Practical 1: Using Git

CDM Computing Subgroup Workshop

Albert Kong

February, 2024

Cloning a Repository

- Grab the web url for the repository (this should end in `.git`)
- Use `git clone <url>`

HTTPS: you will be prompted for your login credentials every time
SSH: need to setup a SSH keypair, but authentication is automatic afterwards

Example repository: https://github.com/AndreScaffidi/NatPy

Tangent: SSH keypairs

- Keys and configuration should be stored in `~/.ssh`
- To generate a new keypair, use: `ssh-keygen -t rsa -b 4096`
- Edit your `config` file and add a new entry:

```
Host github.com
     Hostname github.com
     User git
     IdentityFile ~/.ssh/<privatekey>
```

Making changes, tracking changes

- After making some changes within the cloned repository, use `git status` to see an overview of what has changed
- Use `git add <file>` and `git rm <file>` to stage changes
- Use `git checkout <file>` to revert changes and `git reset HEAD <file>` to unstage changes
- When all desired changes have been staged, use `git commit` to generate a new snapshot

Configuring Git

To make commits, you will need to configure a few options:

- `user.name` and `user.email` are used for attributing any commits you author
- `core.editor` is used whenever git prompts you for text input (typically commit messages)

To configure an option, use `git config --global <key> "<value>"`

History, Tags and Branches

- View every commit saved in Git with `git log`
- Revert the repository to a previous commit using `git checkout <commit hash>`
- Tag a commit for easy reference using `git tag <name> [<commit hash>]`
- Start a new line of development with `git branch <name>`
- The same `git checkout` command can be used to swap between branches and tags too

Starting a new Repository, working with Remotes

- Use `git init` in your top level directory, then `git add` and `git commit` as usual
- To link it with GitHub, first set up a new repository online and copy the clone url
- Use `git remote add <name> <url>` to tell git where to sync changes
- Upload your commit(s) with `git push`
- To check for upstream changes use `git fetch` and `git pull`
- Bonus: add a `.gitignore` file to keep things clean

When in doubt...

Use `git status`!