



Introduction to Git



Basics of Git and its working

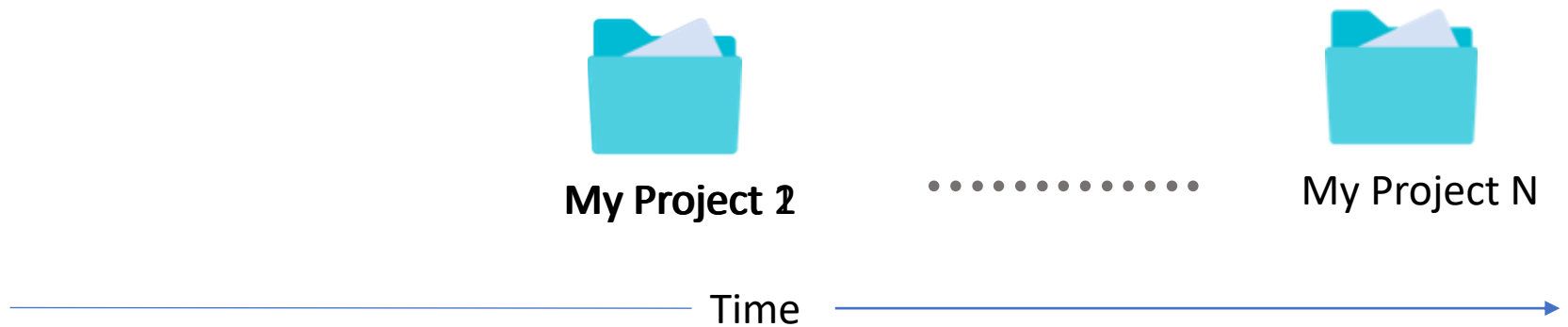
By

Bhuvaneshwaran Ilanthirayan

What is Git ?

It is a version control System

What is a Version Control System?



What are these?

Versions

What is Git?

- It is a version control system
- Distributed Version control
- Cross- platform
- Open source, free
- A de factor standard of its kind



Wait, then what is GitHub?

- A web-based hosting service
- Popular option



You can still use git without GitHub



Installing Git

- For Linux,
 - `sudo apt-get update`
 - `sudo apt-get install git`
- For Windows,
 - Go to gitforwindows.org
 - Download the installer and install it
- For Mac,
 - Go to sourceforge.net/projects/git-osx-installer/files/
 - Follow the prompts and install it

Configuring Git

```
$ git --version  
git version 2.42.0.windows.1
```

```
$ git config --global user.name "<Your name>"
```

```
$ git config --global user.email "<youremail@anywebsite.com>"
```

