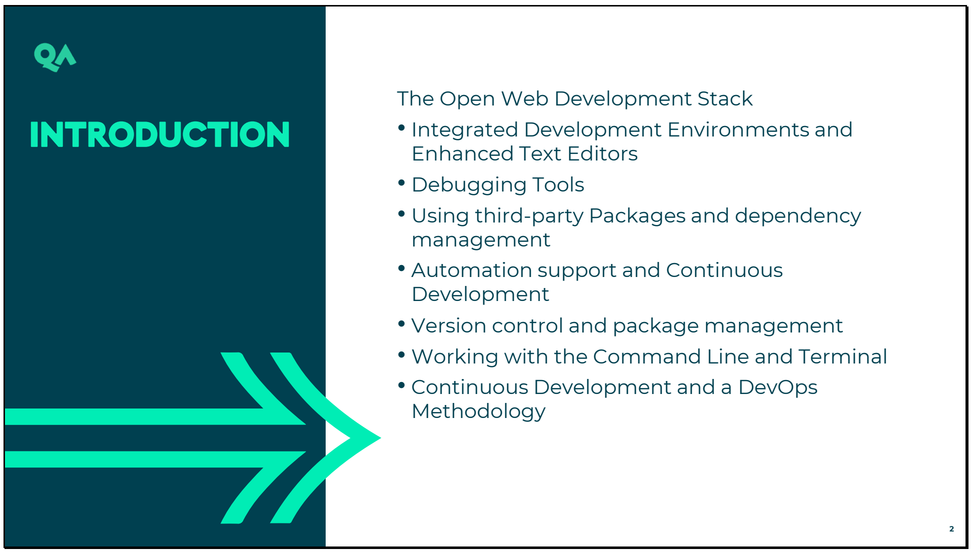
Slide 1

# WEB DEVELOPMENT TOOLS

JavaScript fundamentals

POWERING POTENTIAL

Slide 2



## INTRODUCTION

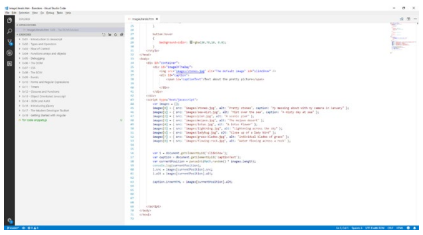
The Open Web Development Stack

- Integrated Development Environments and Enhanced Text Editors
- Debugging Tools
- Using third-party Packages and dependency management
- Automation support and Continuous Development
- Version control and package management
- Working with the Command Line and Terminal
- Continuous Development and a DevOps Methodology

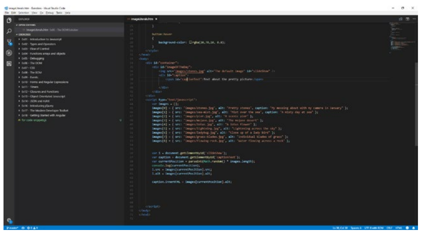
Slide 3

# Walkthrough – using Visual Studio Code

- Visual Studio Code is a free, open source enhanced text editor and it was built using JavaScript!
- Not to be confused with Visual Studio (a paid-for, full IDE designed specifically for .NET development)



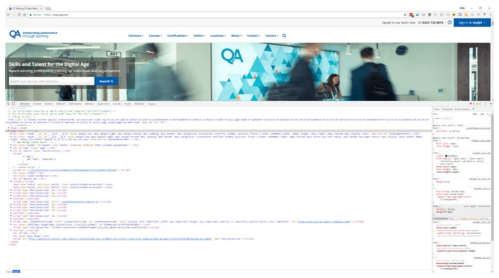
```
        }

        button:hover
        {
            background-color: rgba(20,70,10, 0.8);
        }
    </style>
</head>
<body>
    <div id="container">
        <div id="imageOfTheDay">
            <img src="images/stones.jpg" alt="The default image" id="slideShow" />
            <div id="caption">
                <span id="captionText">Text about the pretty picture</span>

            </div>
        </div>
    </div>
    <script type="text/javascript">
        var images = [];
        images[0] = { src: "images/stones.jpg", alt: "Pretty stones", caption: "My messing about with my camera in January" };
        images[1] = { src: "images/sea-mist.jpg", alt: "Mist over the sea", caption: "A misty day at sea" };
        images[2] = { src: "images/pier.jpg", alt: "A scenic pier" };
        images[3] = { src: "images/mojave.jpg", alt: "The mojave desert" };
        images[4] = { src: "images/lotus.jpg", alt: "A lotus flower" };
        images[5] = { src: "images/lightning.jpg", alt: "Lightening across the sky" };
        images[6] = { src: "images/ladybug.jpg", alt: "Close up of a lady bird" };
        images[7] = { src: "images/grass-blades.jpg", alt: "Individual blades of grass" };
        images[8] = { src: "images/flowing-rock.jpg", alt: "Water flowing across a rock" };


        var i = document.getElementById('slideShow');
        var caption = document.getElementById('captionText');
        var currentPosition = parseInt(Math.random() * images.length);
        console.log(currentPosition);
        i.src = images[currentPosition].src;
        i.alt = images[currentPosition].alt;

        caption.innerHTML = images[currentPosition].alt;




    </script>
</body>
</html>
```



Slide 4

## Walkthrough – Chrome development tools

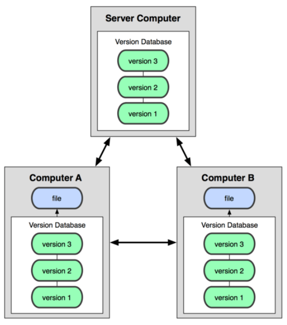
- Most browsers have development and debugging tools
- Incredibly useful for testing and management


Speak to our team now UK 0333 130 8816
Sign in to myQA
QA transforming performance through learning
Solutions
Courses
Certifications
Online
Locations
About
Contact
Careers
Skills and Talent for the Digital Age
Award-winning professional training for individuals and organisations
Search our courses and solutions...
Search

