Intro to Git

Version Control and Collaboration

An Interactive Workshop

Instructor: Justin Hunt

Lecture Agenda

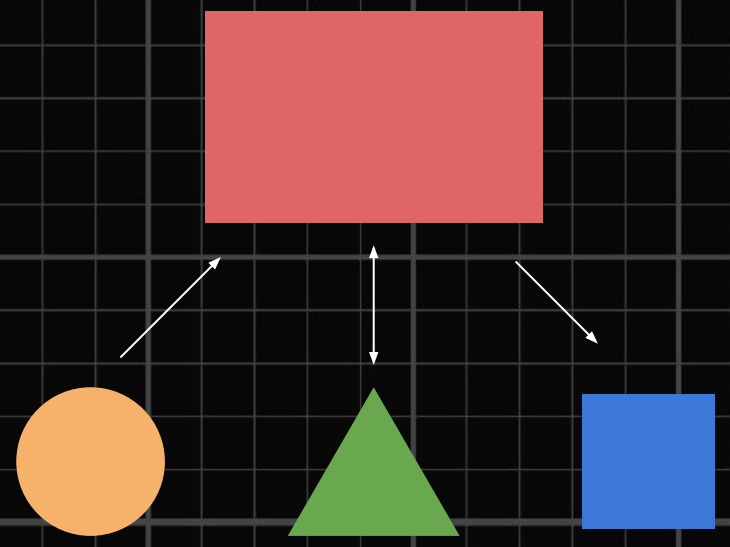
- What is git?
- Getting started w/ Git
- The Git workflow (Local)

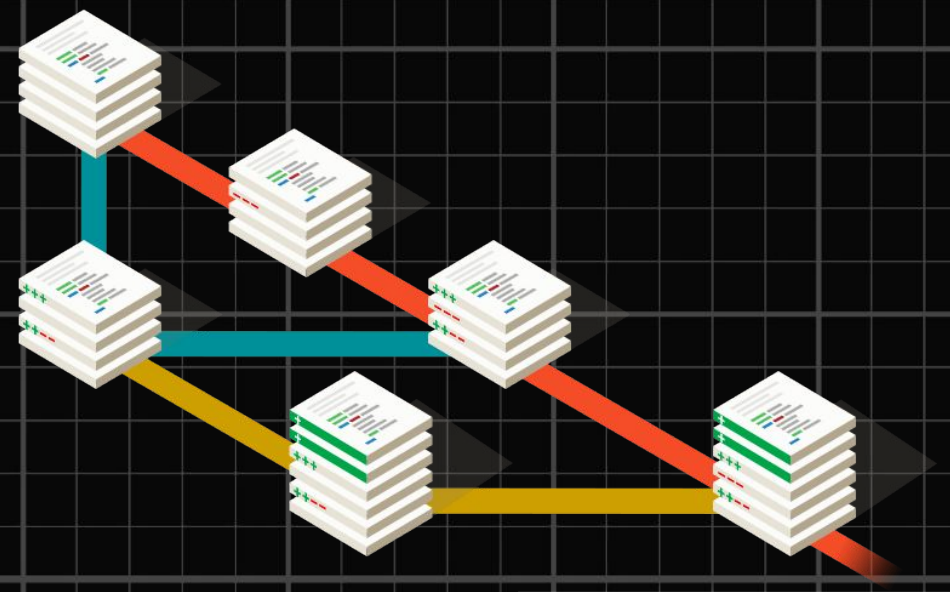
What is Git?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

For the following examples, we'll use source code, documentation, scripts, and config files as the files being version controlled, though in reality we can do this with nearly any type of file on a computer.





Source: git-scm.com

Starting with Git

Ensure `git` is installed.

```
sudo apt install git
```

(Debian)

```
sudo dnf install git
```

(Fedora)

```
sudo pacman -Syu install git
```

(Arch)

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

(Windows)

Verify git

Ensure `git` is installed.

```
git --version
```

(Debian)

(Fedora)

(Arch)

(Windows)

```
┌─(~/repos)──────────────────────────
└─(15:46:53)──> git --version
git version 2.51.0
```