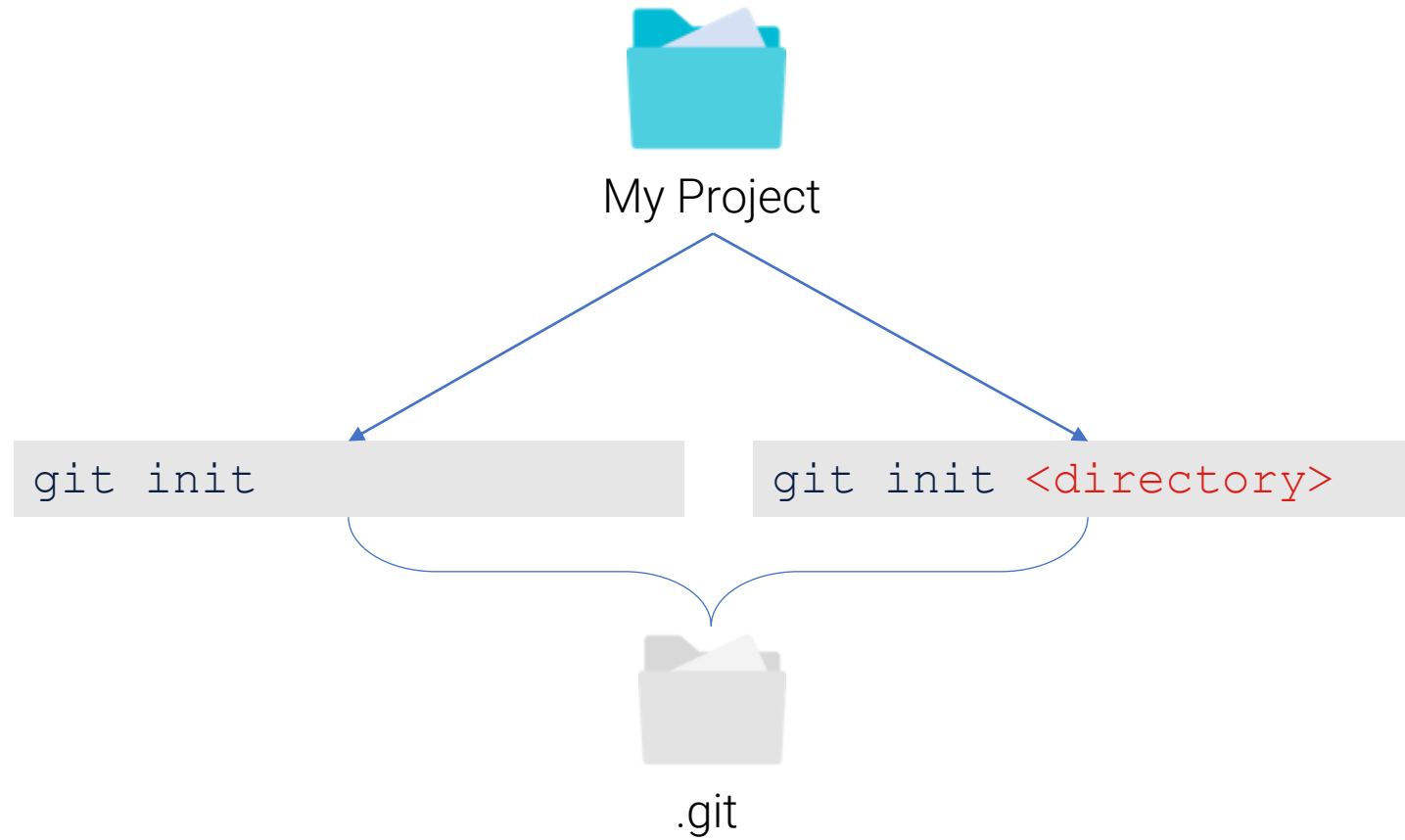
Lets move on to Git Basics

Setup a Repository, Saving the changes, Inspecting a repository, undoing changes

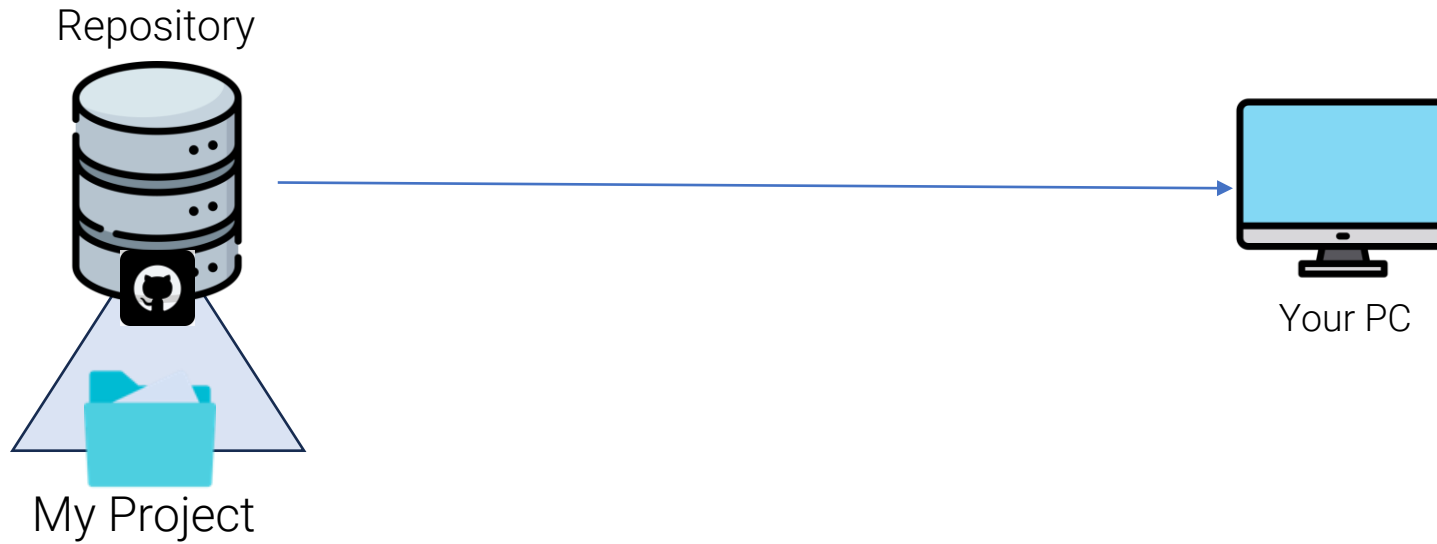
Git Basics

Setup a Repository

git init



git clone



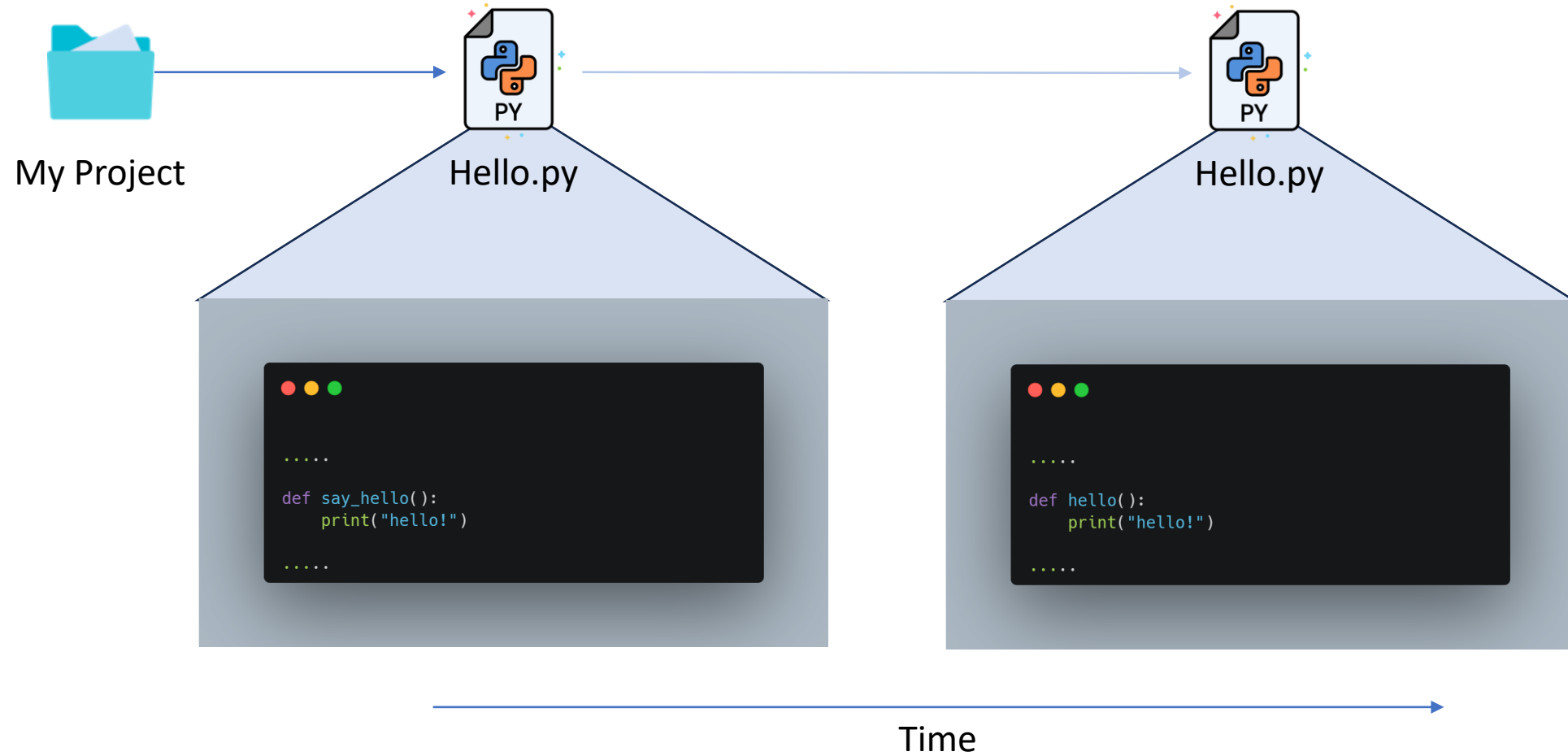
```
git clone <repo>
```

```
git clone <repo> <directory>
```

