POWERING POTENTIAL

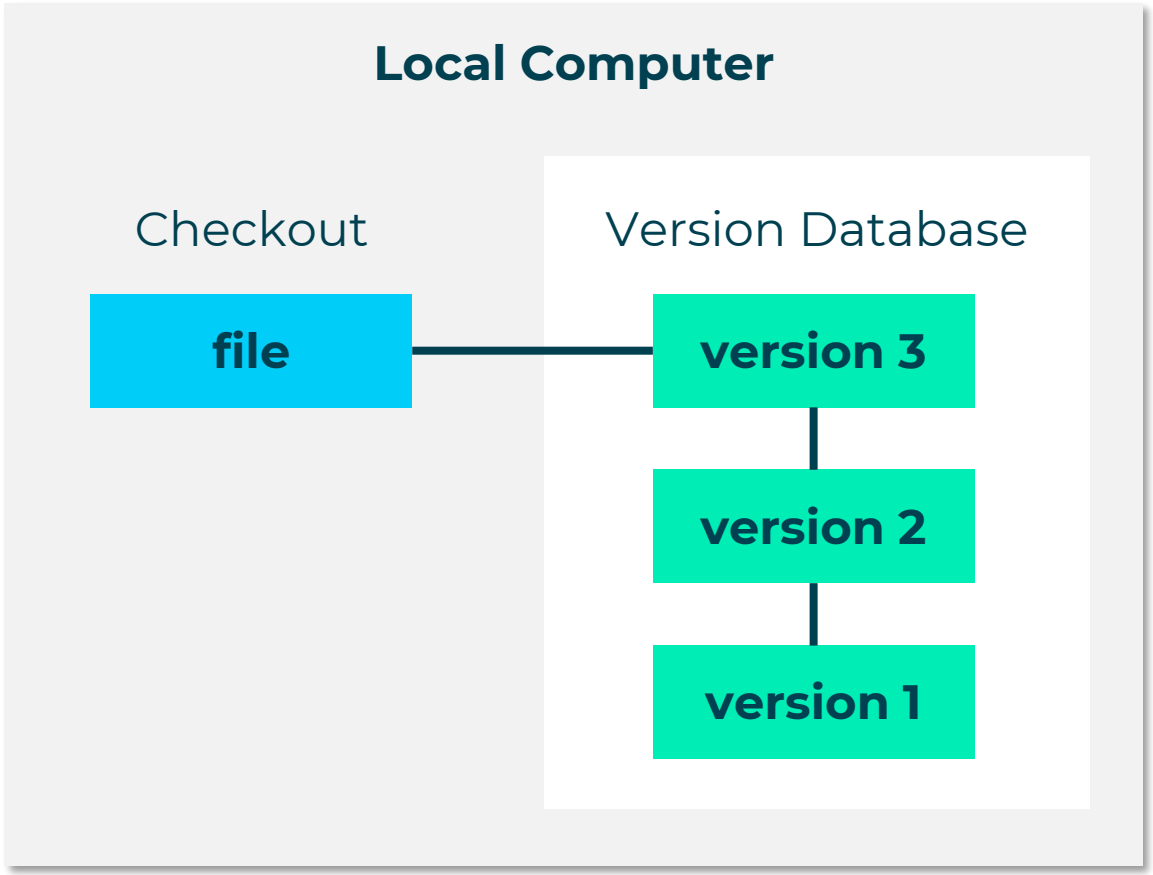
VERSION CONTROL

What is Version Control?

Version control is the process of recording changes to files.