Create authorship

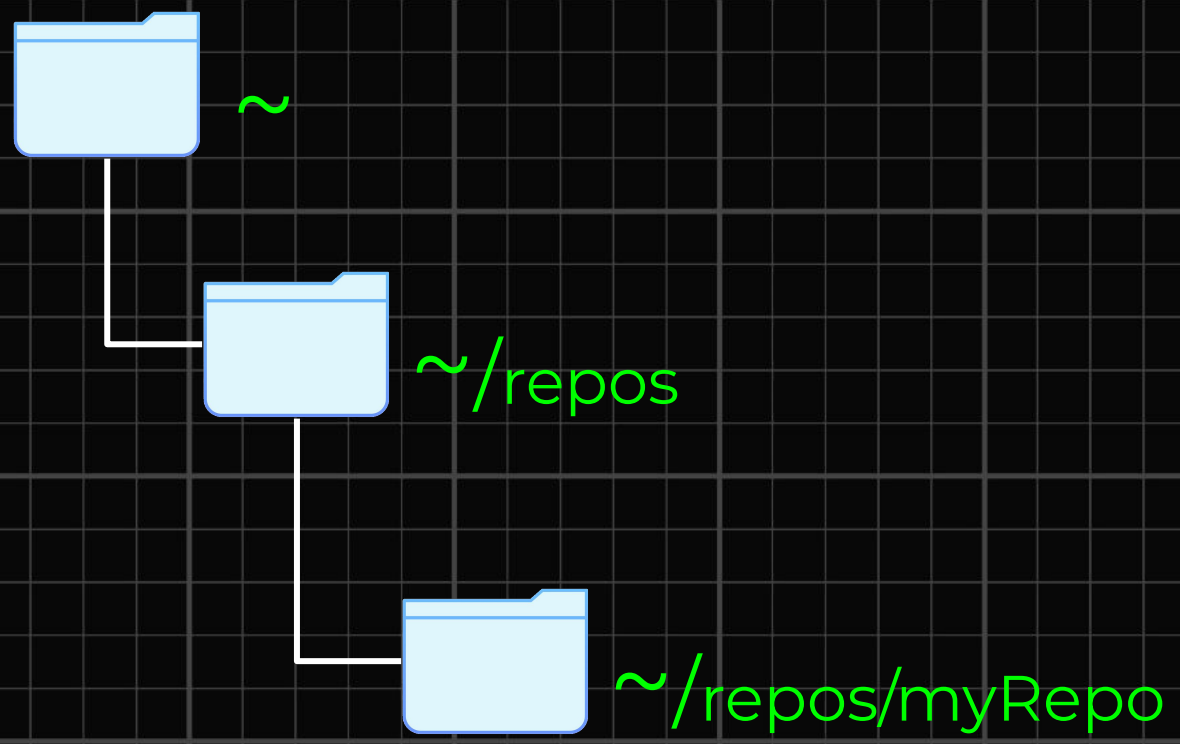
Now, let’s create authorship configurations.

```
git config –-global user.email “Email@Address.com”
git config --global user.name “FirstName LastName”
```

Starting with Git

Git'ing started...

Let's create the directory structure:




```
mkdir -p repos/myRepo
```

make directory

make directory 'repos' with subdirectory 'myRepo'

make parent directories as needed.

```
cd repos
```

```
pwd
```

```
>>> /home/<user>/repos
```

Cloning

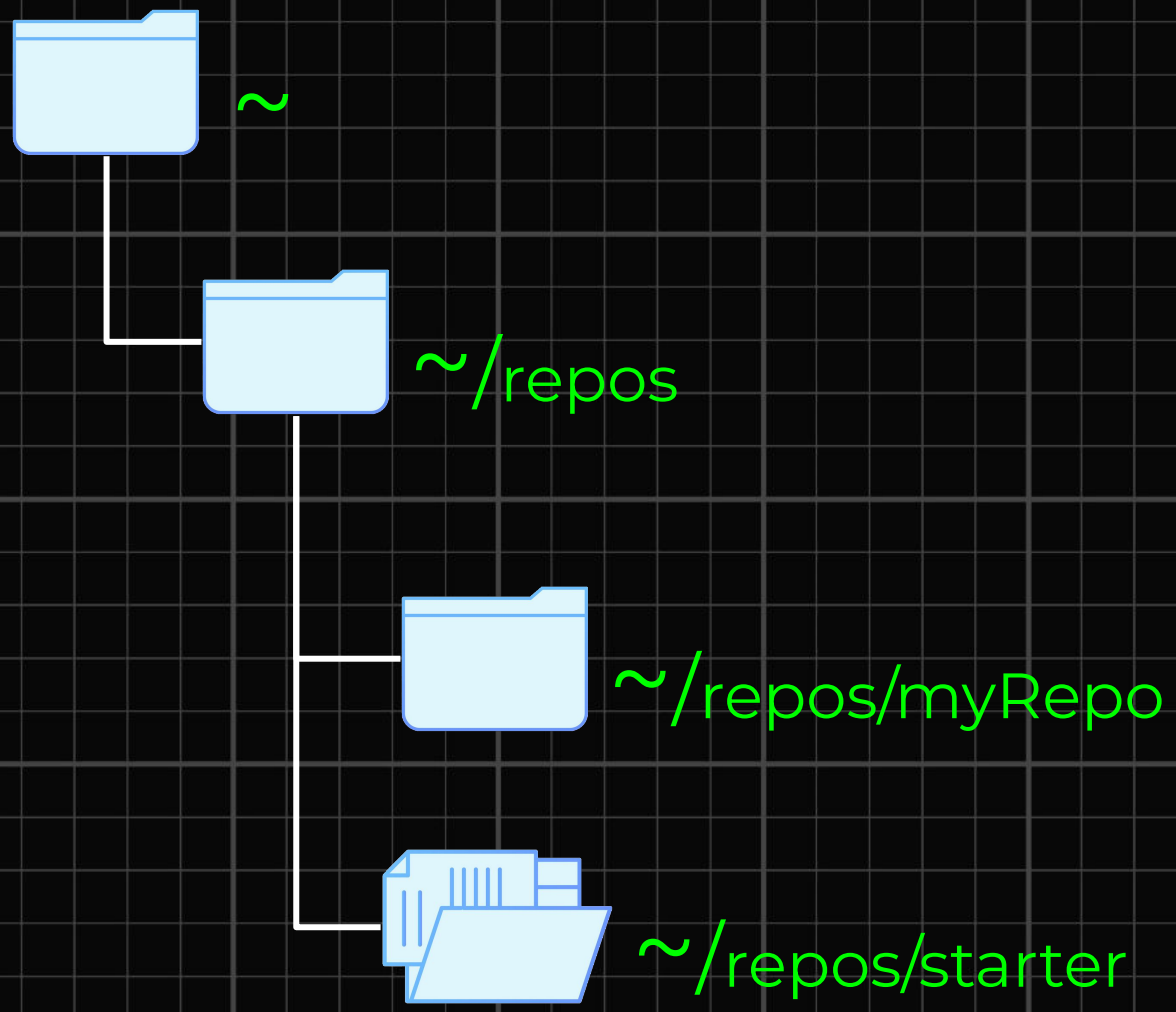
A common practice, even for individuals who don't have their own repos, is cloning. Cloning allows you to fetch the latest version of someone's project and host it locally.