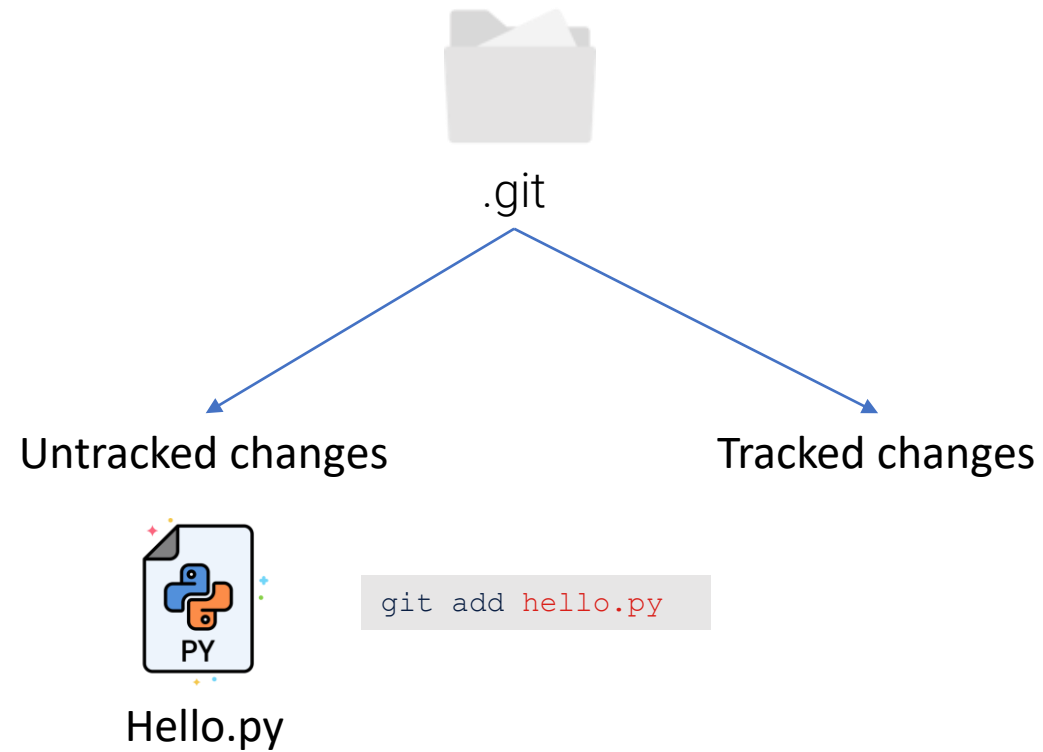
Git Basics

Saving the changes

How to save changes in git?



git add



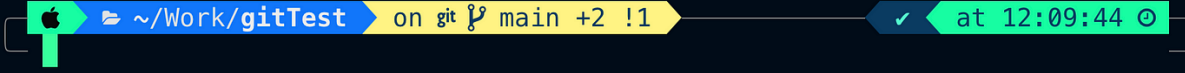
git status

```
ebcom@Cains-blade:~/Work/gitTest
> git status
On branch main

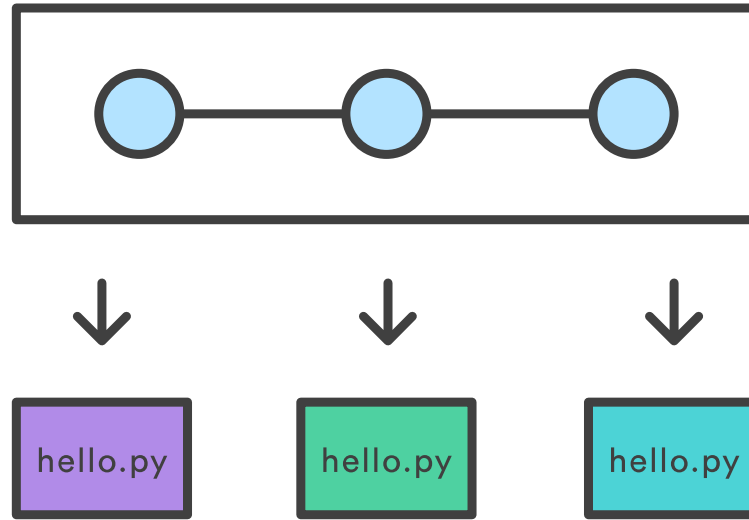
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file2.txt
        new file:   script.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2.txt
```

