POWERING POTENTIAL

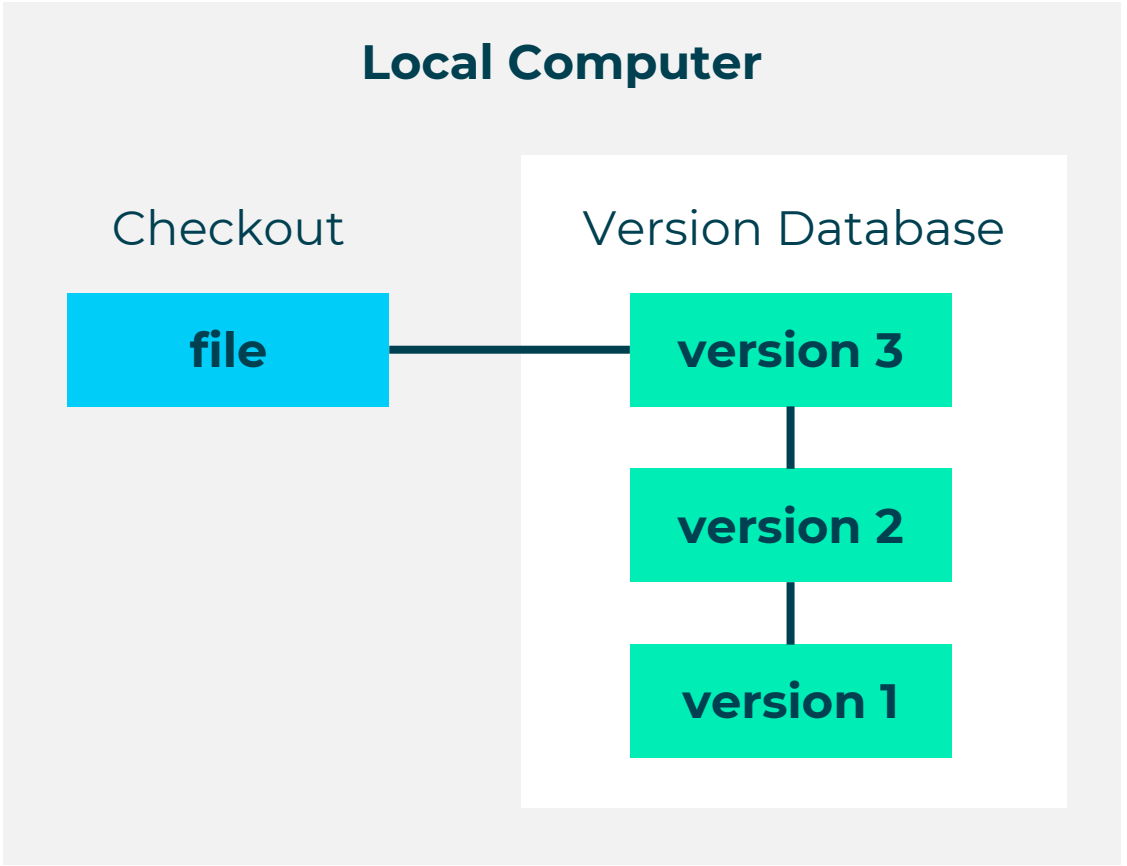
Version Control

What is version control?

Version control is the process of recording changes to files.