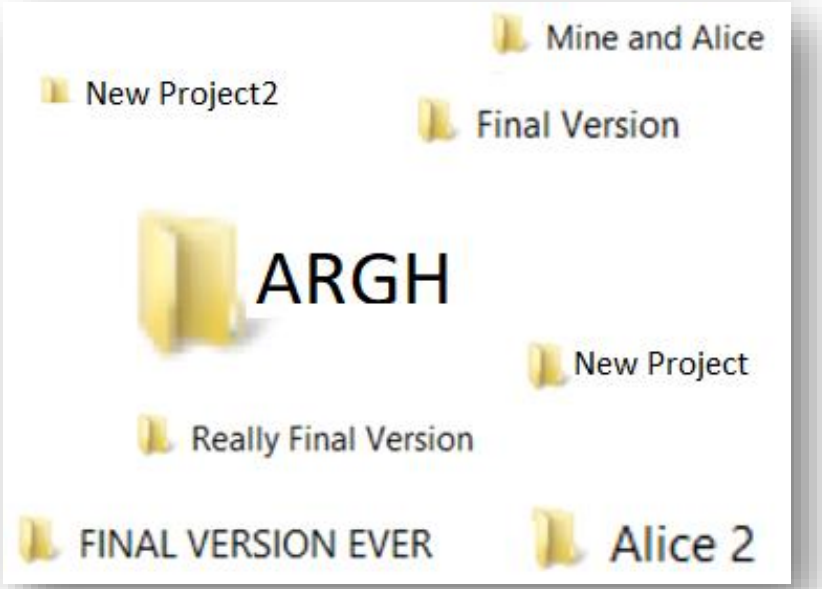
A version control system (VCS) allows you to manage file history, so you can:

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



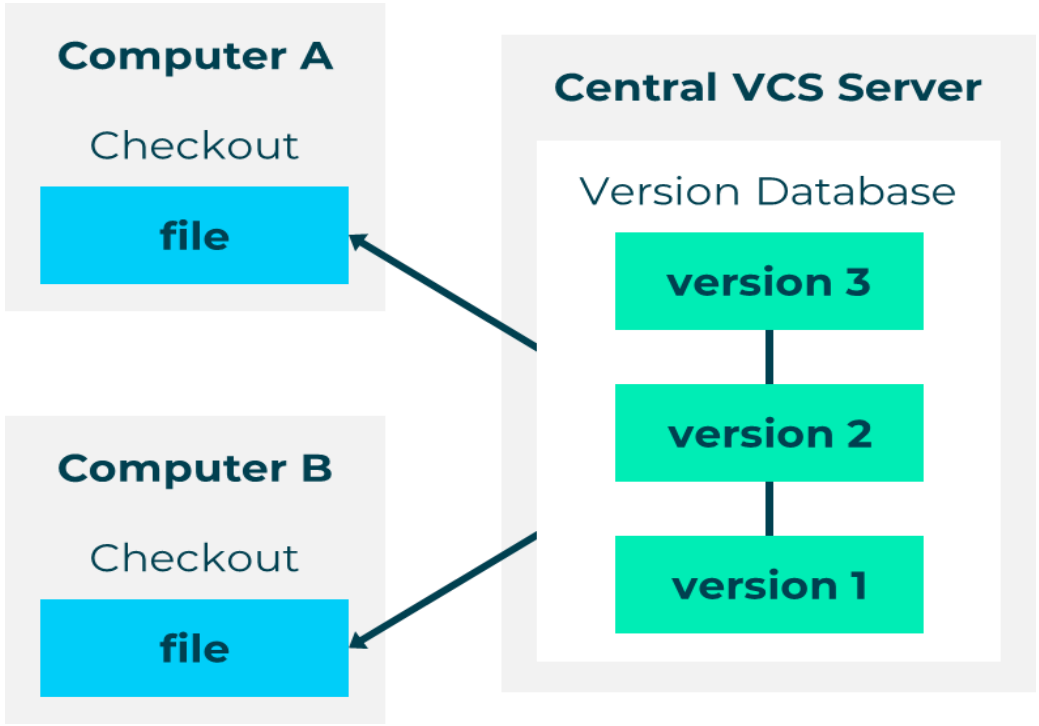
Benefits of Version Control

1. **Keep track of code and changes.**
   - Automated version management
   - One copy of the code everyone can has access
   - No more mailing around code and 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. **Create a Branch to work on specific parts.**
   - Version 2.3 doesn't need to die because someone else wants to look at version 3