A horizontal status bar at the bottom of the terminal window. It contains an Apple logo icon, a folder icon, the path ~/Work/gitTest, the text 'on git main +2 !1', a checkmark icon, and the time 'at 12:09:44' followed by a circular icon.

git commit

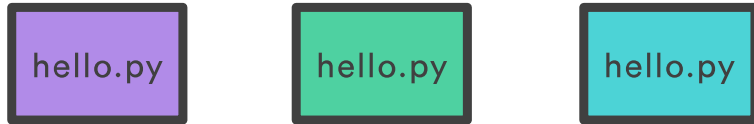
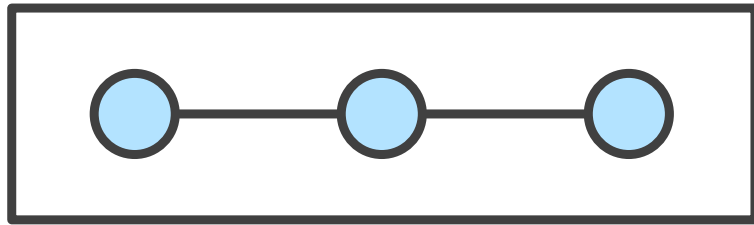


```
git commit
```

```
git commit -a
```

```
git commit -m "commit message"
```


git diff



What is the difference?

```
git diff
```

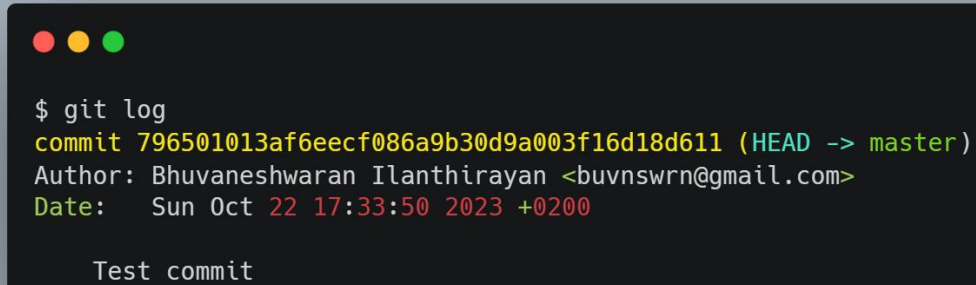
```
git diff HEAD ./path/to/file
```

```
$ git diff HEAD hello.py
diff --git a/hello.py b/hello.py
index d55f52f..3dfa812 100644
--- a/hello.py
+++ b/hello.py
@@ -1,2 +1,2 @@
-def say_hello():
+def hello():
     print("hello!")
```

Git Basics

Inspecting a repository

git log

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays the output of the 'git log' command. The output shows a single commit with a green commit hash, a green branch name in parentheses, the author's name and email, a date and time in red, and a commit message.

```
$ git log
commit 796501013af6eecf086a9b30d9a003f16d18d611 (HEAD -> master)
Author: Bhuvaneshwaran Ilanthirayan <buvnswrn@gmail.com>
Date:   Sun Oct 22 17:33:50 2023 +0200

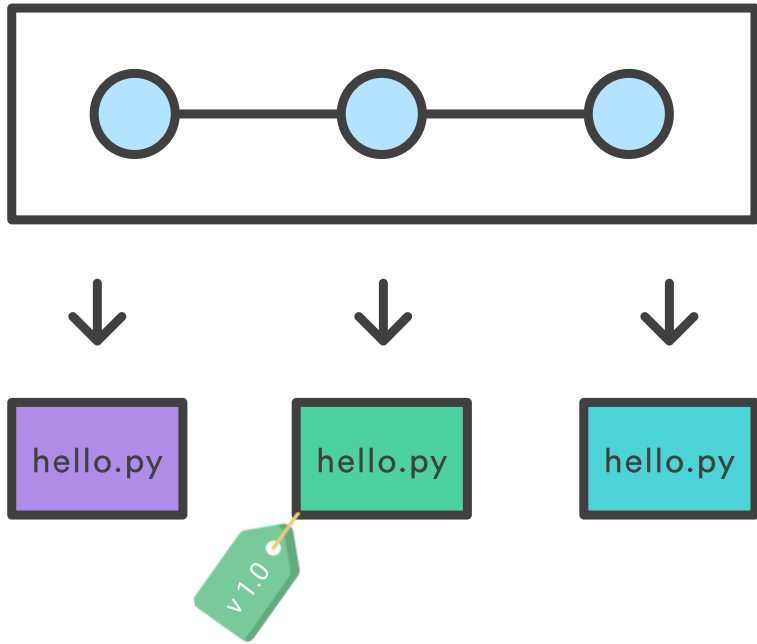
    Test commit
```

```
git log
```

```
git shortlog
```

```
git log --graph --oneline --decorate
```