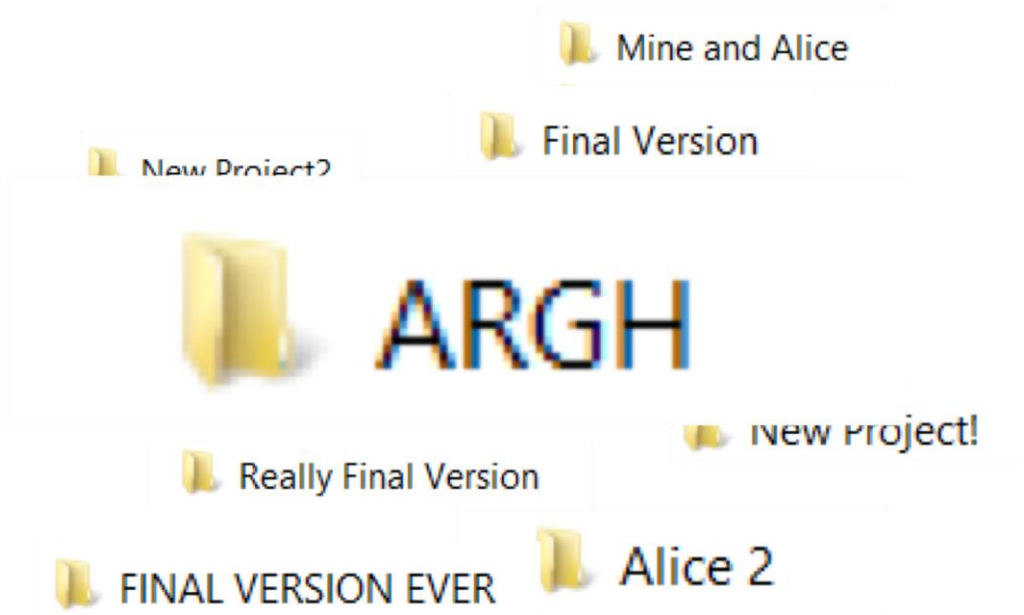
A Version Control System (VCS) allows you to manage file history, so you can:

- **Roll back** to previous states of a file if something goes wrong
- Maintain a **log of changes**
- **Compare** versioning issues



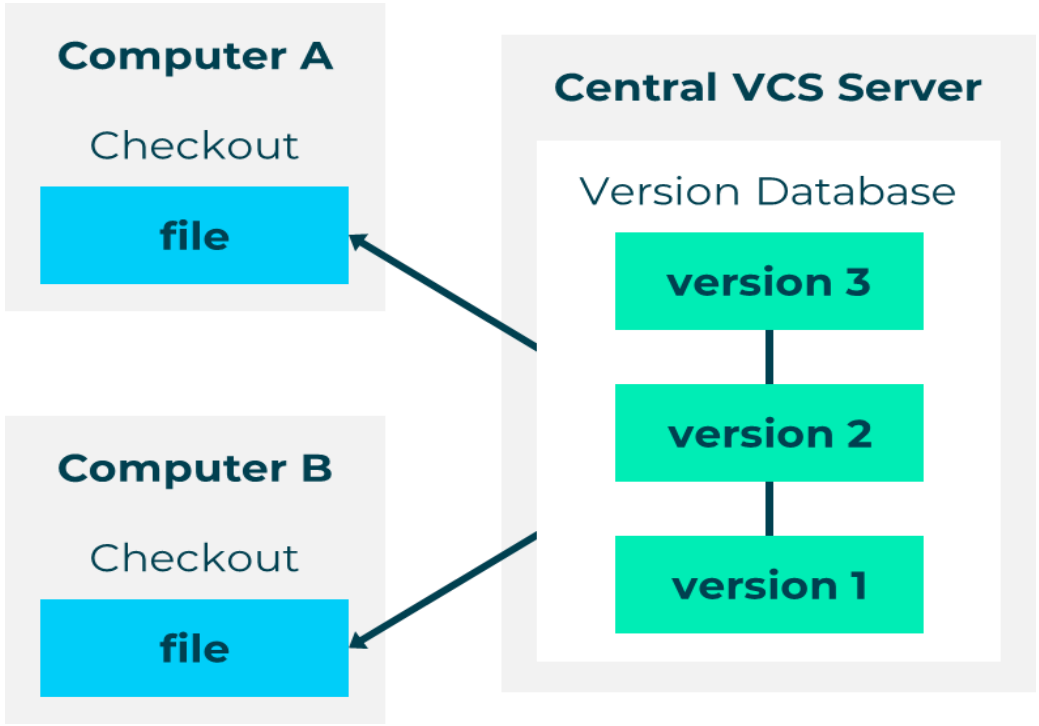
Benefits of version control

1. **Keep track of code and changes**
   - Automated version management
   - One copy of the code everyone can access
   - No more mailing around code or confusion trying to integrate it
2. **Multiple people can edit a single project at the same time**
   - Push changes to the central repository
   - Merge together changes in files where there are conflicts
3. **Branch code to work on specific parts**
   - Version 2.3 doesn't need to die because someone else wants to look at version 3

