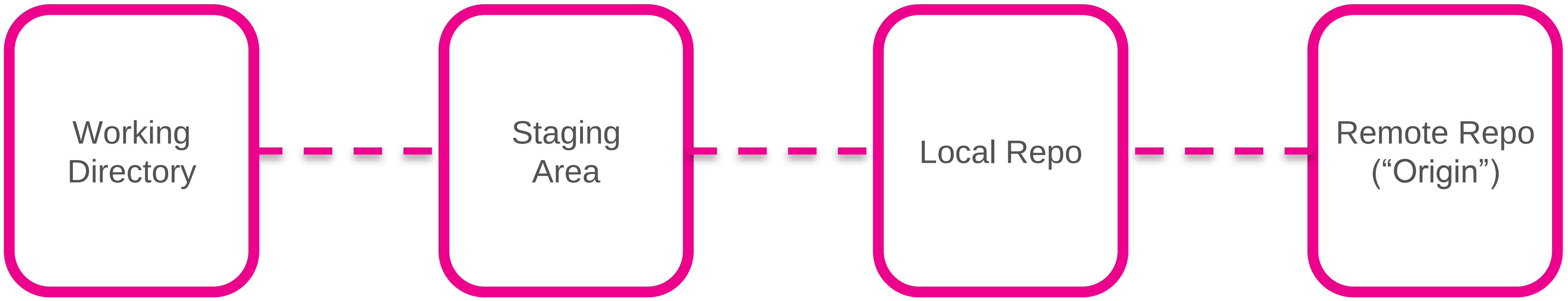
Git version control



Source: https://www.atlassian.com/git/tutorials

- Git records version history by storing sets of changes in units called **commits**
- A commit is just the deltas required to change the previous state of the files in the repo to the next state
- Each commit has its own ID code (SHA-1 hash)
- Version history is often visualised as a tree with dots representing commits - blue dots in the image above represent commits on the **Master** branch (default)
- The purple/green dots represent branches off the Master (more on branches later)

Git Areas



Working Directory – The files you're editing

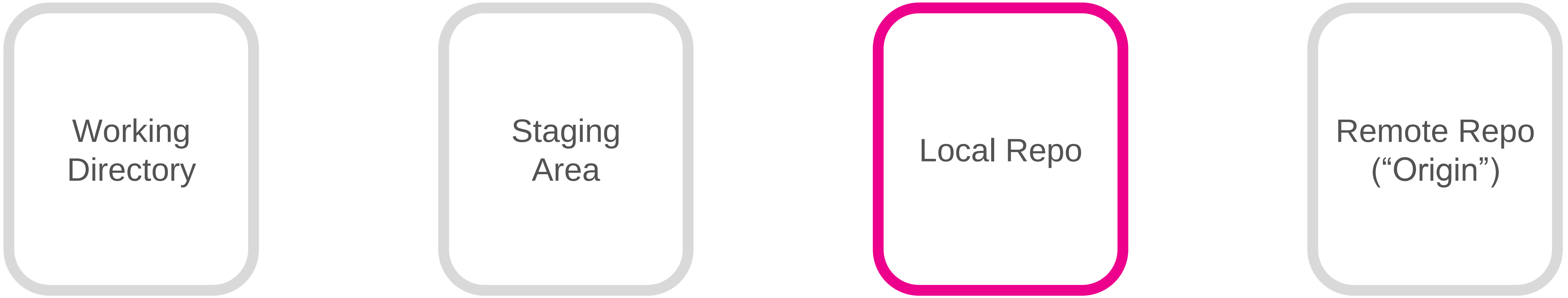
Staging Area – Where you assign which files you want to commit into the repository

Local Repo – The repository on your own machine

Remote Repo – The repository in the cloud or network (e.g. BitBucket), often named as "origin"