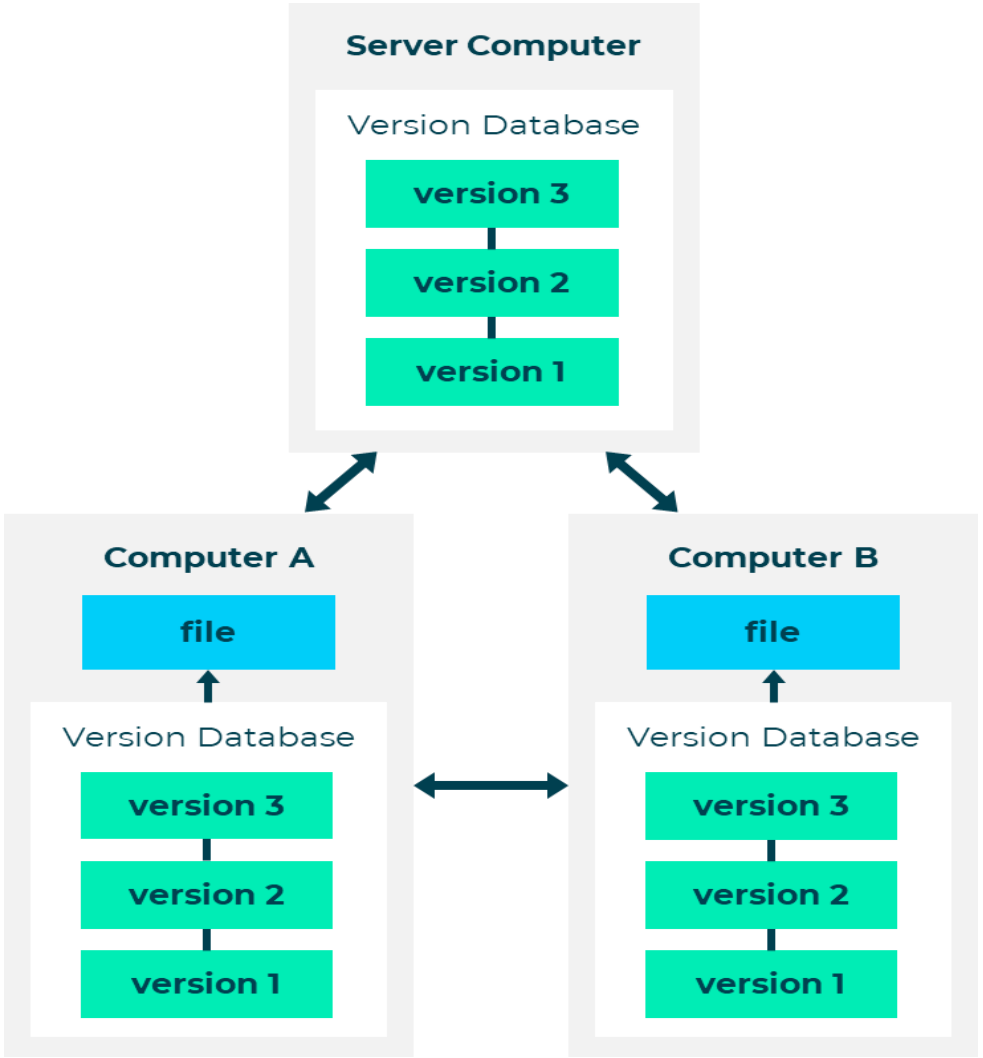


Types of Version Control Systems

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 in 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