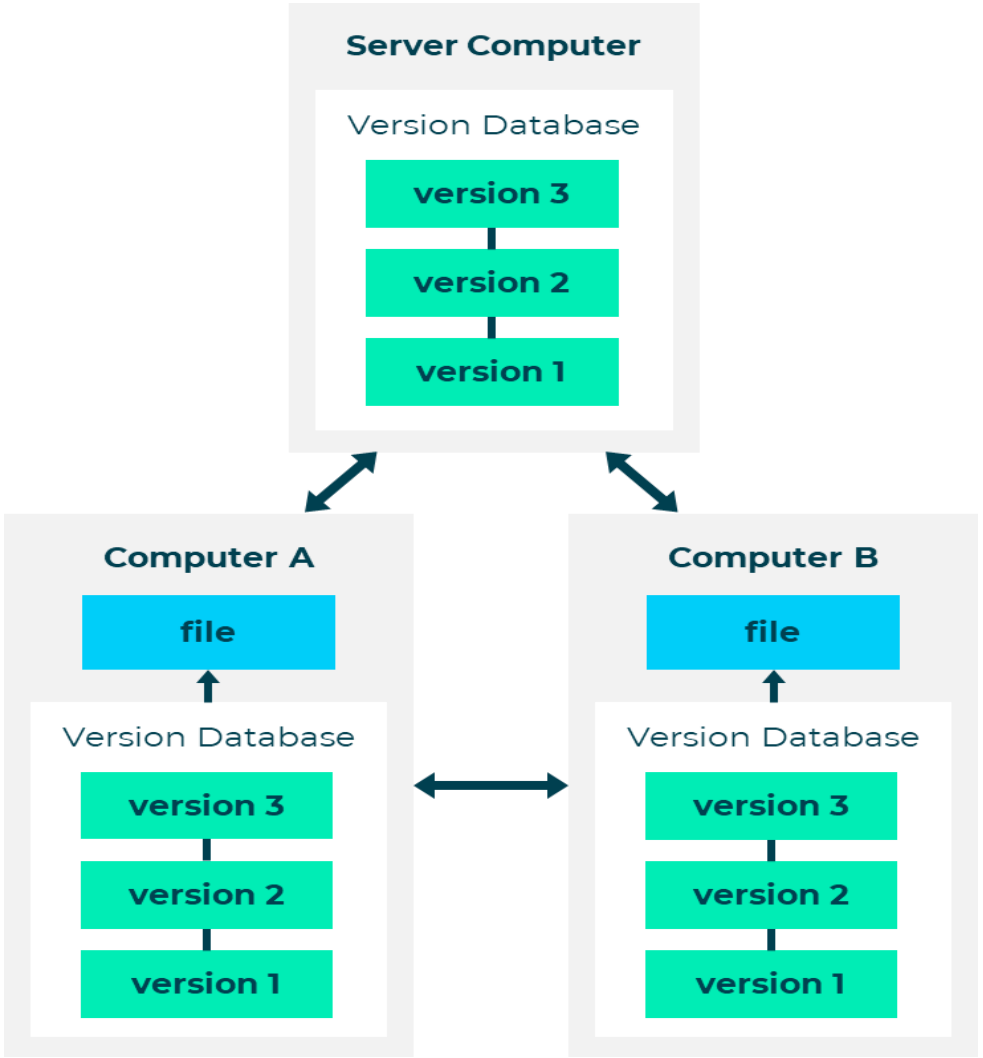


Types of Version Control Systems

Centralised Version Control System (CVCS)



Distributed Version Control System (DVCS)




GIT as a DVCS

GIT AS A DVCS

Git is a powerful version control tool.
Its origins are from Linux development, so it's **open source**.