Creating Repositories

Create a local repo



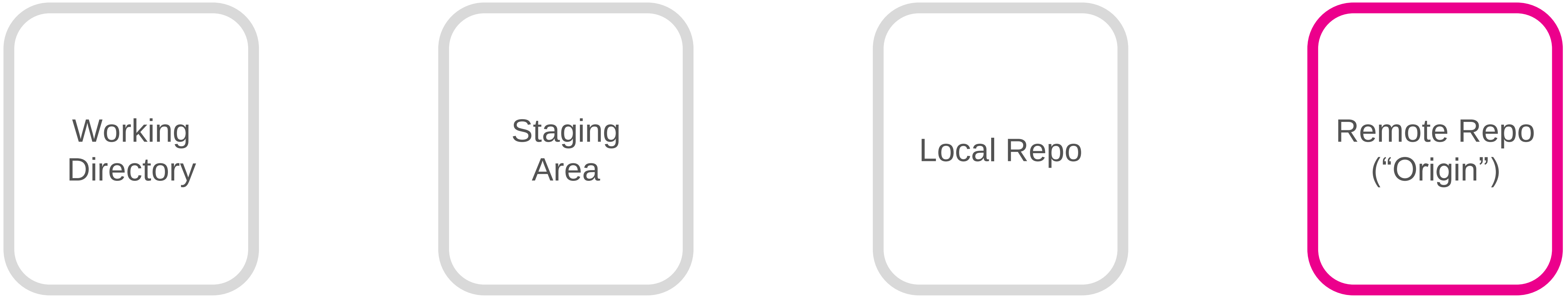
```
git init
```

Create a local git repository in the current directory

You can check by seeing if there's a hidden ".git" directory created

Creating Repositories

Create a remote repo



Bitbucket: New remote repositories

- You can create your own private repositories or fork a project repo
- Ask a lead to create a shared repository in a project.
- Good practice to create a README.md file

Creating Repositories

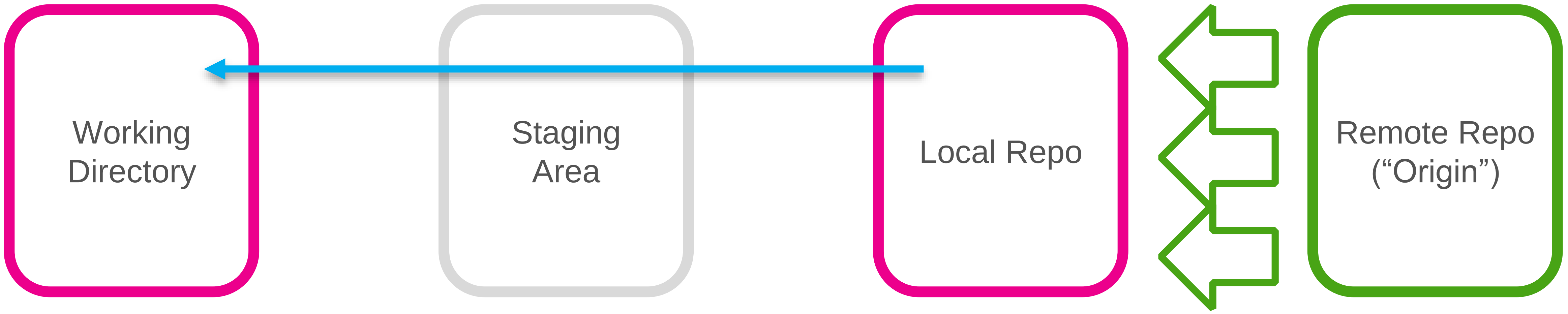
Best Practice: README.md

```
# Project Name
One-line description of the project

## Overview
Overview of the project
What does this do?
What was the motivation behind creating this?
When do you use it?

## Installation
Detailed instructions on how to install and set it up
Prerequisites?
Settings that need to be configured?
Tests to check that it's working properly?

## Examples
Give examples on how to use it
```

.md files use “markdown” for formatting
https://en.wikipedia.org/wiki/Markdown

Creating Repositories

Create a local repo that's a clone of a remote repo



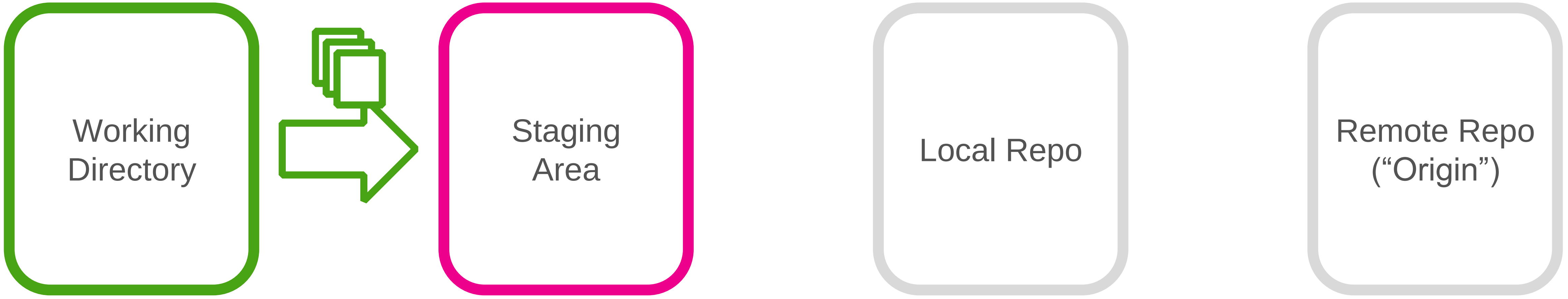
git clone <remote repo URL>

Create a git repository in the current directory that is a **copy of the remote repo** with full history of changes
Use --depth=1 if you only want to get the latest version without all the history