git tag



```
git tag <tagname>
```

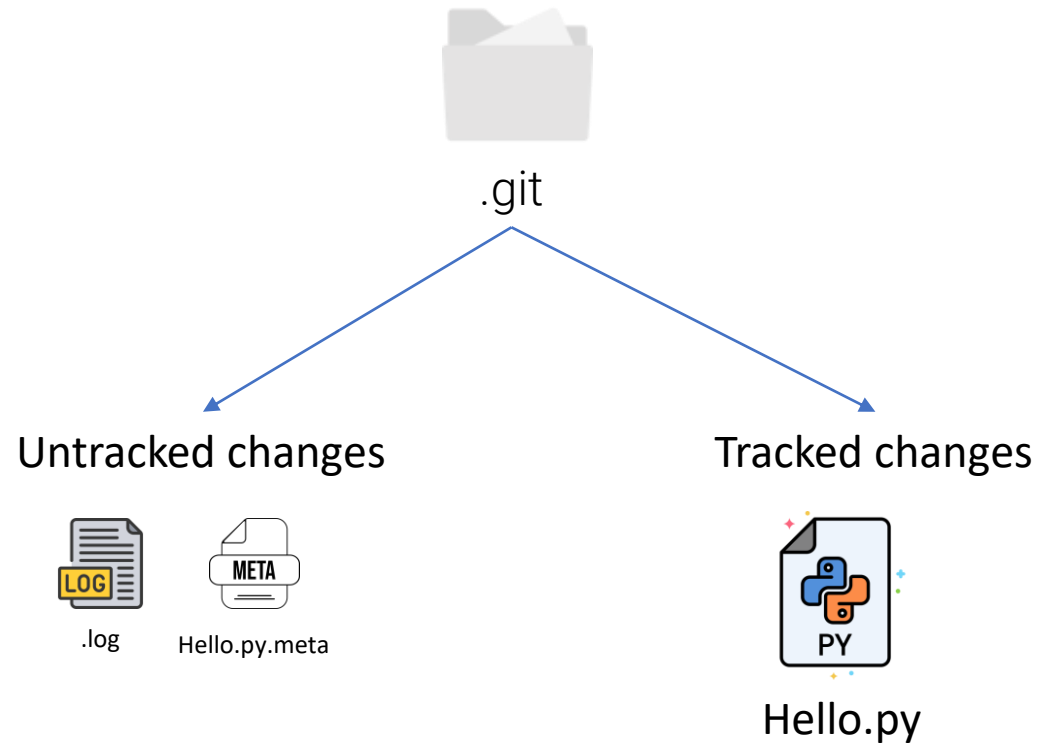
```
git tag -a v1.0 -m "my version 1.0"
```

```
git tag
```

Git Basics

Undoing changes

git clean



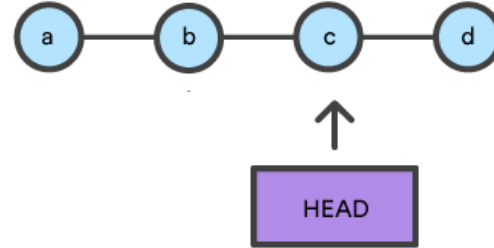
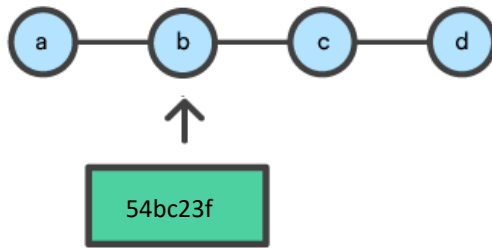
```
git clean
```

```
git clean -n
```

```
git clean -f
```

```
git clean -di
```

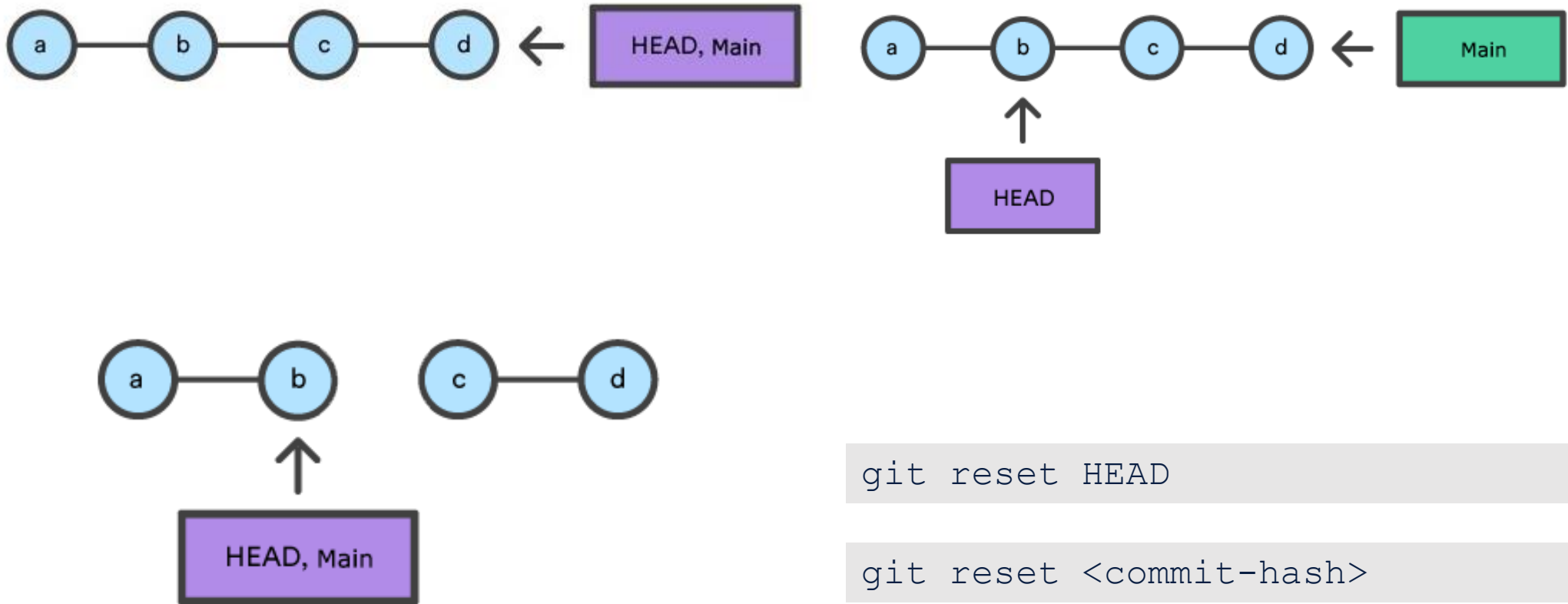
git revert



```
git reset HEAD
```

```
git reset <commit-hash>
```