Its goals are:

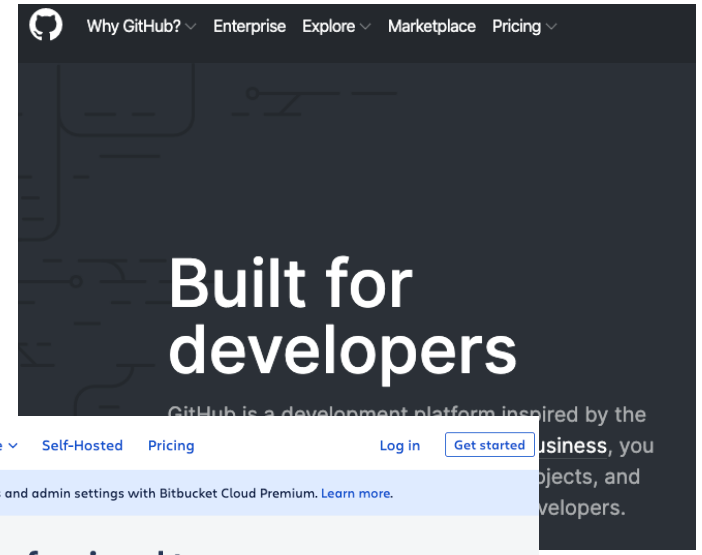
- Speed
- Simplicity
- Strong support for non-linear development
- Full distribution
- Ability to handle large projects efficiently, e.g. Linux kernel

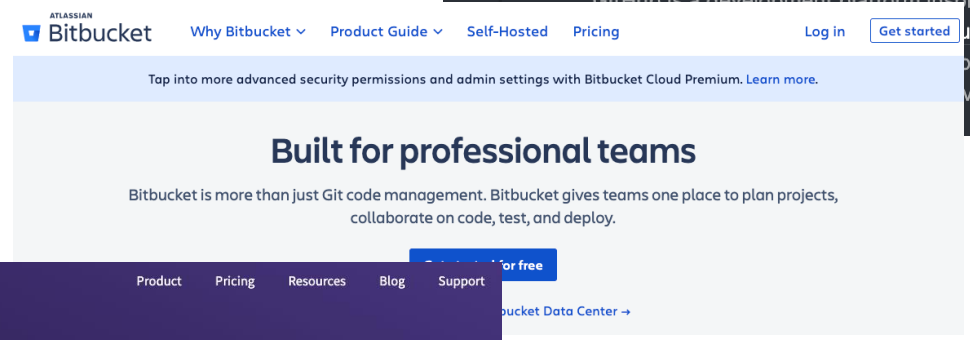
CHOOSING A HOSTING SERVICE FOR GIT

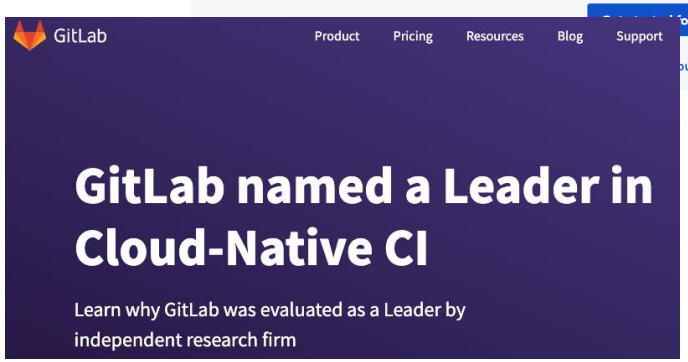
GitHub

BitBucket

GitLab







USING A HOSTING SERVICE FOR GIT

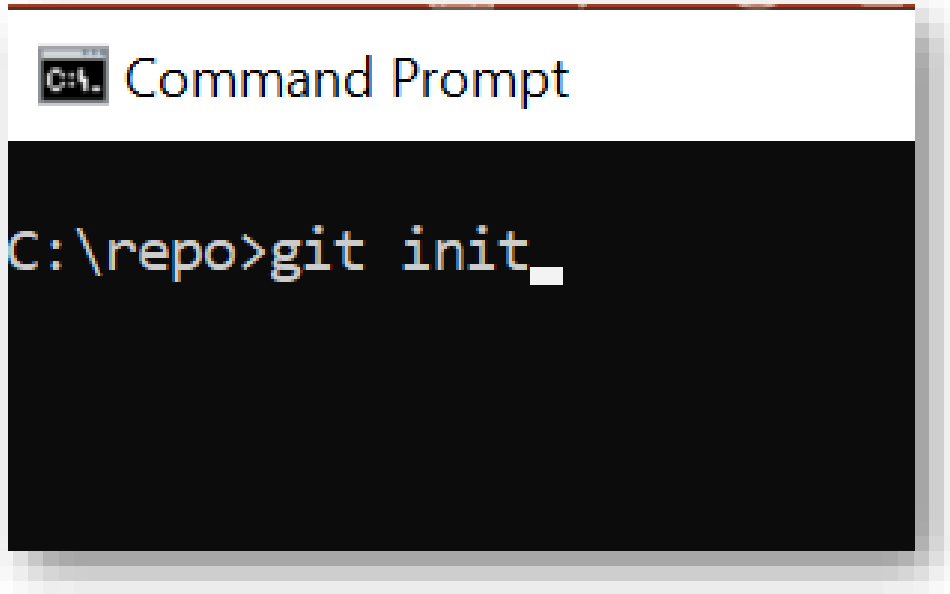
To make changes to a Git repository, follow these steps:

1. **Create** a repository on a hosting site, or your own server
2. **Check out** the repository to your own machine using **remote add**
3. **Add** some code
4. **Commit** your changes to the local repository
5. **Push** changes to the remote repository using **git push**

GIT Basics

INITIALISING REPOSITORIES

To create a new subdirectory and a Git repository skeleton, use:

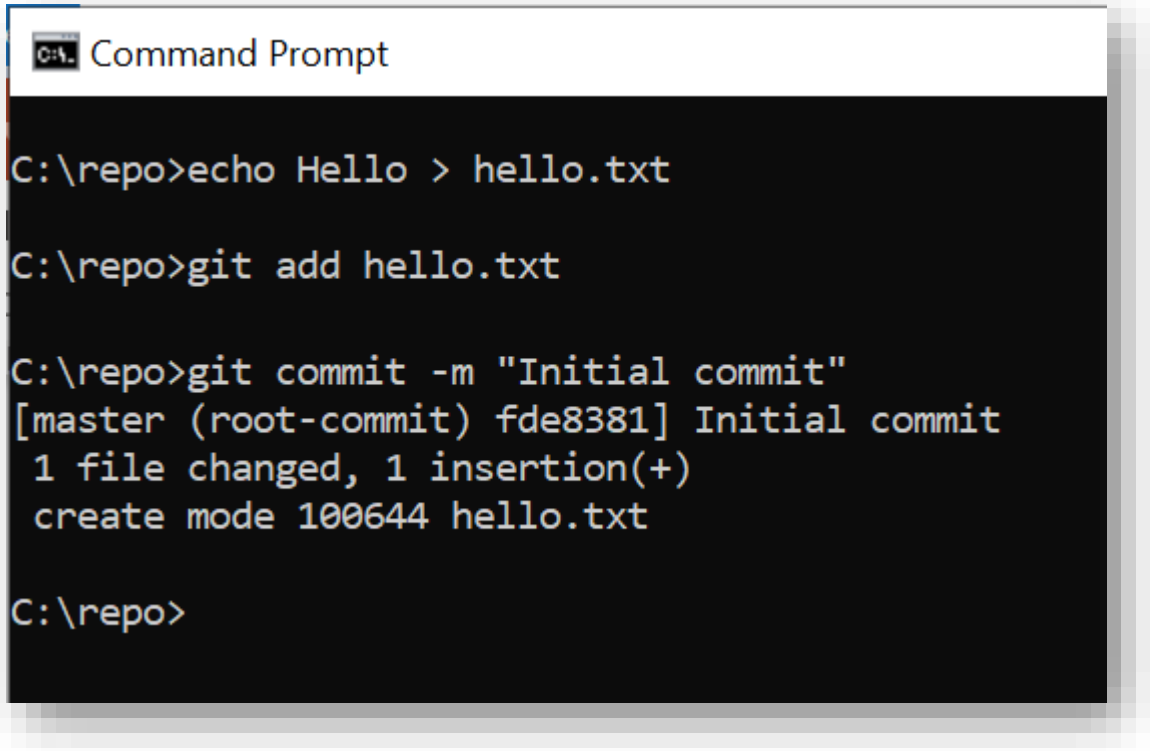


INITIALISING A REPOSITORY WITH EXISTING FILES

```
git add *.pp
```

```
git add README.md
```

```
git commit -m "Initial commit"
```