Committing changes

Committing Changes

Specify what needs to be committed



git add <file>

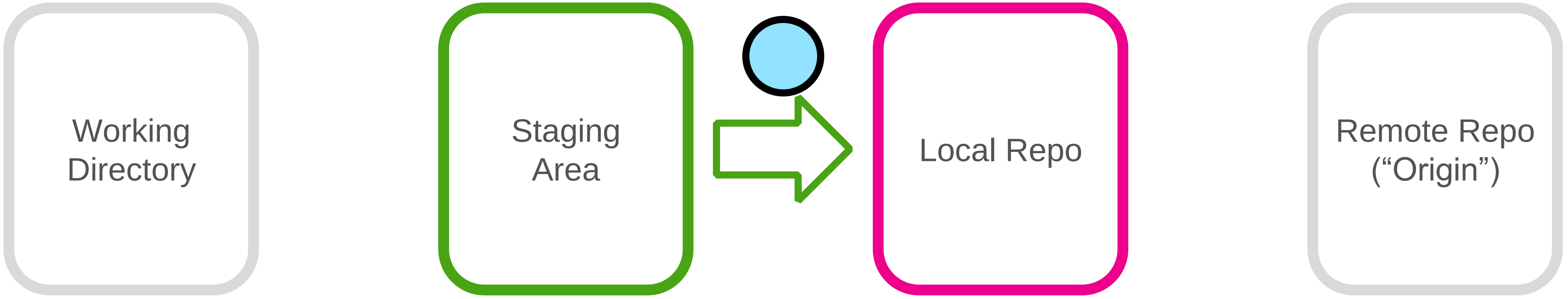
Add file(s) to staging area so they are ready to be committed
Only stage atomic changes

If you want to un-stage a file, use **git reset <file>**

TIP: Git does not track empty folders. If you think it's necessary, then create a placeholder file in there (e.g. ".keep").

Committing Changes

Commit changes into local repo



git commit -m "<message>"

Commit staged changes into the local repo ("add a dot to the chain")

Every commit should have a properly written message, there are two options:

- Use the -m option to write the message on command line
- If you have core.editor defined, you can omit the -m option and it will load up the text editor

Committing Changes

Best Practice: Atomic Commits

After your initial commit of the project, you should only commit atomic changes going forward.

Atomic changes are small and only encompass one irreducible task

- E.g. feature, fix, or improvement
- Just large enough to be a completed block of work so the code base is never broken

Benefits include:

- Easier code reviews, as all changes in the commit are for a single task
- Easier code merging
- Easier roll back, so if there’s a bug you can exclude one change at a time

TIP: If you make multiple changes at once on different files, you can use the staging process to divide it into multiple commits

Committing Changes

Best Practice: Commit Message Format

Subject line – Required, first line of the message

- Limit to 50 characters
- Use proper capitalisation
- Do not end with a period
- Use the imperative mood, like you're giving a command (e.g. "Fix" rather than "Fixed")
  - Think of it like this: "Applying this commit will <subject line>"

Body text – Optional

- Separate from subject line with a blank line
- Wrap text at 72 characters
- Explain what and why, rather than how

TIP: Use #<number> in message to link to a GitHub Issue

Source: https://chris.beams.io/posts/git-commit/

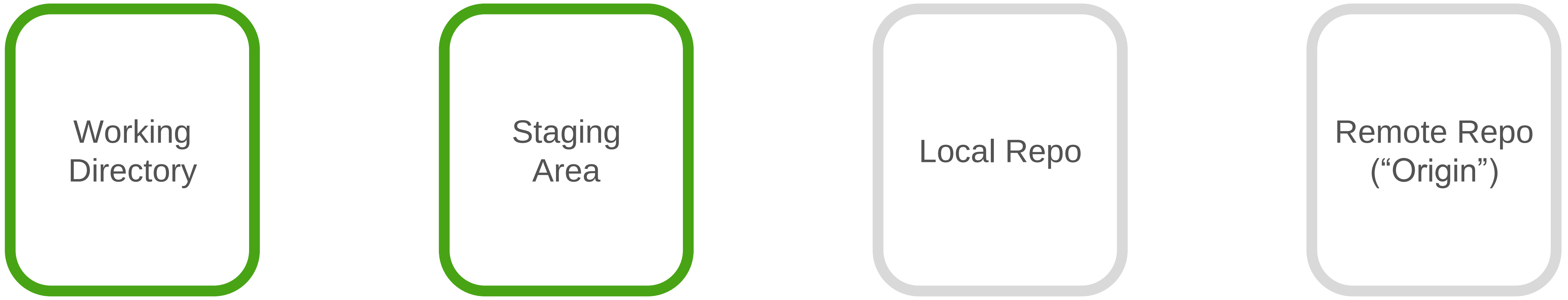
Don't get lazy! (source: xkcd)

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Inspecting Changes Before Committing

What files will be committed?