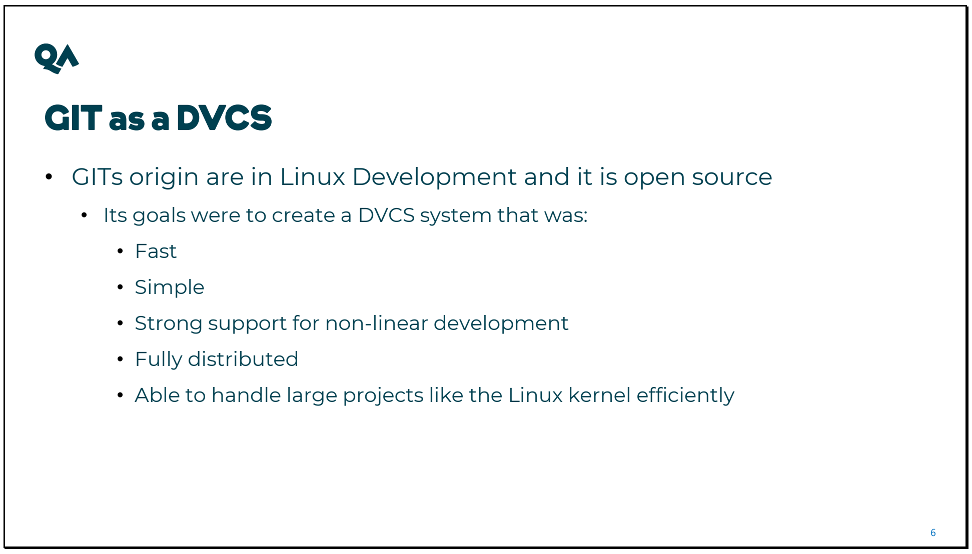
Slide 5

## Distributed version control systems

- DVCS do not just check out the most local snapshot of a file
  - They mirror the repository
  - so each checkout is like a backup of the repository



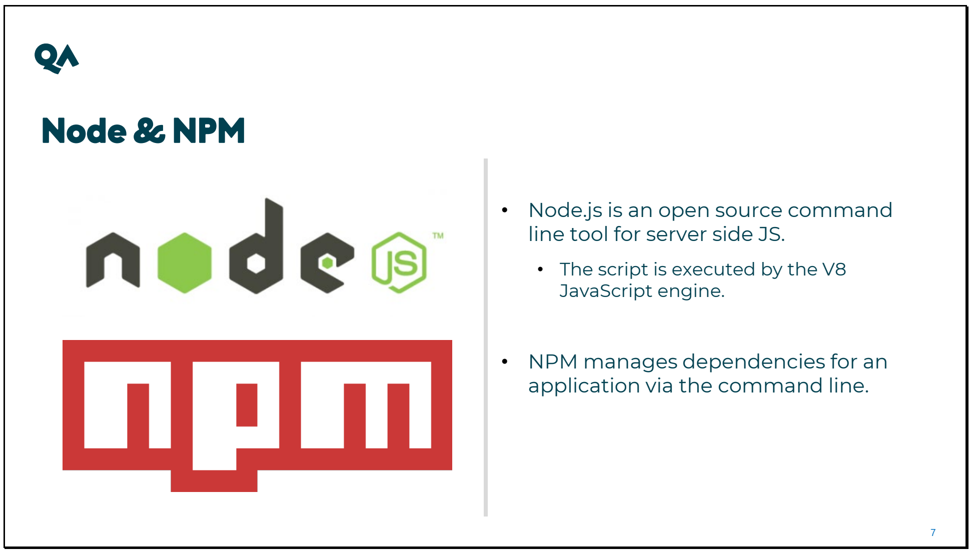
Slide 6



## GIT as a DVCS

- GITs origin are in Linux Development and it is open source
  - Its goals were to create a DVCS system that was:
    - Fast
    - Simple
    - Strong support for non-linear development
    - Fully distributed
    - Able to handle large projects like the Linux kernel efficiently

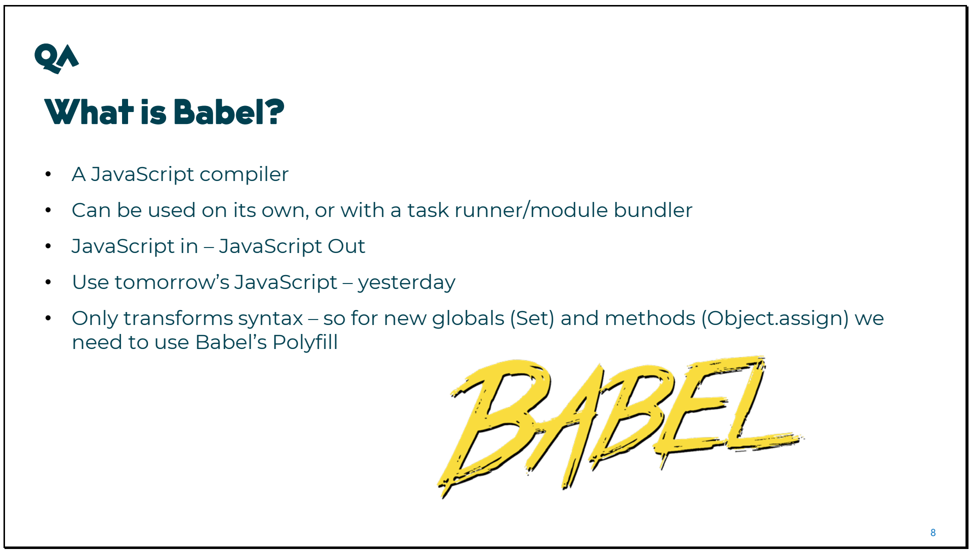
Slide 7



## Node & NPM

- Node.js is an open source command line tool for server side JS.
  - The script is executed by the V8 JavaScript engine.
- NPM manages dependencies for an application via the command line.