CLONING AN EXISTING REPOSITORY

Git can use a number of different protocols, including http and SSH:

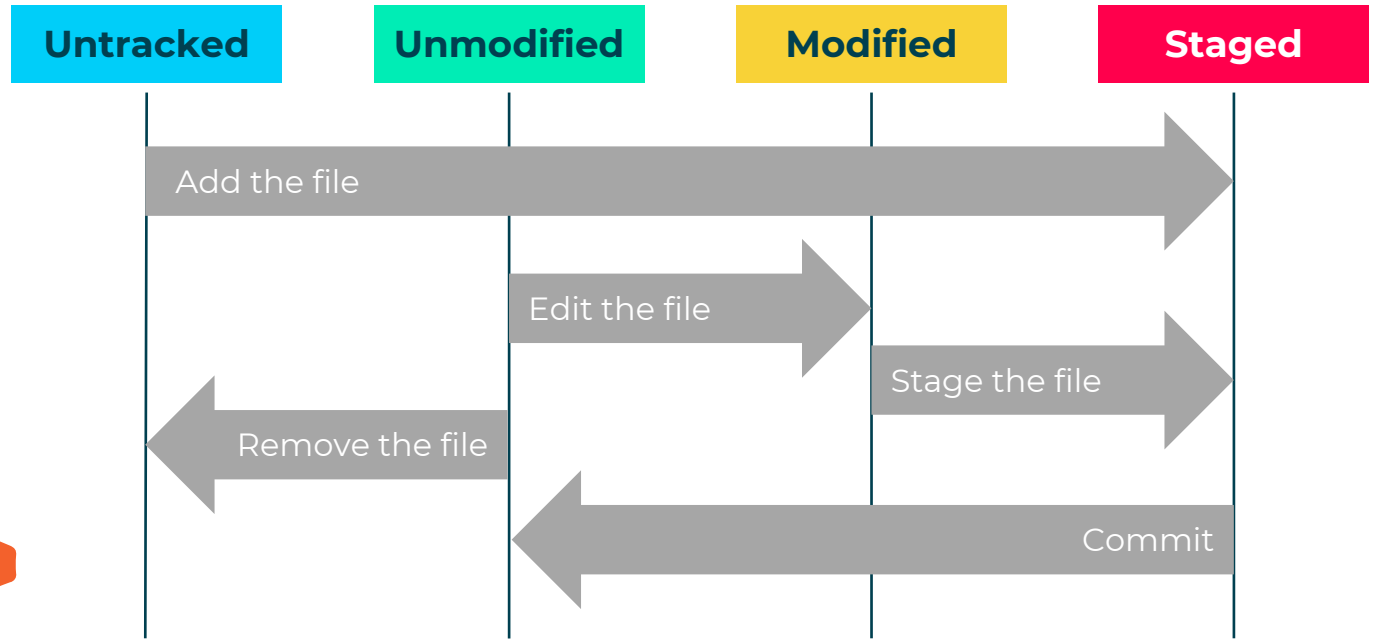
```
git  clone    git://github.com/resource
```

RECORDING CHANGES TO A REPOSITORY

Each file in a Git directory can be **tracked** or **untracked**.

1. Tracked files are files that were in the last snapshot. They can be unmodified, modified, or staged. Tracked files are also files that Git knows about.