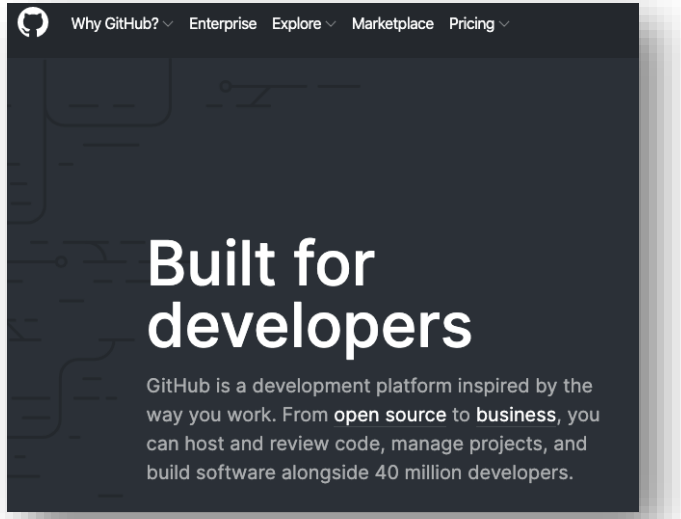
Linus Torvalds

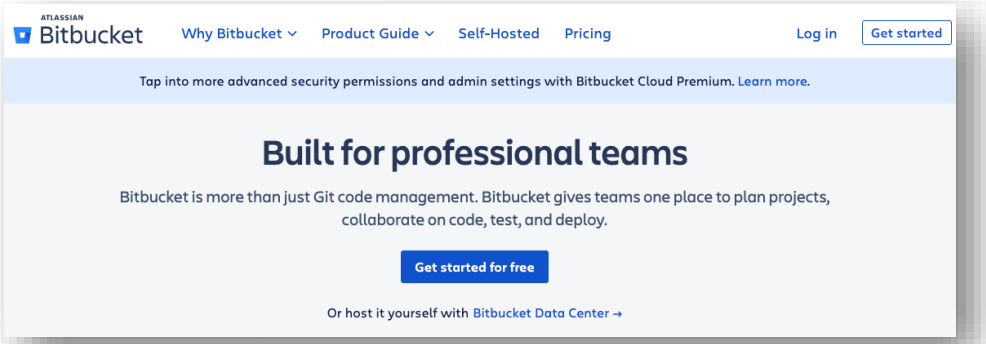
CHOOSING A HOSTING SERVICE FOR GIT

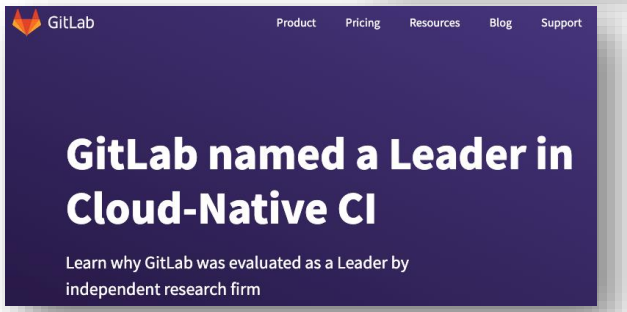
GitHub

BitBucket

GitLab







USING A HOSTING SERVICE FOR GIT

To make changes to a Git repository, do:

1. **Create** a repository on a hosting site, or 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

Getting started

Install GIT

https://git-scm.com/download/win
https://git-scm.com/download/linux
https://git-scm.com/download/linux

Configure GIT to identify yourself so others know the changes you make

```
git config  --global user.name  "Bob"
git config  --global user.email "bob@qa.com"
```

Generate keys for GitHub

Using the SSH protocol, you can connect and authenticate to remote servers and services. With SSH keys, you can connect to GitHub without supplying your username and personal access token at each visit. You can also use an SSH key to sign commits.

ssh-keygen -t rsa -b 4096 -C bob@qa.com

→ myKey, is the private key

→ myKey.pub is your public key

→ In GitHub add your Public key

→ **Settings > SSH and GPG keys and > New SSH key**
then paste the public key's text