git reset



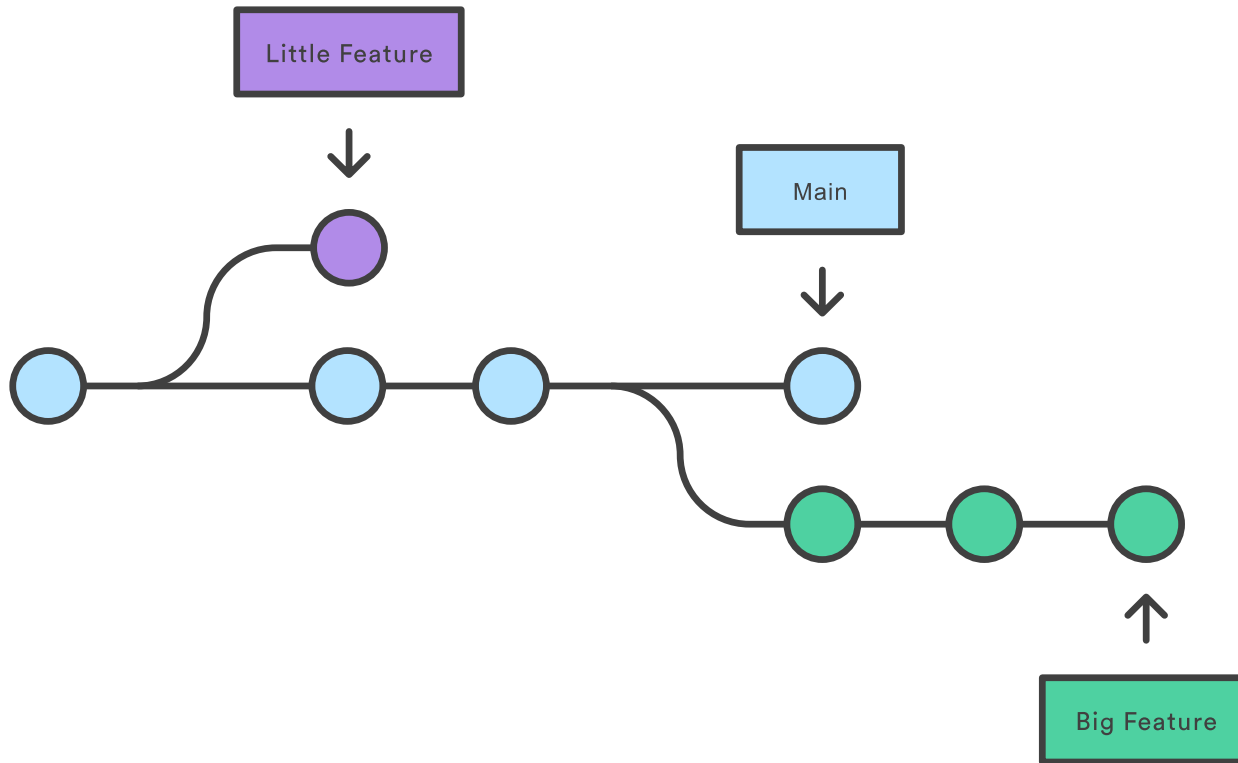
Lets see some advanced topics

Syncing, Git Branch, Rewriting History

Git Advanced Topics

Git Branch

git branch and git checkout



```
git branch
```

```
git branch <branch>
```

```
git branch -d <branch>
```

```
git branch -m <branch>
```

```
git branch -a
```

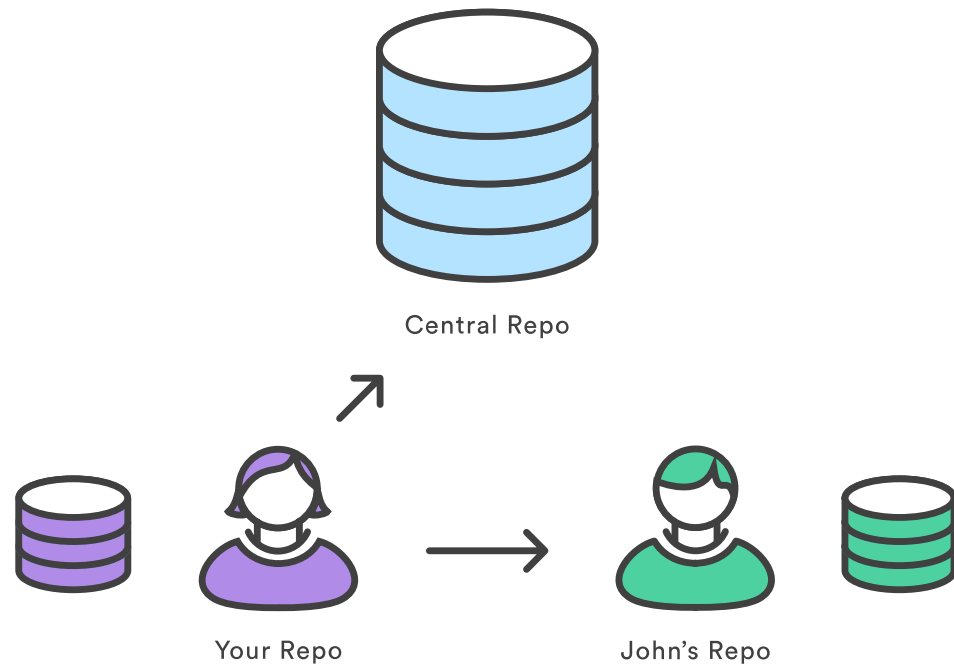
```
git checkout <branch name>
```

```
git checkout -b <new-branch>
```