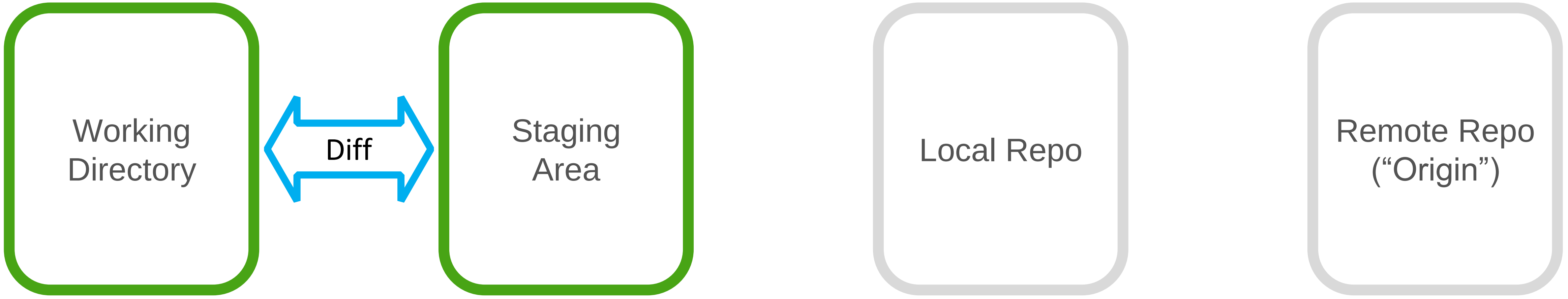
```
git status
```

Lists files in staging area that will be committed, AND
Lists changed files in working directory have not been staged

TIP: Always use this before committing to check you are committing exactly what you intended

Inspecting Changes Before Committing

What changes have I made?

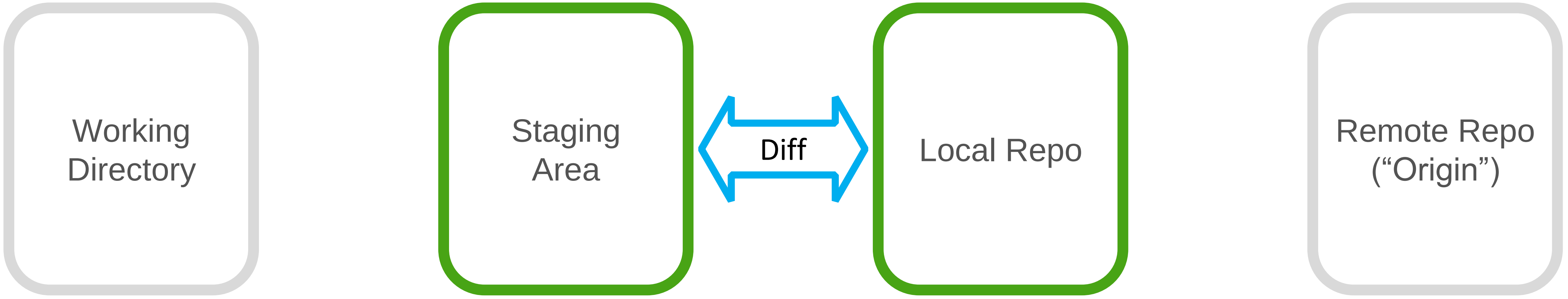


```
git diff
```

Compare differences between working directory and staging area
Effectively viewing the actual code/text changes that haven't been staged yet