Slide 8



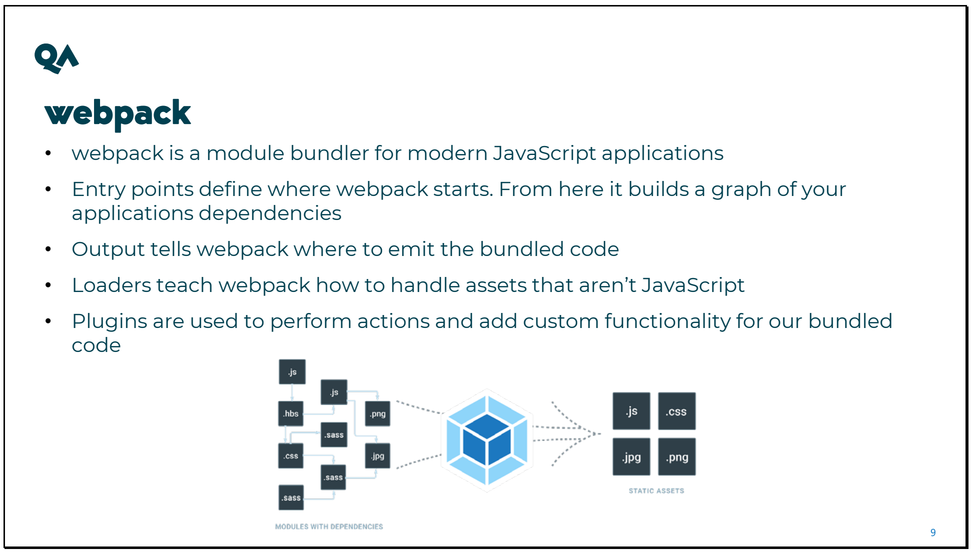
## What is Babel?

- A JavaScript compiler
- Can be used on its own, or with a task runner/module bundler
- JavaScript in – JavaScript Out
- Use tomorrow's JavaScript – yesterday
- Only transforms syntax – so for new globals (Set) and methods (Object.assign) we need to use Babel's Polyfill

8

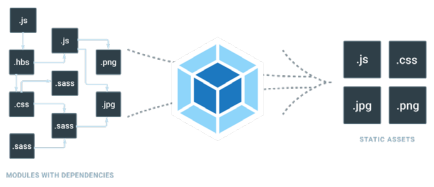
Babel transpiles ES2015+ Syntax back to ES3+ syntax and they have created a polyfill to create the new globals and methods found in ES2015+.  This means that tomorrow's JavaScript works in yesterday's browsers!

Slide 9

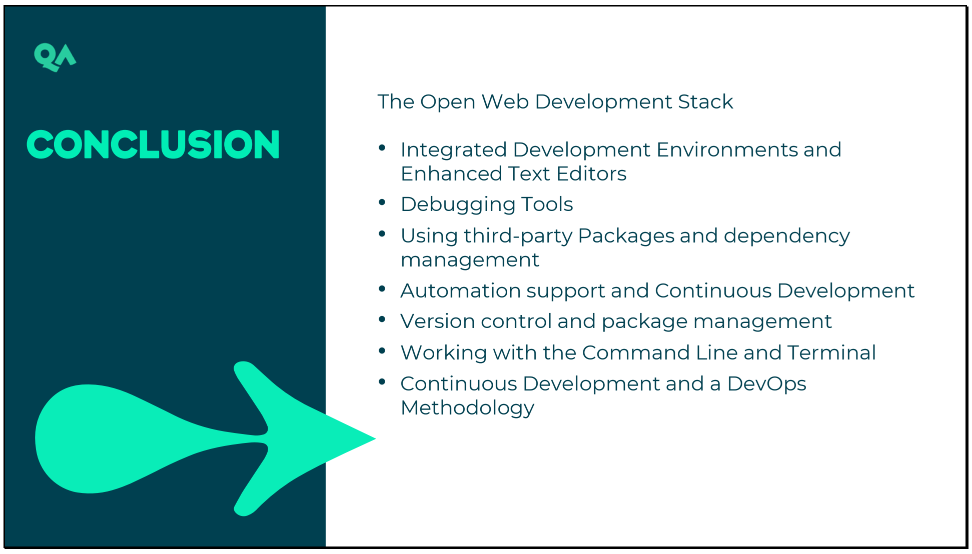


## webpack

- webpack is a module bundler for modern JavaScript applications
- Entry points define where webpack starts. From here it builds a graph of your applications dependencies
- Output tells webpack where to emit the bundled code
- Loaders teach webpack how to handle assets that aren't JavaScript
- Plugins are used to perform actions and add custom functionality for our bundled code



Slide 10

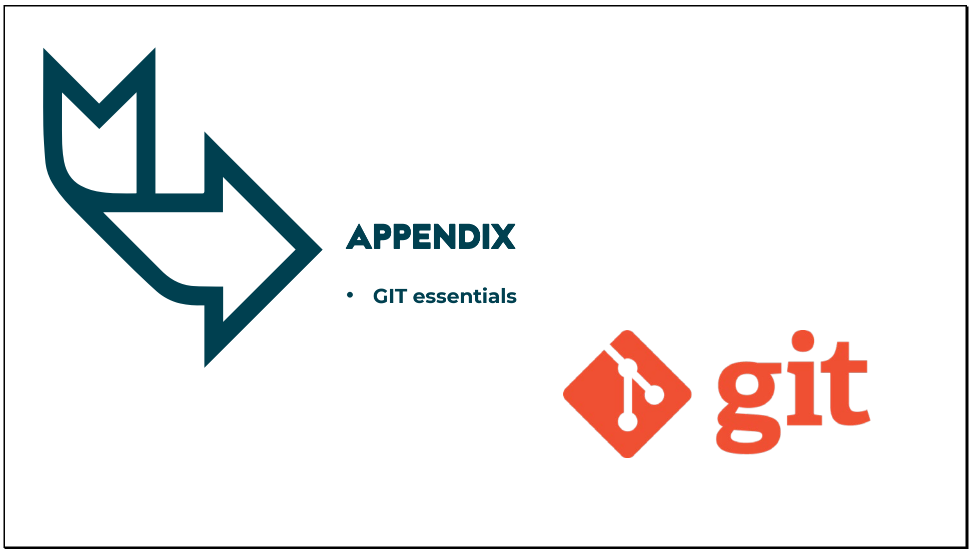


## CONCLUSION

The Open Web Development Stack

- Integrated Development Environments and Enhanced Text Editors
- Debugging Tools
- Using third-party Packages and dependency management
- Automation support and Continuous Development
- Version control and package management
- Working with the Command Line and Terminal
- Continuous Development and a DevOps Methodology