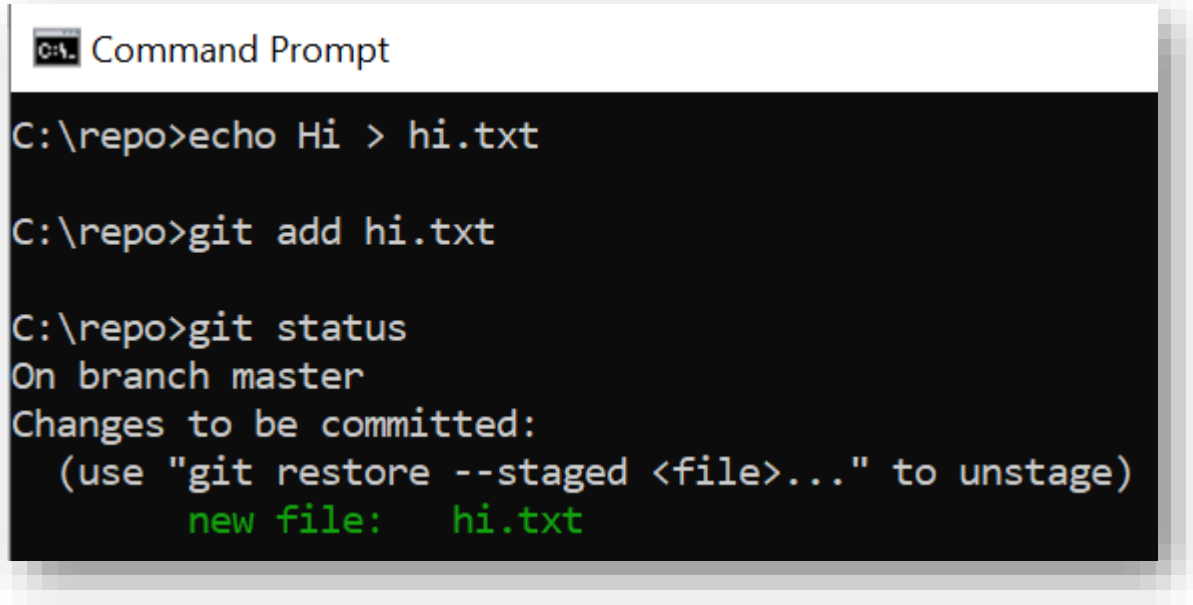
2. Untracked files are everything else. They're not in your last snapshot or staging area.



RECORDING CHANGES TO A REPOSITORY, CONT.

The main tool you use to determine which files are in which state is the git status command. If you run this command directly after a clone, you should see something like this:

```
git status
On branch master
Nothing to commit, working directory clean
```



STAGING NEW OR ‘MODIFIED’ FILES

To add a file, use:

```
git add <filename>
```

To tell Git to ignore files or folders, name them:

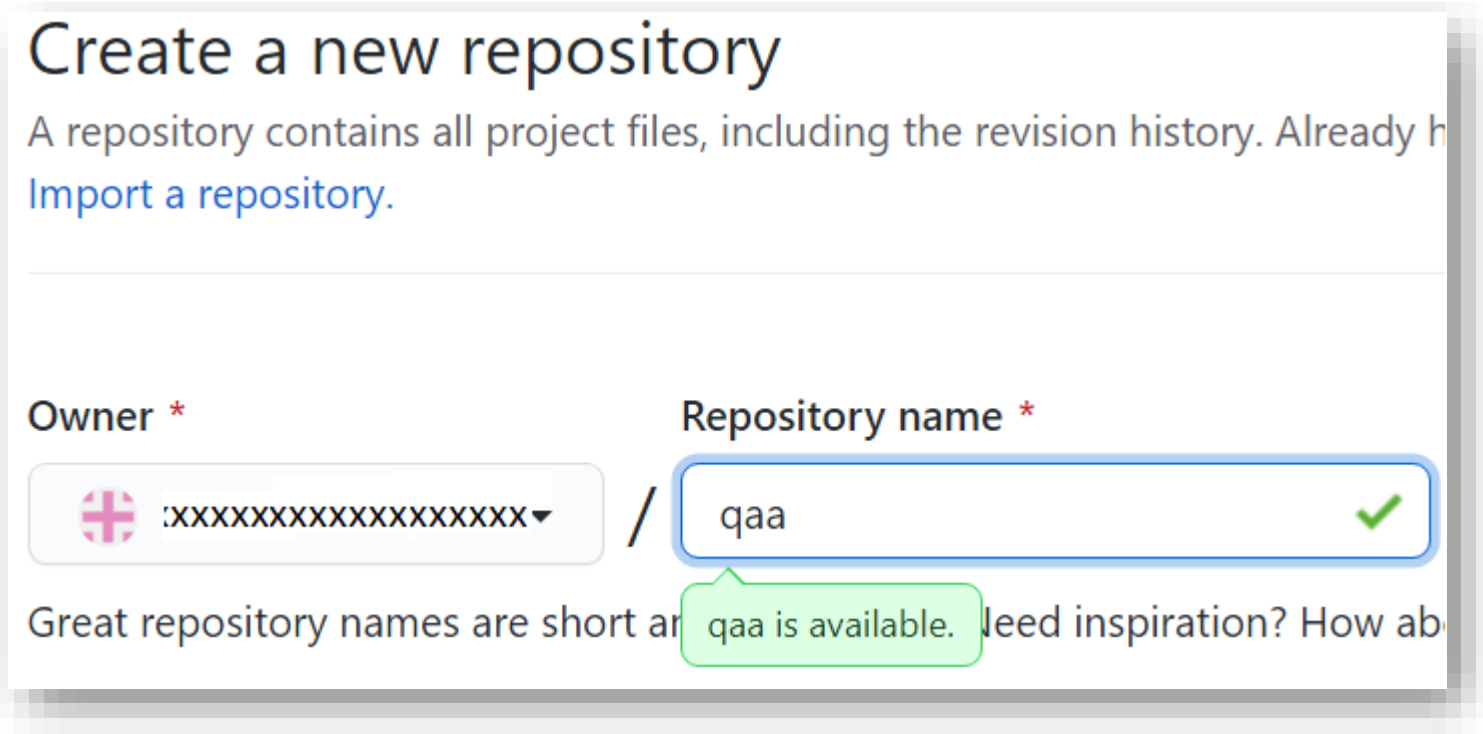
```
.gitignore
```

```
*.exe
*.dll
*.lib

.bin
```