C:\QA\gitDemos> **type .\mikeKey.pub**

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABQDQ00F3dL2NdejpQZaEfvhZhGP2Luoh4fv+MeBrPfekHO7yVHYp
4Xw7l77d7YY5BphtkZ3rExPsY5WP1OE3u6pGh6y9xdiX+Q+MIyWDTeDY+Prbt01dpvMXcGmYdLoEa
UoifTcNutR686p+OUAy2mMRYghcanTrRhlsbz5JCRnCnptHfZSwpucUaLxYkc/izwyofzEVb2rV4L6ik
2M6IKMbUZdKx+JqP1oh5/EPH8iZK+OOWcPJ/fnT/GLE3xnncwCSzV2ji6t+oML/Bq8R30+qgv8kV0EV
qLGrqdwGiSmeGKFu4fVLAjEdiPwGIlMWes+Hc6sxmdVo1RkuMAuRl7AgbPzsyfKTAPHK5YJww81w
apZ61nO3rmTI3yeVl8wIHWUdj3K6Ua+ck005GLZidfxtsGkuniMTnvHXMYhNgxQyG5TunElkxnO9n4n
GqATC/nGaivTlwTQ3gjeZGp3v4XQaz90cHxfF0bD1YJ+LGhQNLkZxVsN7xQgl8JY3++6YqWkUaPen6
J0CLUW5Uq4f+V89QlZDLH6H4UVEqmF3Ey6Zv5wkEtHbepCKWPMfHTU6wd6zAywDP4ZhnzHk6F
uSYVU4UEovZ5tGet0qipltXagttOB9kt1AETkNUgI9MAXXTLdBPQhgW4Ch+zrHK1SkZchQuSRsQfyT
Ozu0O37CD1hk4264jjNQZAO== mik5e@hotmail.com

CLONING A REPO

Cloning a Repository (git clone)

To download a remote repository, use the **git clone** command and provide the URL of the remote repository.

- It configures the local repository for you
- and the remote repository is configured for when you need to push your new changes to it.

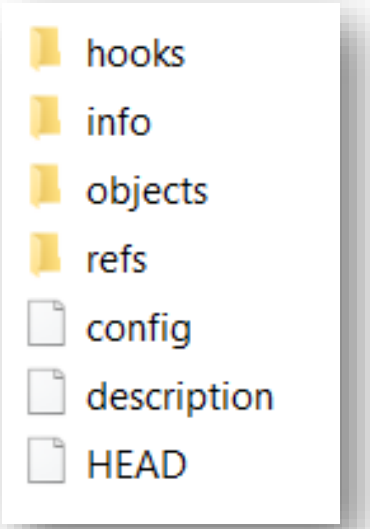
git clone git@github.com:bob-Smith/qa.git

INITIALISING REPOSITORIES

To create a new subdirectory and a Git repository skeleton

```
C:\Demos>git init

Initialized empty Git repository in C:/Demos/.git/
```



Git uses this structure to keep track of your changes

You do not work directly with the **.git** folder

INITIALISING A REPOSITORY WITH EXISTING FILES

```
git add *.txt
```

```
git add README.md
```

```
git  add  .
```

The git add command record changes to the **staging area**.

It is the way you tell Git to include updates when you commit them

Staging area is what you propose for the next commit
It is like what happens in deploying files in a dev environment

To remove a file/s from the staging area type:

```
git rm --cached hello.txt
```

STAGING NEW OR ‘MODIFIED’ FILES

To tell Git to ignore files or folders, name them in **.gitignore** file

```
*.exe
*.dll
*.lib
.bin
node_modules
```

COMMITTING CHANGES

```
git commit –m "a short message of changes"
```

So, you use **git add** to **stage** files

Use the **commit** command to commit the changes

A few notes about commit:

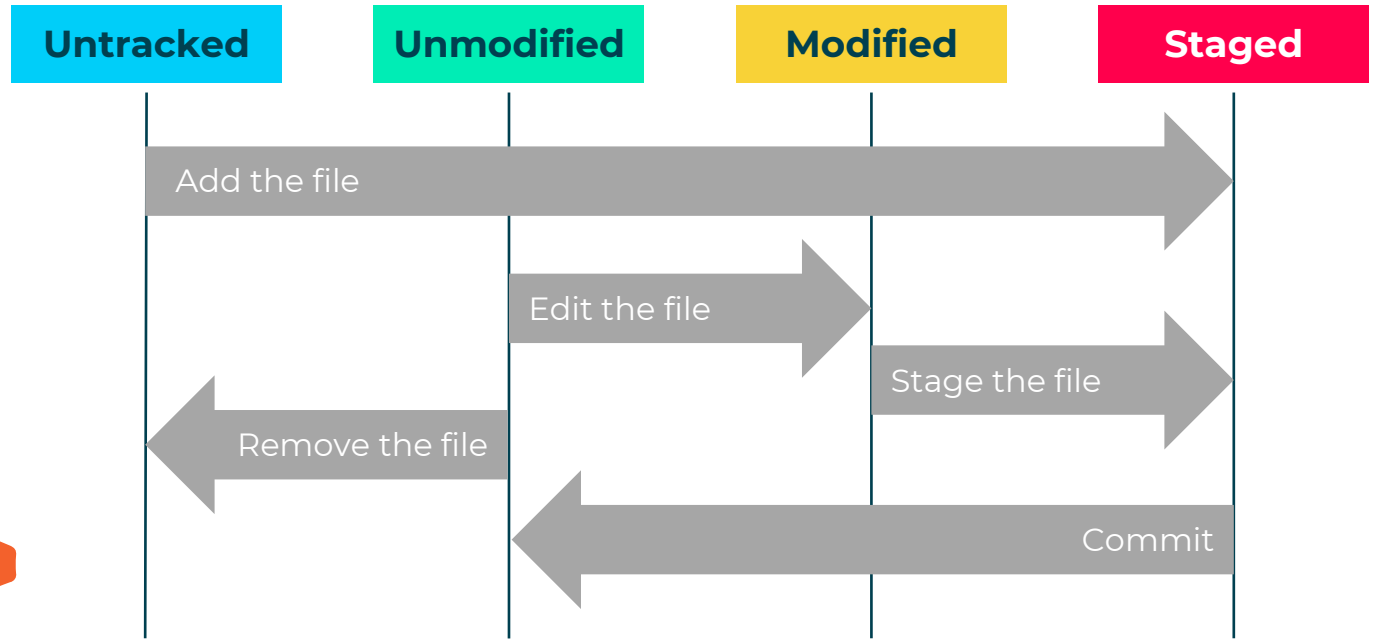
- Don't make micro commits! Wait until you've made substantial change, but don't wait a week/month either!
- It does not empty the staging area. So, if you delete a file in the working directory, you must **git add** it!
- When you **change** a file, the old contents are still in the *staging area and you must* add it again!
- If you **delete** a file/s you must still git commit the changes
- Same goes for **renaming** a file.
- git does not just keep deltas. It copied files for speed.

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. Are files that Git knows about.

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



GIT STATUS COMMAND

The main tool you use to determine which files are in which state is the **git status** command.

```
C:\demo>git add .

C:\demo>git status
On branch master

No commits yet

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

```
C:\demo>git commit -m "first commit"
[master (root-commit) c0b399b] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt

C:\demo>git status
On branch master
nothing to commit, working tree clean
```

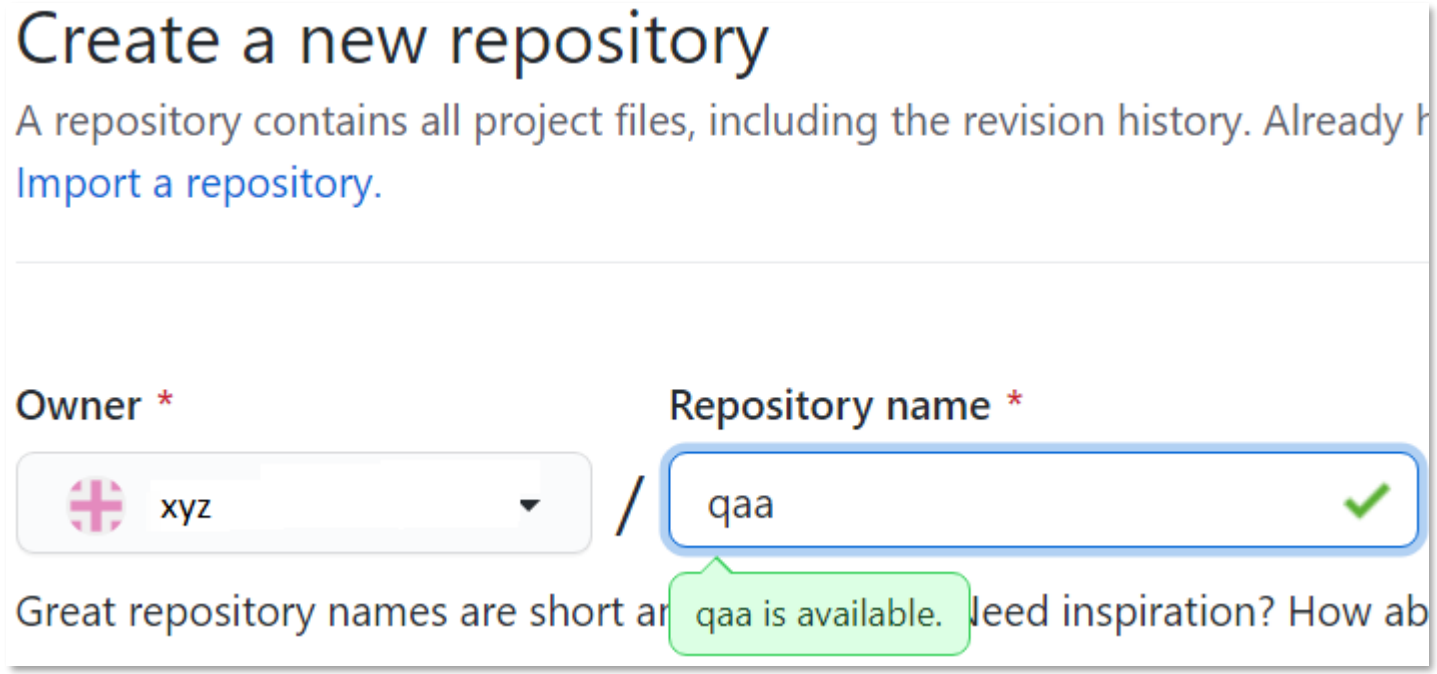
CLONING AN EXISTING REPOSITORY

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

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

WORKING WITH REMOTE REPOSITORIES

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



WORKING WITH REMOTE REPOSITORIES

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/xyz/qaa.git
```

```
C:\demo>git pull origin master
```

PUSHING TO A REPOSITORY

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

```
git push origin master
```

-v shows you the URL that Git has stored for *shortname*

```
C:\demos>git remote –v
origin  https://github.com/mikebaradaran/gitDemo2.git (fetch)
origin  https://github.com/mikebaradaran/gitDemo2.git (push)
```

You can also rename/remove the reference
(see notes below)

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:\demos>git  pull  origin  master

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

FORKING A REPOSITORY

Fork creates a copy of an existing repo under your account.

- It allows you to freely experiment with the existing code base, without breaking the project. You can then contribute by adding additional functionality.

Proposing a change scenario

Imagine you are using someone's project and you find a bug. Usually, you would raise an issue for this; however, forking means you would be able to attempt a fix yourself.

The steps would be:

- Fork the repository
- Create a new branch to fix the issue
- Submit a pull request to the owner of the original project
- If the owner approves your fix, your work can be merged into the original repository.

CREATING A NEW BRANCH

To create a branch, use:

```
git checkout -b newBranchName
```

To add & commit any changes :

```
git commit -am "update server error message"
```

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

```
git checkout master
```

```
git merge newBranchName
```

OTHER USEFUL GIT COMMANDS GIT DIFF

Find differences between the current file/s3 and index

```
git  diff

Git diff script.js
```

```
diff --git a/script.js b/script.js
index 65c652f..f7f74f9 100644
--- a/script.js
+++ b/script.js
@@ -1 +1,4 @@
-alert("Hi there!");
\ No newline at end of file
+alert("Hi there!");
+alert("added line");
```

CHECKING OUT A VERSION

You can check out any version from the history

```
C:\demos>git log --oneline

2f2977a (HEAD -> newBranch) Change script for a new message
3be3a2a Hello there message changed
bf5d588 (origin/master, master) fourth
eb62fe9 Added a javascript file
afca5dd First commit
```

```
C:\demos>git checkout afca5dd
```

Back to master again

```
C:\demos>git checkout master
```

ALTERNATIVES TO GIT

Git is popular due to:

- Open source nature
- Simplicity
- Context switching between branches 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

Experiment with GIT