Inspecting Changes Before Committing

What's the difference between staging and the repo?



```
git diff --staged
```

Compare differences between staging area and repository

View the actual code/text changes that you are about to commit

Diff Example

How to read diff output

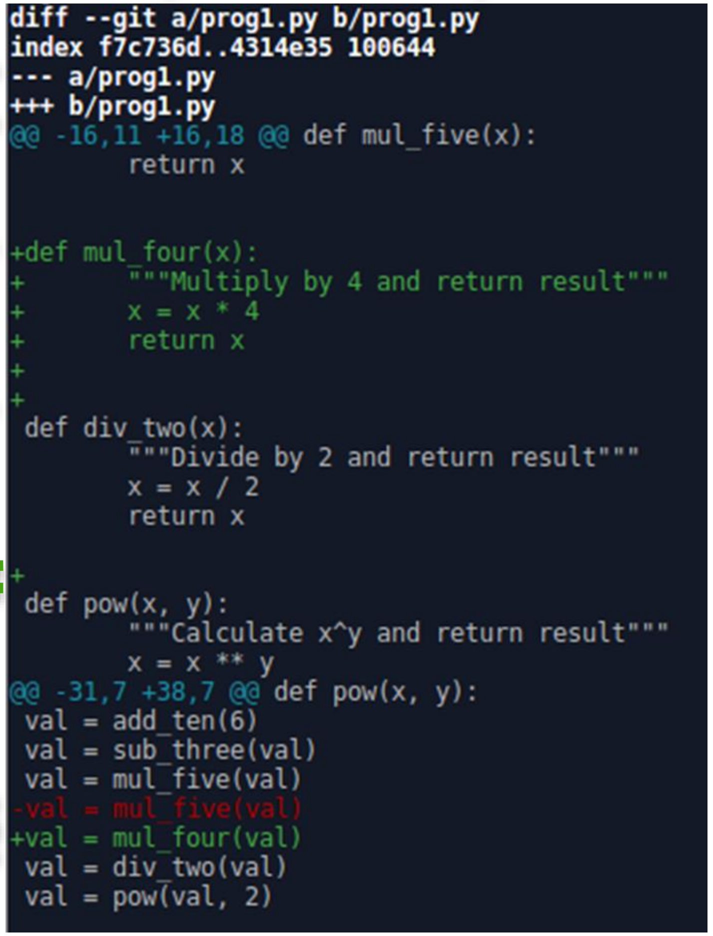
Comparing different versions of the same file

Version **a** represented by -
Version **b** represented by +

Version **b** (+) has these lines

Version **b** (+) has this blank line

Version **a** (-) has this line
Version **b** (+) has this line



```
diff --git a/prog1.py b/prog1.py
index f7c736d..4314e35 100644
--- a/prog1.py
+++ b/prog1.py
@@ -16,11 +16,18 @@ def mul_five(x):
        return x


+def mul_four(x):
+       """Multiply by 4 and return result"""
+       x = x * 4
+       return x
+
+
 def div_two(x):
        """Divide by 2 and return result"""
        x = x / 2
        return x

+
 def pow(x, y):
        """Calculate x^y and return result"""
        x = x ** y
@@ -31,7 +38,7 @@ def pow(x, y):
 val = add_ten(6)
 val = sub_three(val)
 val = mul_five(val)
-val = mul_five(val)
+val = mul_four(val)
 val = div_two(val)
 val = pow(val, 2)
```

@@ indicates start of a chunk of code

-16,11
From version **a** (-), starting line 16, showing 11 lines

+16,18
From version **b** (+), starting line 16, showing 18 lines

@@ indicates start of another chunk of code

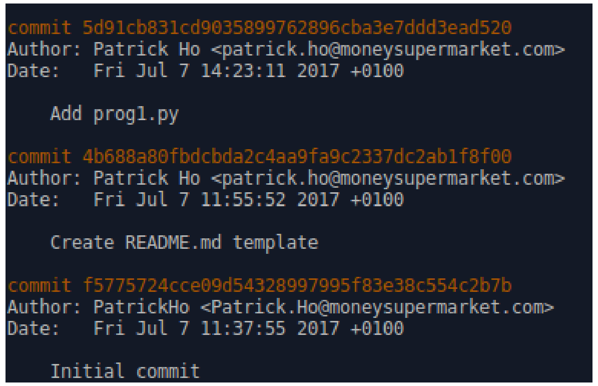
-31,7
From version **a** (-), starting line 31, showing 7 lines