WORKING WITH REMOTE REPOSITORIES

Remote repositories hold versions of a project or dependencies on the web / network such as GitHub.



WORKING WITH REMOTE REPOSITORIES

To see configured remote repositories, run the following command:

If you have cloned a repository you should see the origin.

To add a repository, use:

```
git remote   add    [shortname][url]
```

‘Shortname’ becomes an alias for access to the repository.

```
git remote add origin https://github.com/xxxxxxxxxxx/qaa.git
```

PUSHING TO A REPOSITORY

To **push** your project upstream, use:

```
git push origin master
```

-V shows you the URL that Git has stored for the shortname

```
C:\repo>git remote -v
qa      https://github.com/xxxxxxxxx/qaa.git (fetch)
```

To rename a reference:

git remote rename

e.g.

git remote rename origin qa

git push qa master

PUSHING TO A REPOSITORY, CONT.

To remove a reference, use **git remote rm**:

```
git  remote  rm   qa
git  remote  origin
```

```
C:\repo>git remote rm  qa
C:\repo>git remote -v

C:\repo>git remote add qaa
https://github.com/xxxxxx/qaa.git

C:\repo>git remote -v
qaa     https://github.com/ xxxxxx /qaa.git (fetch)
qaa     https://github.com/ xxxxxx /qaa.git (push)
```

PULLING FROM A REPOSITORY

To pull all the changes made to the repository, use:

```
git pull
```

Pull the repository before pushing changes

- You get an up-to-date copy of the repo to push to
- You can see any conflicts before they are pushed
- You can **stash** your changes before pulling the remote branch

```
C:\repo>git pull qaa master

From https://github.com/xxxxxx/qaa
 * branch            master     -> FETCH_HEAD
Already up to date.
```

CREATING A NEW BRANCH

To create a branch, use:

```
git  checkout  -b  newBranchName
```

To commit any changes to your code, use:

```
git  commit  -am  "updated some file(s)"
```

To merge a branch back into the main line, use:

```
git checkout master
```

```
git merge newBranchName
```

CREATING A NEW BRANCH ALTERNATIVE

Rather than use the **checkout** command shown above, you could use the following two commands:

```
git branch newBranchName

git checkout newBranchName
```

ALTERNATIVES TO GIT

Git is popular for the following reasons:

- Its open source nature
- Simplicity
- Context switching between branches is easier
- Local staging area for commits
- GUI tools available such as *Sourcetree*
- Built-in tools in eclipse

...but it isn't the only option. Alternatives include:

- Subversion
- CVS
- Mercurial
- Fossil
- Veracity
- SSH

‘Introduction to GIT’