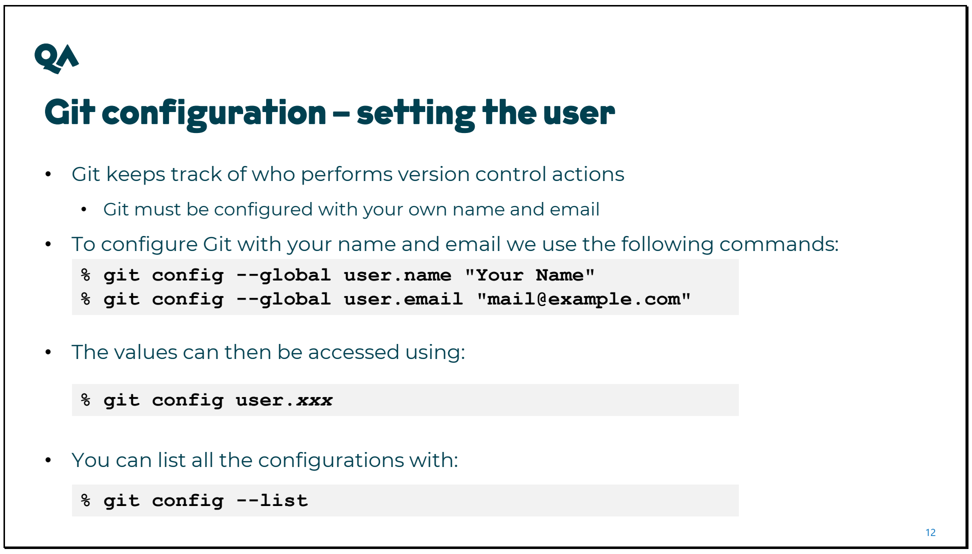
Slide 11



## APPENDIX

- GIT essentials

Slide 12



## Git configuration – setting the user

- Git keeps track of who performs version control actions
  - Git must be configured with your own name and email
- To configure Git with your name and email we use the following commands:

```
% git config --global user.name "Your Name"
% git config --global user.email "mail@example.com"
```

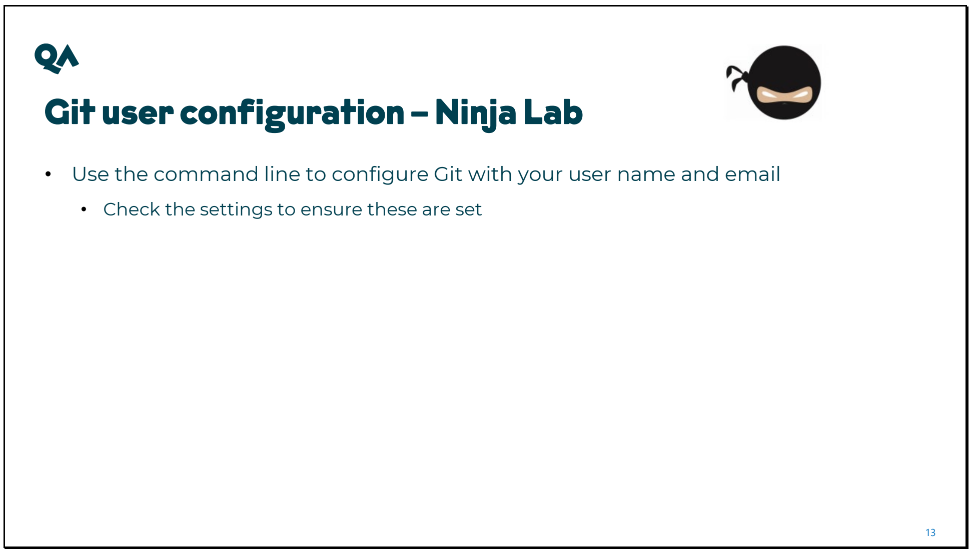
- The values can then be accessed using:

```
% git config user.xxx
```

- You can list all the configurations with:

```
% git config --list
```

Slide 13

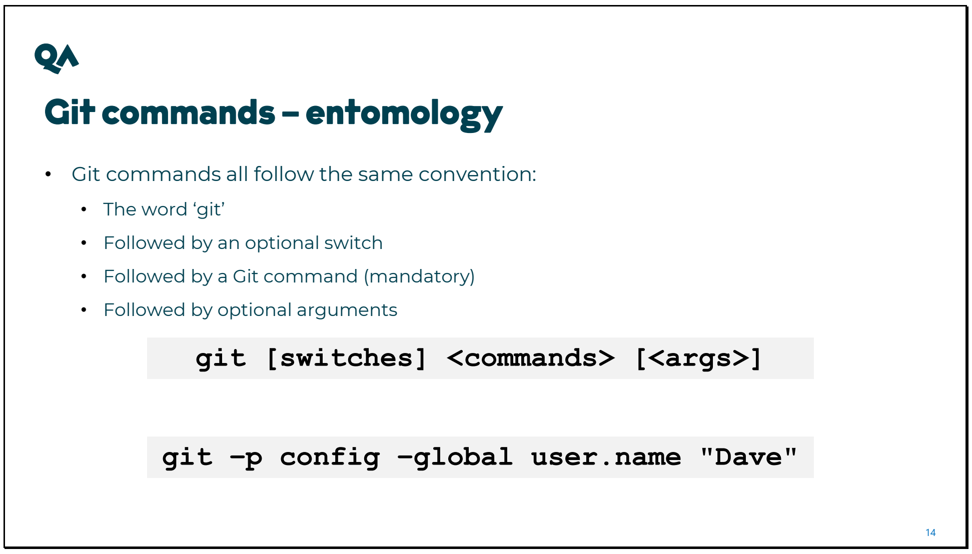


## Git user configuration – Ninja Lab

- Use the command line to configure Git with your user name and email
  - Check the settings to ensure these are set