Git Advanced Topics

Syncing

git remote



```
git remote
```

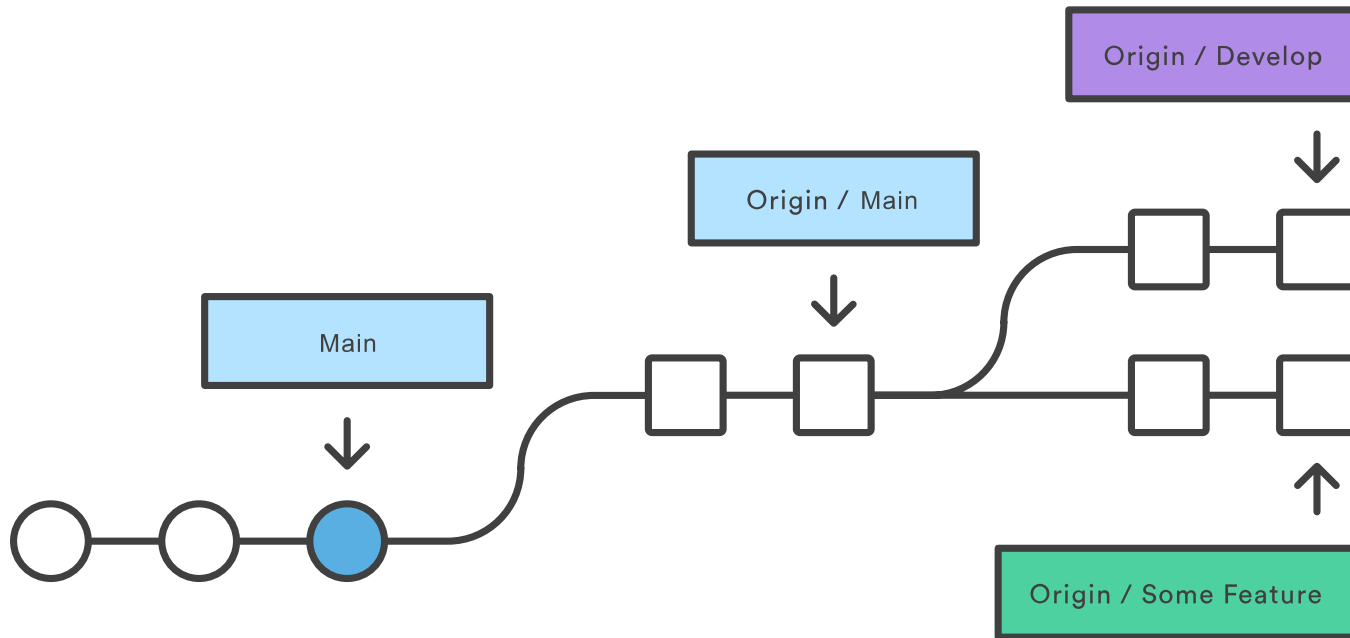
```
git remote -v
```

```
git remote add <name> <url>
```

```
git remote rm <name>
```

```
git remote rename <name> <newname>
```

git fetch



```
git fetch <remote>
```

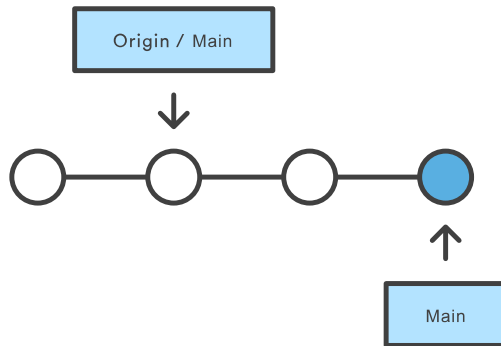
```
git fetch <remote> <branch>
```

```
git fetch --dry-run
```

```
git fetch origin
```

git push

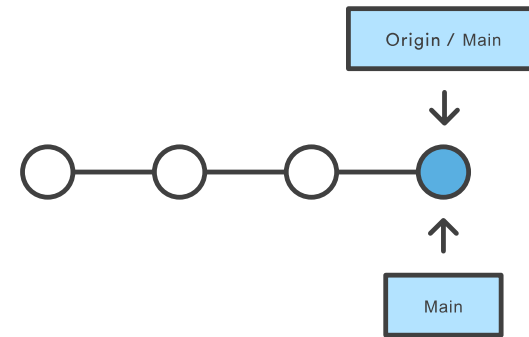
Before Pushing



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

```
git push <remote> --force
```