+38,7
From version **b** (+), starting line 38, showing 7 lines

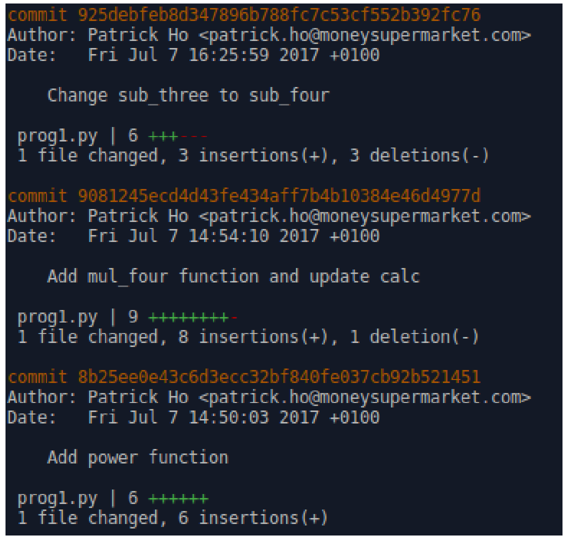
Inspecting Change History

Inspecting Commit History

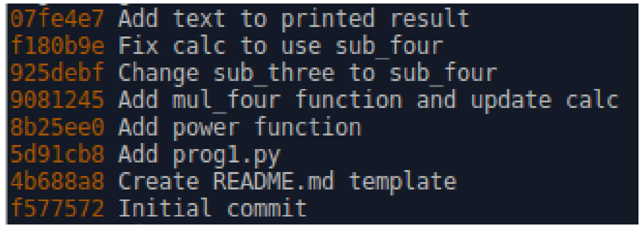
What changes have been committed to the repo?



```
commit 5d91cb831cd9035899762896cba3e7ddd3ead520
Author: Patrick Ho <patrick.ho@moneysupermarket.com>
Date:   Fri Jul 7 14:23:11 2017 +0100

    Add prog1.py

commit 4b688a80fbdcbda2c4aa9fa9c2337dc2ab1f8f00
Author: Patrick Ho <patrick.ho@moneysupermarket.com>
Date:   Fri Jul 7 11:55:52 2017 +0100

    Create README.md template

commit f5775724cce09d54328997995f83e38c554c2b7b
Author: PatrickHo <Patrick.Ho@moneysupermarket.com>
Date:   Fri Jul 7 11:37:55 2017 +0100

    Initial commit
```



```
commit 925debfeb8d347896b788fc7c53cf552b392fc76
Author: Patrick Ho <patrick.ho@moneysupermarket.com>
Date:   Fri Jul 7 16:25:59 2017 +0100

    Change sub_three to sub_four

 prog1.py | 6 +++---
 1 file changed, 3 insertions(+), 3 deletions(-)

commit 9081245ecd4d43fe434aff7b4b10384e46d4977d
Author: Patrick Ho <patrick.ho@moneysupermarket.com>
Date:   Fri Jul 7 14:54:10 2017 +0100

    Add mul_four function and update calc

 prog1.py | 9 ++++++++-
 1 file changed, 8 insertions(+), 1 deletion(-)

commit 8b25ee0e43c6d3ecc32bf840fe037cb92b521451
Author: Patrick Ho <patrick.ho@moneysupermarket.com>
Date:   Fri Jul 7 14:50:03 2017 +0100

    Add power function

 prog1.py | 6 ++++++
 1 file changed, 6 insertions(+)
```



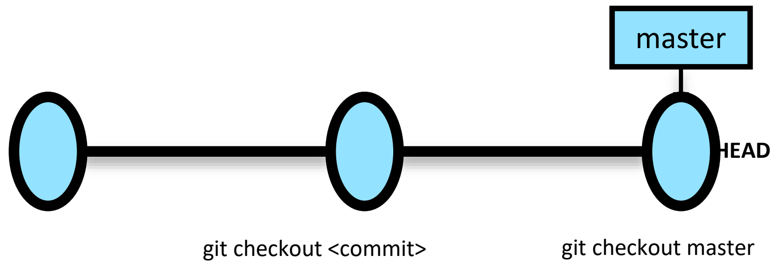
```
07fe4e7 Add text to printed result
f180b9e Fix calc to use sub_four
925debf Change sub_three to sub_four
9081245 Add mul_four function and update calc
8b25ee0 Add power function
5d91cb8 Add prog1.py
4b688a8 Create README.md template
f577572 Initial commit
```

git log

Show the commit history
Use --stat to show additional information about the committed changes
Use --oneline to condense each commit to a single line

TIP: Good commit messages == Better looking and more useful commit history!

The HEAD



HEAD

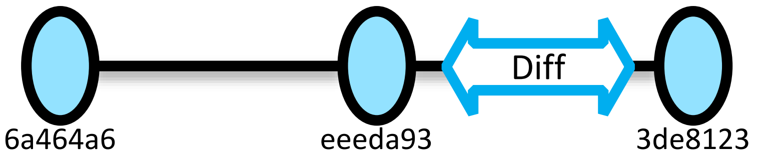
This is a reference to the most recent commit in the currently checked out branch

If you checkout a specific commit instead of a branch, you will be in a 'detached HEAD' state
Any new commits while in this state won't be reachable by any branch – you won't see the commit in your git log

TIP: If you need to make changes to old versions, create a branch: git checkout -b <branch> <commit>
TIP: If you need to find 'unreachable' commits, you can use git reflog

Inspecting Commit History

What changed between these two commits?

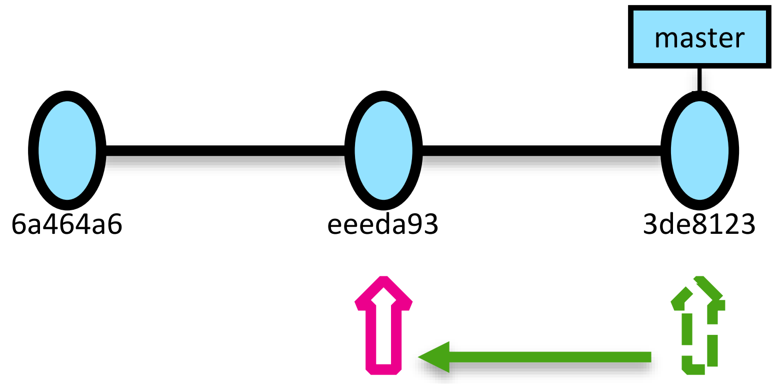


`git diff` <base commit ID> <compare commit ID>

Compare differences between two commits (typically the base commit is older)
The two commits don't have to be adjacent
You can use shortened commit IDs

Inspecting Commit History

What did the files look like as at a previous commit?