13

Slide 14



## Git commands – entomology

- Git commands all follow the same convention:
  - The word ‘git’
  - Followed by an optional switch
  - Followed by a Git command (mandatory)
  - Followed by optional arguments

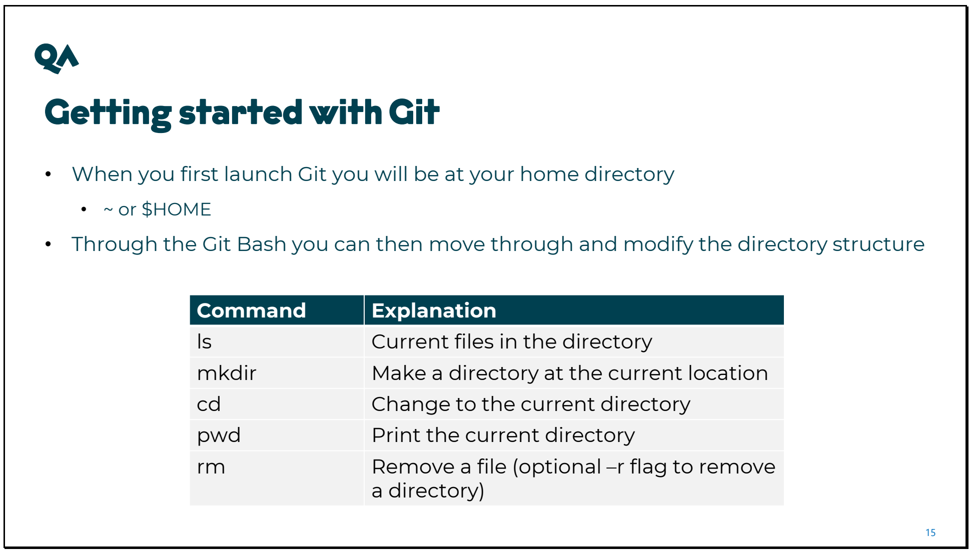
```
git [switches] <commands> [<args>]
```

```
git -p config -global user.name "Dave"
```

14

The -p switch paginates the output if needed.

Slide 15

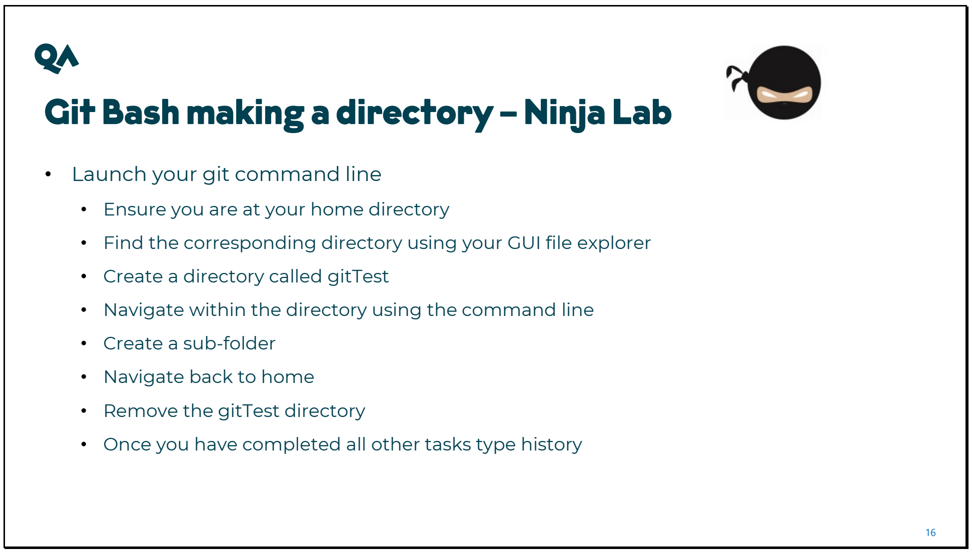


## Getting started with Git

- When you first launch Git you will be at your home directory
  - ~ or $HOME
- Through the Git Bash you can then move through and modify the directory structure

| Command | Explanation |
|---|---|
| ls | Current files in the directory |
| mkdir | Make a directory at the current location |
| cd | Change to the current directory |
| pwd | Print the current directory |
| rm | Remove a file (optional –r flag to remove a directory) |