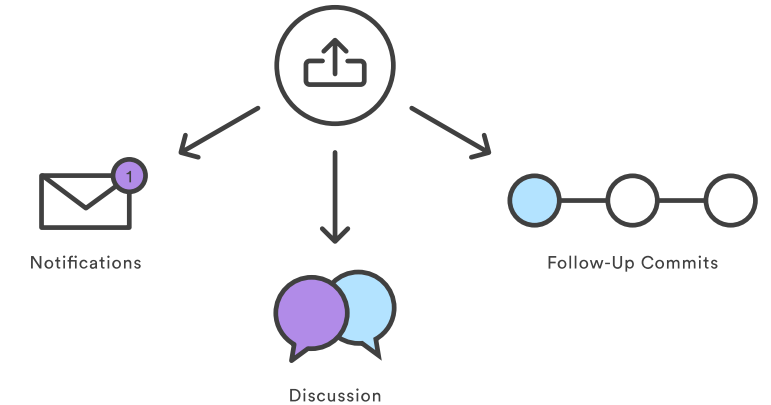
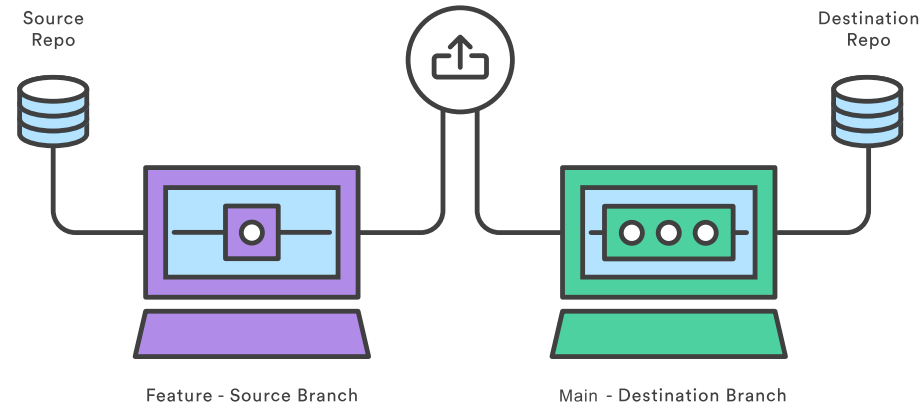
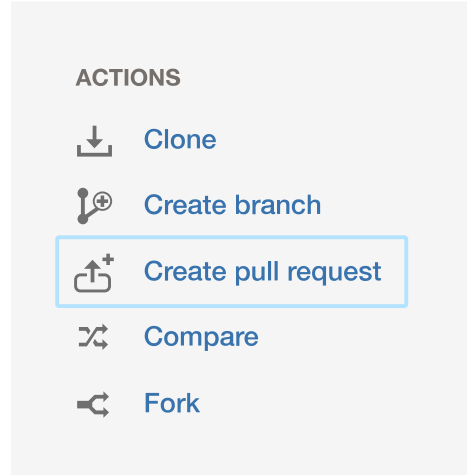
After Pushing



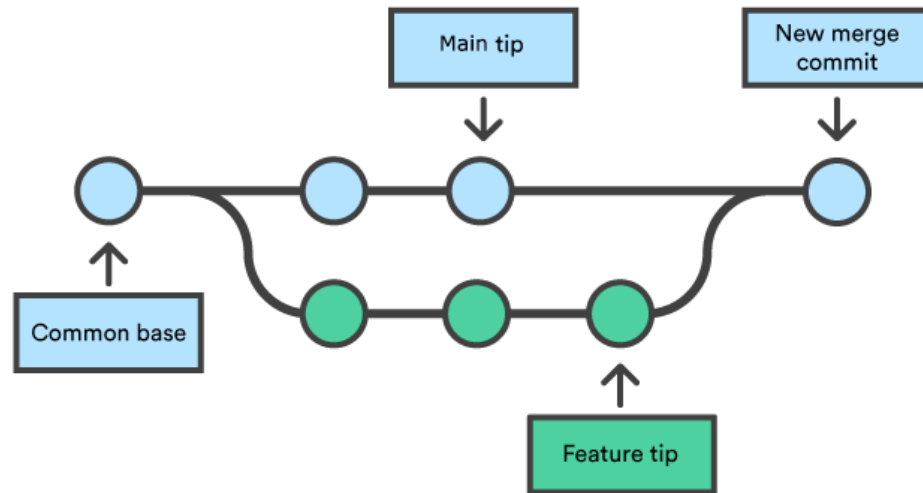
```
git push <remote> --tags
```

```
git push origin :branch_name
```

Is `git pull` similar to pull request?



git merge



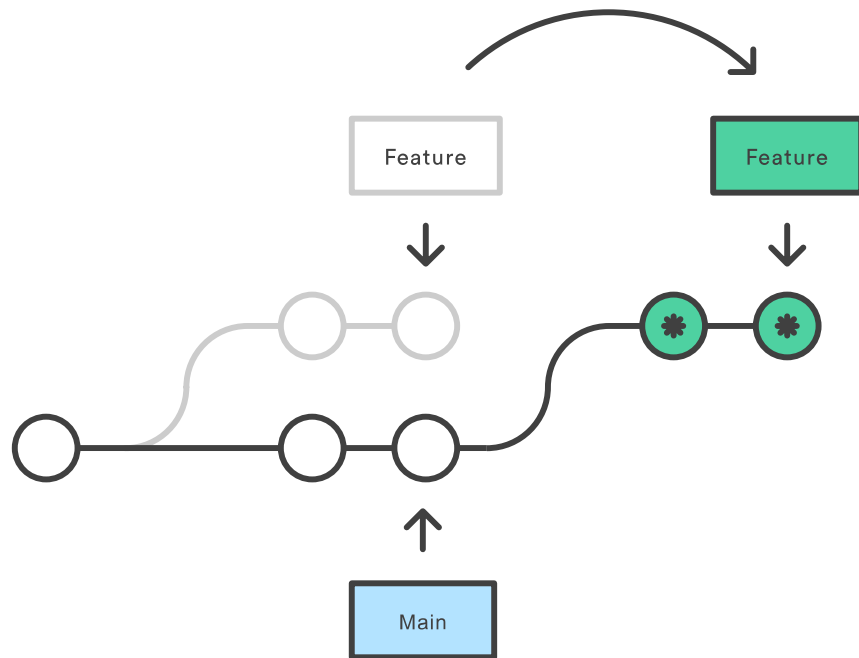
```
$ git add <file>
$ git commit -m "Make some super-stable changes to main"
$ # Merge in the new-feature branch
$ git merge new-feature
$ git branch -d new-feature
```

```
here is some content not affected by the conflict
<<<<<<< main
this is conflicted text from main
=====
this is conflicted text from feature branch
>>>>>> feature branch
```

Git Advanced Topics

Rewriting History

git rebase



* Brand New Commits

```
git rebase <base>
```

```
git rebase --onto <newbase> <oldbase>
```

git cheat sheet



Please Scan this to download your git cheat sheet or use the link [here](#)

Demo