`git checkout` <commit ID>

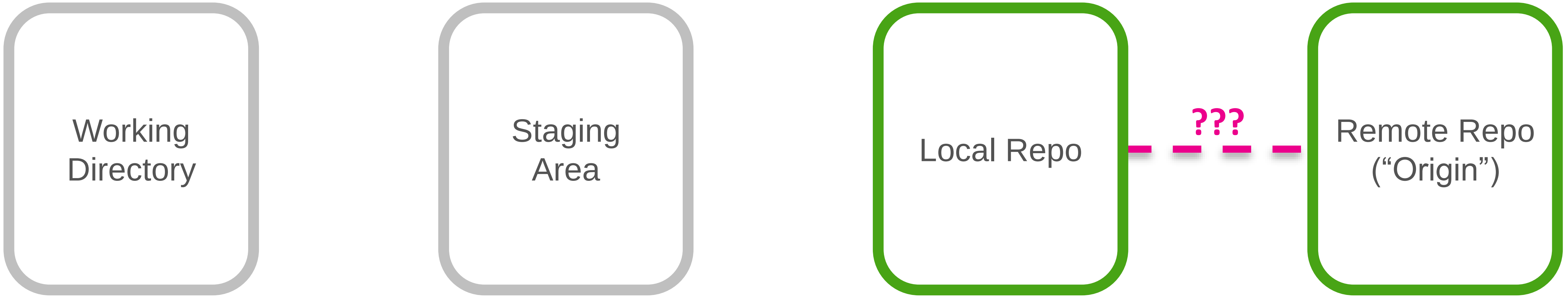
View files as at a particular point in history

This moves you 'back in time', so you won't see later commits in the log
To get back to the latest version of the branch: git checkout <branch, e.g. master>

Synchronising repositories

Managing Remote Repositories

What remote repositories are connected with this repository?



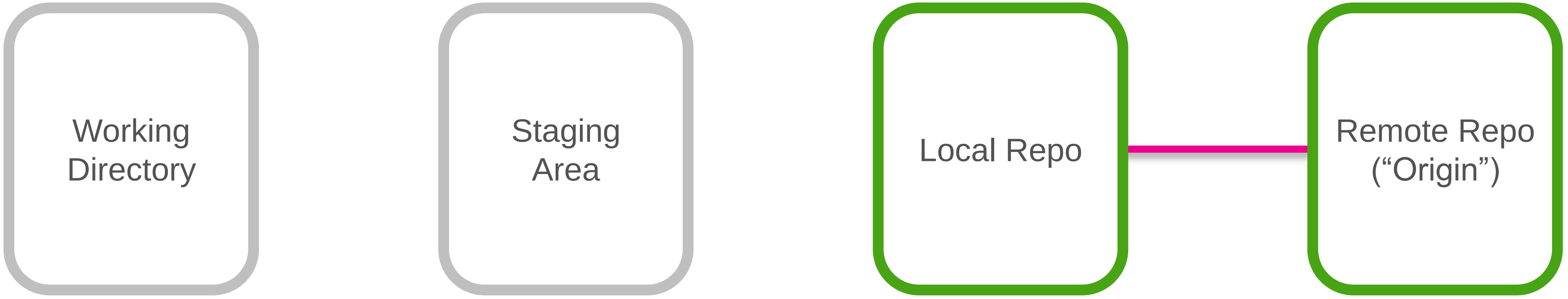
```
git remote
```

View remote repositories connected with this local repo

Use -v for verbose listing

Managing Remote Repositories

Connect a remote repository

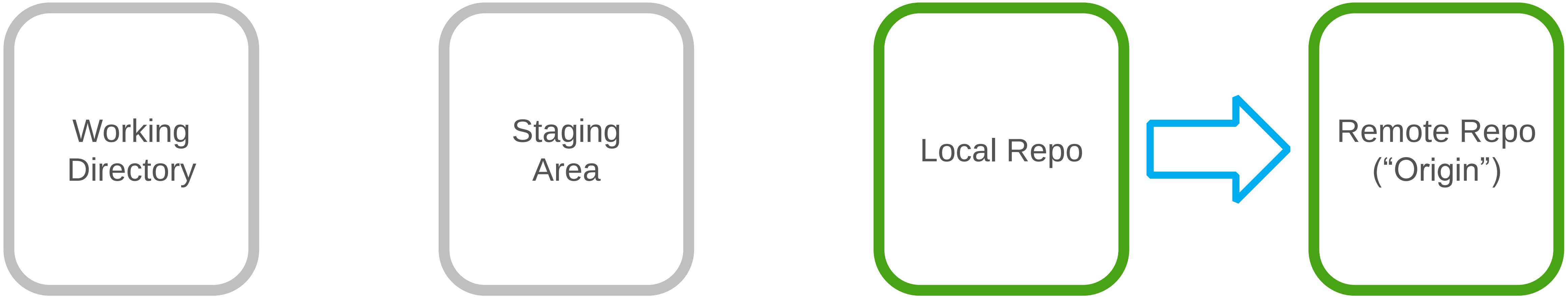


`git remote add` <name of remote, e.g. origin> <address, e.g. BitBucket repo>

Create a remote connection
If the local repo was cloned from a remote repo, the connection will already be established