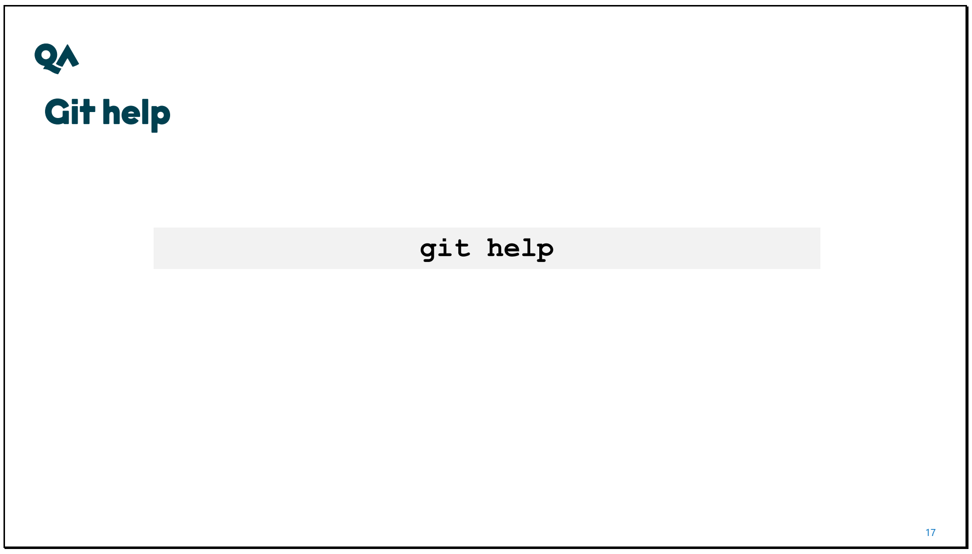
Slide 16



## Git Bash making a directory – Ninja Lab

- Launch your git command line
  - Ensure you are at your home directory
  - Find the corresponding directory using your GUI file explorer
  - Create a directory called gitTest
  - Navigate within the directory using the command line
  - Create a sub-folder
  - Navigate back to home
  - Remove the gitTest directory
  - Once you have completed all other tasks type history

Slide 17

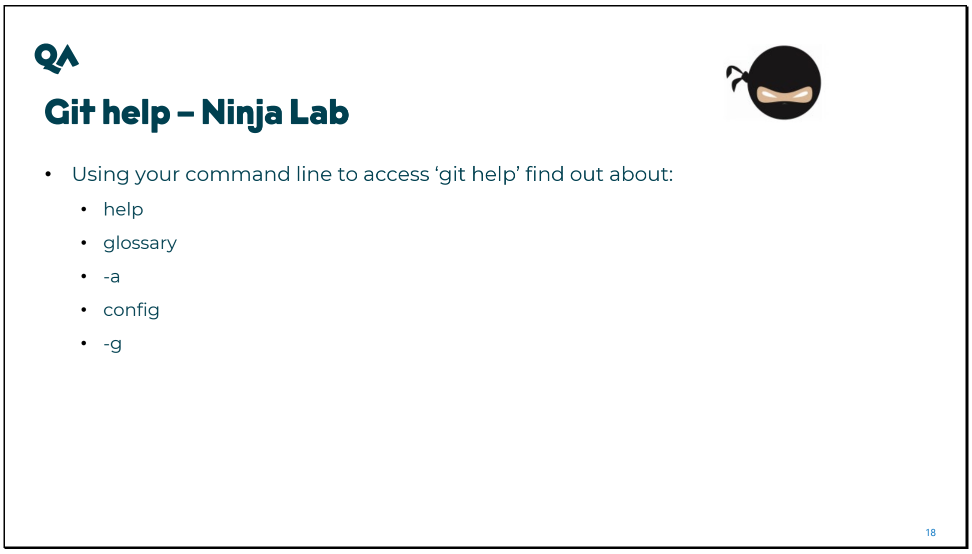


## Git help

```
git help
```

17

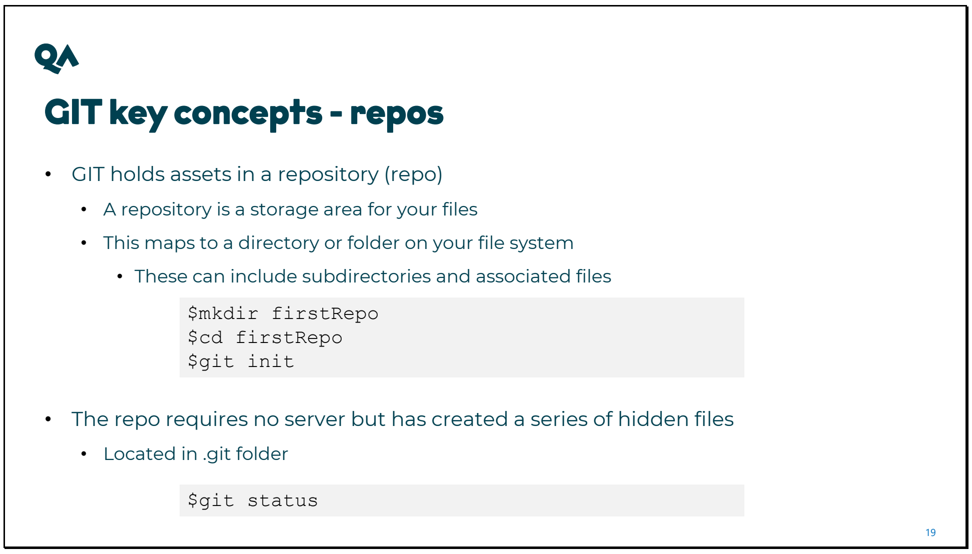
Slide 18



## Git help – Ninja Lab

- Using your command line to access ‘git help’ find out about:
  - help
  - glossary
  - -a
  - config
  - -g

