02. Git and Version Control

What is version control?

Source code is stored in directories (and sub-directories). The source code changes over time and this change can be quite frequent and extensive.

Often, things happen that require people to access code as it looked in the past. Thus, it is useful to store backups of the state of the directory frequently, so you can go back to it later.

What git does

Git stores the history as a database called a repository. The database consists of commits, which are a snapshot of the code at a particular point in time. We can switch between commits, and git will change the contents of the directory in-place for us.

Commits are named using an ID, which is a SHA-1 hash of the source tree stored in the database.

Commits

A commit is a single snapshot of the state of the source tree.

```
git log # displays history of commits
git log --reverse # displays the logs backwards
git log --all # displays logs from all branches

git checkout commitID # jump to previous commit

git init # intialize an empty repository
git status # shows current status

git add filename # add a file in the directory to the staging area

git commit # commits the contents of the staging area to the repository dat
abase
git commit -m "commit message"
```

```
# git commit opens a text editor to type the commit message (first line for s
ummary, then more details).
```

Git database is stored kind of like a linked list, if we go back to a previous state, there is no reference for newer commits.

Branches

The history of real software projects will often be complex and nonlinear. Real software doesn’t just follow a linear history of changes, people will often create diverging histories from the main codebase.

Git supports this: you can create a new branch from an existing commit, and allow it to develop independently.

```
git branch # show the current branches in the repo

git checkout [id | tag] # switch the repo to the specified commit

git checkout branch_name # switch the repo to the specified branch
git switch branch_name

git branch branch_name # create a new branch with the specified name

git diff [id | tag] # show a list of the changes applied to the repository betw
een the current and specified commit

git switch -c branch_name # create and switch to a new brach with the spe
cified name
```

HEAD always points to current state

main always points to latest commit on the main branch

Hosting and remotes

It’s useful to keep a copy of your repository on a remote server.

Hosting services

Git is a distributed version control system (everyone has their own copy of the complete repo history). Having a copy of the repository on a remote server is a good idea.

Github is the primary example of a hosting service for git, but many others exist as well.

Remotes

A copy of a repository on a server is called a remote repository. Git can store the URL of a remote in the database, so your local copy can be kept in sync with it.

Hosting commands

```
git clone URL # download a local copy of the repository at the given URL

git remote [-v] # show a list of remote copies of the current repository

git remote add name URL # add the given URL as a remote to the current re
pository, giving it a name which can be used as a shorthand.

git push name_branch # push the given branch to the named repository
git push --all name # push all branches to the named repository
git fetch name_branch # fetch the most up to date version of the branch fro
m the named remote

git pull # runs a git fetch and a git merge
```

Merging

Combining existing branches together. Sometimes we want to combine branches together, this process is called merging. There are two types: a fast forward and a three-way.

Fast-forward merge

Suppose you have two branches, A and B.

Suppose A is an ancestor of B (i.e. if we trace back through the history from B, we will get to A).

This means the two histories don’t diverge, a series of changes to the code turned A into B. For this reason, merging the branches simply involves moving the human-readable name from A to B.

Three-way merge

If the branches diverge, a fast-forward merge won’t be possible. Instead, git will need to add a new commit to combine the two branches together, this is called a three-way merge.

Git will try to do a three-way merge automatically, but it won’t be able to do this if it comes across a merge conflict. You will need to resolve these conflict manually to complete the merge.

```
git merge branch # merge the currently active branch with the named branch

git pull # fetch and merge all branches from the remote repo

git merge --continue # continnue a merge that was paused due to a merge conflict
```

Configuration

Configuring git with the git config command.

```
git config --global user.name name # the author name that appears in commits

git config --global user.email email_address # the email address that appears in commits

git config --global core.editor editor_name # the editor used to write commit messages
git config --global init.defaultBranch name # change (or set) the default bra
nch name
```

.gitignore

Sometimes, you will have files in your git repo that you don’t want to commit to the remote (e.g. build artifacts, secret credentials, data, etc).

If git finds a file called .gitignore in the root of the directory, it will read it and ignore any files or folders listed in it. You can also list patters using wildcard operators.