Syncing Local and Remote Repos

Push local commits up to remote repo

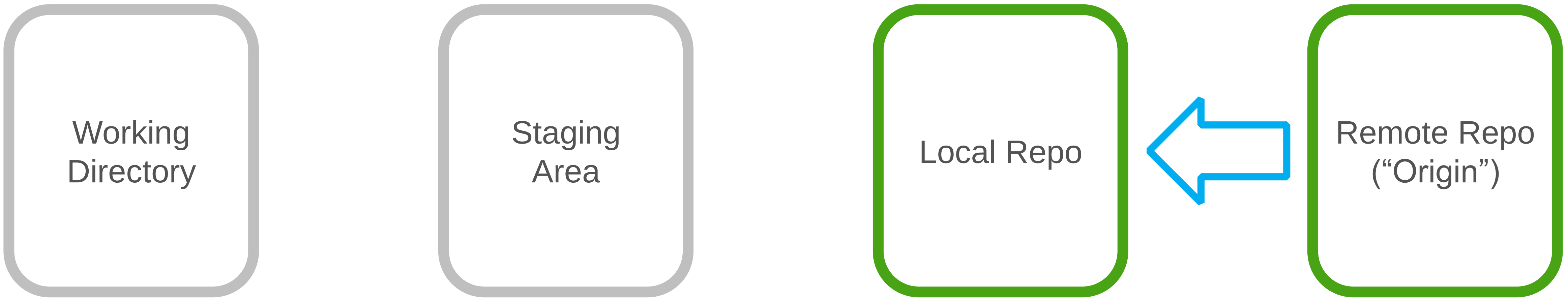


```
git push <remote> <branch>
```

Push (upload) commits on the branch to the remote repository

Syncing Local and Remote Repos

Fetch remote commits down to local repo



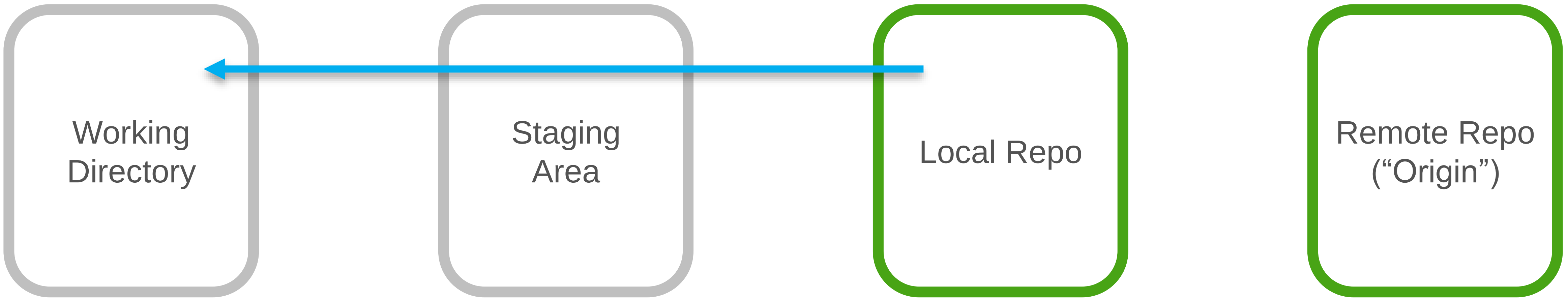
```
git fetch <remote> <branch>
```

Download commits on the specified branch from the remote repository to your local repository.

- The resulting commits will appear in your local repo on a “remote branch”
- This allows you to see what’s been done on the remote repo without impacting your own work
- You can also work on the remote branch and push changes back up to the branch on the remote repo

Syncing Local and Remote Repos

Merge commits from one branch into another



`git merge` <source branch to merge>

Merge commits from specified branch into the current working branch.

The merge command can do many other types of merge – more details here:
https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
https://git-scm.com/docs/git-merge

Pull examples

`git pull origin master`

Pull changes from the remote to the current branch

`git pull`

Short form of **`git pull origin master`**

`git pull origin feature/GRPDT-100`

Fetch and merge the remote feature branch into the current branch in your local repo.