Git Quick Start

Hello and welcome to the Git Quick Start course!

This course is designed to provide those with basic Linux skills a general proficiency in the Git source control utility. This is achieved by reviewing short, practical use cases.

You are currently looking at the companion diagram for the course which notes key commands and concepts.

Please click **Main Menu** at the bottom-left of this page to get started!

Git Quick Start

Introduction and Architecture

Install and Configure

Working with Repositories

Ignoring Irrelevant Content

Cloning Repositories

Git Logging

Working with Branches

Pushing and Merging

Want More?

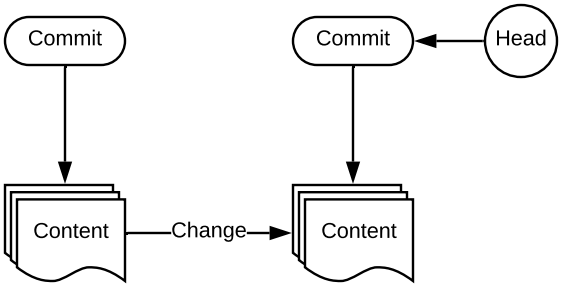
Git Quick Start

The `git` program is a source control tool created by Linus Torvalds.

Simply stated, `git` manages content snapshots, checksums, and metadata to keep track of changes in files.

Some basic terminology (we will go into more detail as we work through!):

- Each state is known as a commit.
- The current commit is called the head.
- Commits are affiliated with repositories and branches.
- The head may be moved between commits.



Git Quick Start

To install Git:

- `sudo yum install git`
  - or -
- `sudo apt-get install git`

Git configuration may be managed using `git config`:

- `git config --global user.name <username>`
- `git config --global user.email <email_addr>`
- `git config --system core.editor vim`

Alternatively, you may assign system-wide configurations in the config files:

- `/etc/gitconfig` which corresponds to --system
- `~/.gitconfig` or `~/.config/git/config` which corresponds to --global
- `.git/config` in a repository which corresponds to --local

Note: Files lower in the list override higher files.

File Format Example:

```
[user]
    name = john smith
    email = john@example.com
```

Git Quick Start

Use `git init <repo-directory>` to create or "initialize" a new repository in the specified directory.

The `init` subcommand will create a .git directory in the repository used for git metadata (see Git Deep Dive course for more information).

`git add <filename>` may be used to indicated relevant files for tracking in the repository.

Tracking information on files may be obtained using `git status`.

Once all relevant changes have been noted, run the following to commit the change to the repository:

- `git commit -m "text describing changes"`

If a file no longer requires tracking, you may stop tracking by using the following command:

- `git rm <file>`

Git Quick Start

Sometimes there are artifacts in your project which are transient or do not otherwise need to be tracked in source control.

There is a mechanism to have certain files disregarded for source control purposes.

The file `.gitignore` (local to a repository or system global) may contain a list of file names or patterns which git will ignore.

It is also possible to exclude certain paths using the following command:

- `git config --global core.excludesfile <path>`

Git Quick Start

Typically, the repository of record is not used for "working" purposes.
Changes to source are managed either in a separate branch or a remote repository.

Cloning Examples:

- `git clone <path_of_original> <clone_repo_location>.`
- `git clone \`
  `user@server:<orig_repo_path_relative_to_user_home> \`
  `<local_repo_path>`
- `git clone https://github.com/username/reponame \`
  `<local_repo_path>`

The cloned repository keeps repository logs separately, making the origin repository log less cluttered.

After local work is complete, the changes may be pushed back to the origin repository using:

- `git push origin master`

Branching achieves a similar effect within a single repository (more information in branching section).

Git Quick Start

Branching allows you to create an effective copy of the master branch within the repository that can be worked on without interfering with the master. This declutters the master branch.

Branching tends to be an expensive operation in other source control tools but it is incredibly efficient in Git.

Branches can be merged back into the master branch when work is complete.

Create a branch:

- `git branch <new branch>`

Begin working in the new branch:

- `git checkout <branch>`

Do both at once:

- `git checkout -b <new branch>`

It is also possible to branch from an existing branch.

Git Quick Start

`git log` will print information regarding commits and changes within the repository.

An abbreviated change log can be listed using `git log --oneline`

More detailed log information may be viewed with `git log -p`

It is also possible to look at information on a particular file:

- `git log -- <filename>`
- `git log --oneline <filename>`

To have a graph printed you can use the `--graph` and `--decorate` options:

- `git log --graph --decorate`
- **Note**: This is particularity useful when you have a multiple branches.

Branches may be pushed to remote sources, just like master.

How to push branches to origin:

- `git push origin --all`

Merging brings branches together (must be in the branch you are merging into):

- `git merge <target branch>`

Merging branches where files may have changed in diverging ways is called a merge conflict (see the Git Deep Dive for more on this).

Git Quick Start

Source Control with Git - More detail on Git!

DevOps Essentials - Learn more about DevOps!

AWS Essentails - Learn more about AWS!