Slide 19



## GIT key concepts - repos

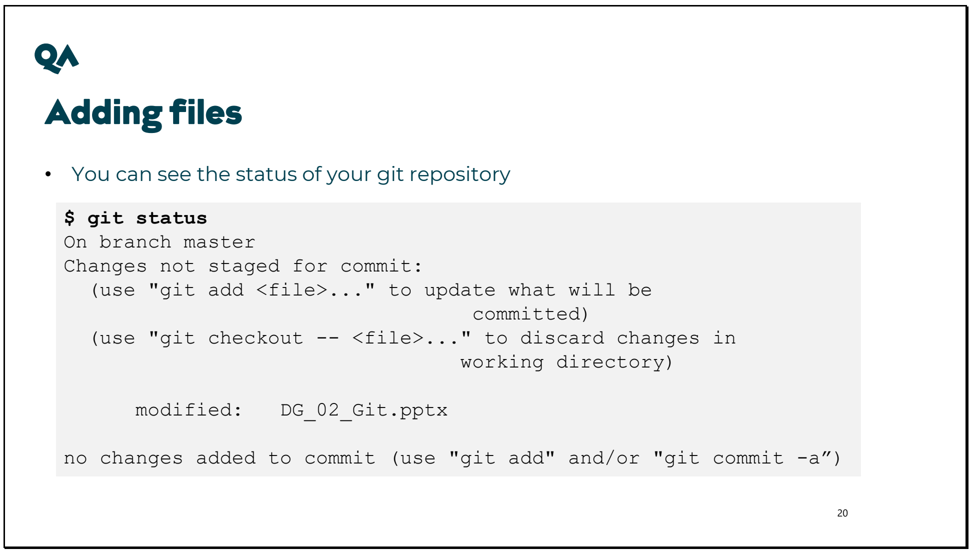
- GIT holds assets in a repository (repo)
  - A repository is a storage area for your files
  - This maps to a directory or folder on your file system
    - These can include subdirectories and associated files

```
$mkdir firstRepo
$cd firstRepo
$git init
```

- The repo requires no server but has created a series of hidden files
  - Located in .git folder

```
$git status
```

Slide 20

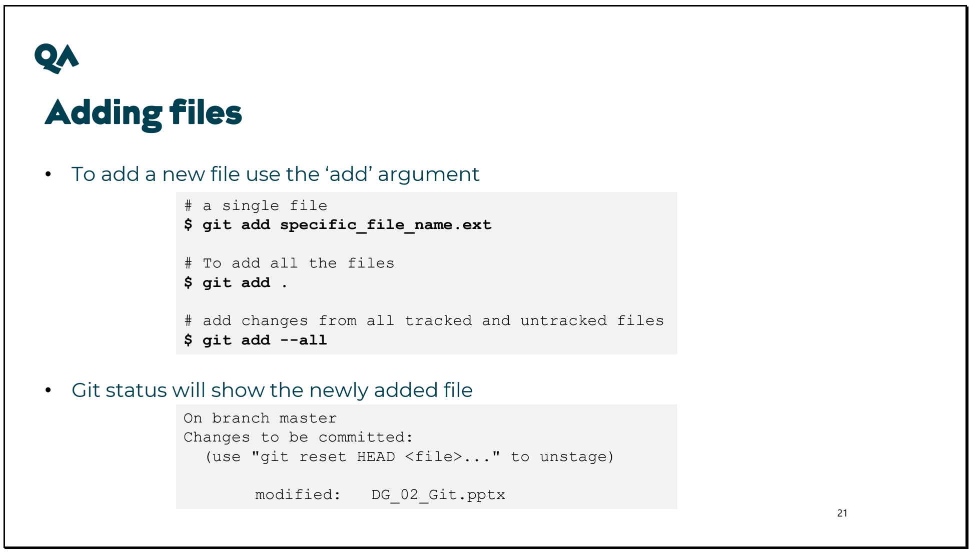


## Adding files

- You can see the status of your git repository

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be
                                  committed)
  (use "git checkout -- <file>..." to discard changes in
                                 working directory)

      modified:   DG_02_Git.pptx

no changes added to commit (use "git add" and/or "git commit -a”)
```

Slide 21



## Adding files

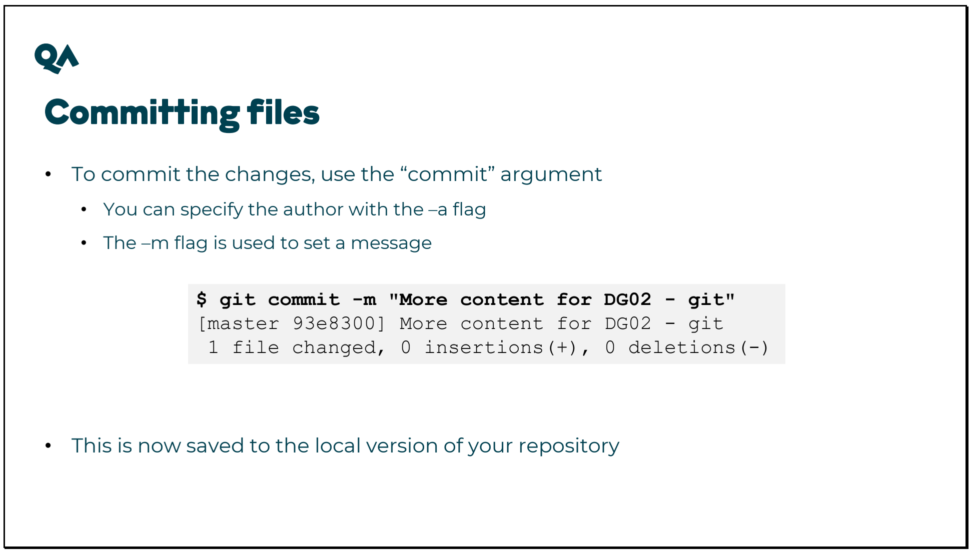
- To add a new file use the ‘add’ argument

```
# a single file
$ git add specific_file_name.ext

# To add all the files
$ git add .

# add changes from all tracked and untracked files
$ git add --all
```

- Git status will show the newly added file

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   DG_02_Git.pptx
```

Slide 22

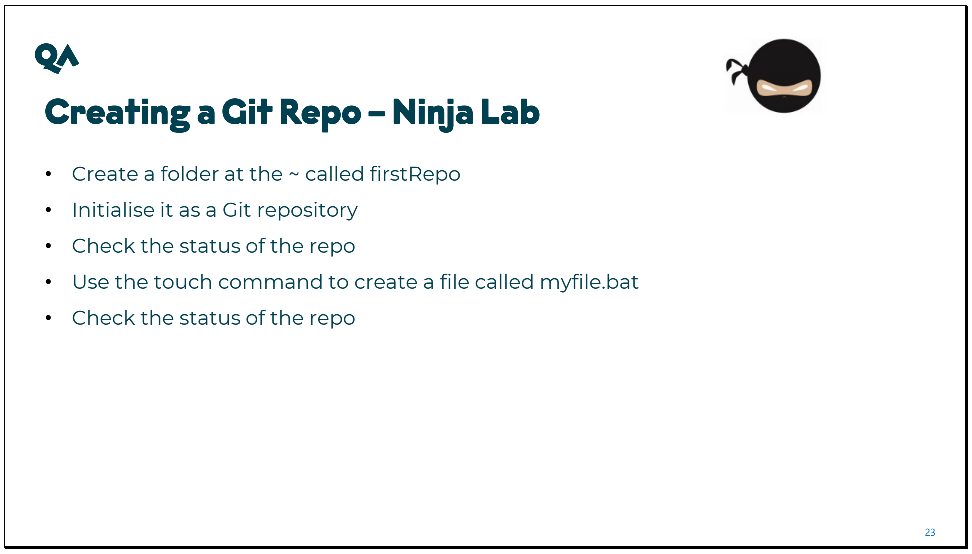


## Committing files

- To commit the changes, use the “commit” argument
  - You can specify the author with the –a flag
  - The –m flag is used to set a message

```
$ git commit -m "More content for DG02 - git"
[master 93e8300] More content for DG02 - git
 1 file changed, 0 insertions(+), 0 deletions(-)
```

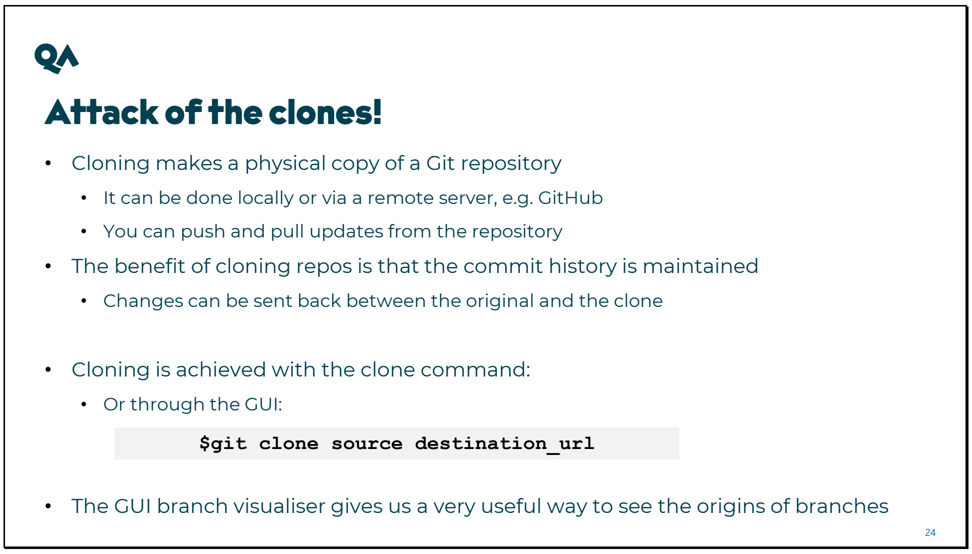
- This is now saved to the local version of your repository



## Creating a Git Repo – Ninja Lab

- Create a folder at the ~ called firstRepo
- Initialise it as a Git repository
- Check the status of the repo
- Use the touch command to create a file called myfile.bat
- Check the status of the repo

Slide 24



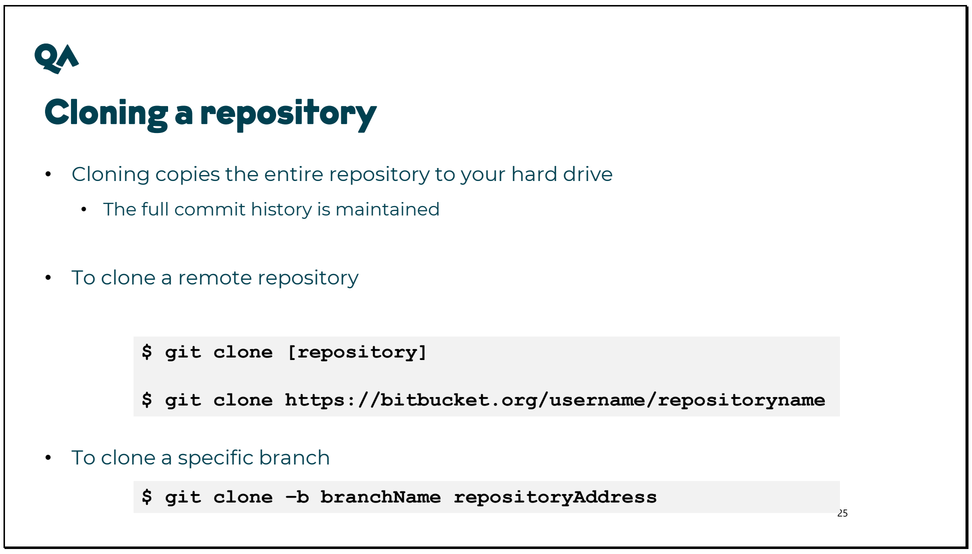
## Attack of the clones!

- Cloning makes a physical copy of a Git repository
  - It can be done locally or via a remote server, e.g. GitHub
  - You can push and pull updates from the repository
- The benefit of cloning repos is that the commit history is maintained
  - Changes can be sent back between the original and the clone
- Cloning is achieved with the clone command:
  - Or through the GUI:

```
$git clone source destination_url
```

- The GUI branch visualiser gives us a very useful way to see the origins of branches

Slide 25



## Cloning a repository

- Cloning copies the entire repository to your hard drive
  - The full commit history is maintained

- To clone a remote repository

```
$ git clone [repository]

$ git clone https://bitbucket.org/username/repositoryname
```

- To clone a specific branch

```
$ git clone –b branchName repositoryAddress
```