We do this as a form of downloading open-source software, seeing other's code, or forking an existing project to work on, ourselves.

Whoa! I said a bunch of buzz-words. Let's clone an existing project that will help us with some definitions.

```
git clone https://github.com/jadamhunt/starter
```

```
[jhunt@fedora ~]$ git clone https://github.com/jadamhunt/starter
Cloning into 'starter'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 29 (delta 2), reused 28 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (29/29), 6.58 KiB | 6.58 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```




```
cd starter
```

```
ls
```

Now, you have the most updated version of a project I've posted for you.

You're free to explore, add to, remove from or otherwise change this starter directory.

The Git workflow (Local)

- Initialize a new repo
- Create content
- Index (add) changes
  - Branch
  - Merge
- Commit changes
- Remote VC

The Git workflow (Local)

- Initialize a new repo
- Create content
- Index (add) changes
  - Branch
  - Merge
- Commit changes
- Remote VC

Start by navigating to the directory that will house your repo, in our case *myRepo*.

We begin with the initialization command:

```
git init
```

This kicks of multiple processes and actions, chief among them:
Creates a new **branch**, called *master*. (eww, we'll fix this).
It also creates a hidden skeleton directory .git.

Let's renamer our branch to the common naming for new branches: main.

```
git branch -m main
```

Next, confirm with:

```
git status
```

```
git status
```

Current **Branch**

```
[jhunt@fedora myRepo]$ git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Content in Staging
(awaiting commit)

```
.git/
├── config
├── description
├── HEAD
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── fsmonitor-watchman.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── pre-merge-commit.sample
│   ├── prepare-commit-msg.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   ├── pre-receive.sample
│   ├── push-to-checkout.sample
│   ├── sendemail-validate.sample
│   └── update.sample
├── info
│   └── exclude
├── objects
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags
```

A repo HEAD is a representative current status marker on give base.
The head is presented in hexadecimal and indicates where the work currently being done.

```
(~/repos/starter/.git/refs)
(22:03:08 on main)──> tree

├── heads
│   └── main
├── remotes
│   └── origin
│       └── main
└── tags
```

```
├── logs
│   ├── HEAD
│   └── refs
│       ├── heads
│       │   └── main
│       └── remotes
│           └── origin
│               └── main
```

the .git/logs directory, tracks every commit from init to present in the form of the commit message and the previous heads.

.git/logs/HEAD

Initial head value | Revised head value | Author | Timestamp | Commit Message

```
0000000000000000000000000000000000000000 91dd2f16e28a7c556a238dcab00bc70f710ccaaa Justin Hunt <justin.adam.hunt@gmail.com> 1761146312 -0700    commit (initial): First Commit
91dd2f16e28a7c556a238dcab00bc70f710ccaaa c5c9f9c97930485827f7097f841ecc7118789f92 Justin Hunt <justin.adam.hunt@gmail.com> 1761146502 -0700    commit: created script folder
c5c9f9c97930485827f7097f841ecc7118789f92 868d83eb2c02fd229b44f6a34d83a6b2cbcc2922 Justin Hunt <justin.adam.hunt@gmail.com> 1761146596 -0700    commit: new script
868d83eb2c02fd229b44f6a34d83a6b2cbcc2922 868d83eb2c02fd229b44f6a34d83a6b2cbcc2922 Justin Hunt <justin.adam.hunt@gmail.com> 1761146657 -0700    checkout: moving from main to features
868d83eb2c02fd229b44f6a34d83a6b2cbcc2922 dfa382297fbcdff14857c41fea3d7e8b1da1a6be Justin Hunt <justin.adam.hunt@gmail.com> 1761146786 -0700    commit: added features
dfa382297fbcdff14857c41fea3d7e8b1da1a6be 868d83eb2c02fd229b44f6a34d83a6b2cbcc2922 Justin Hunt <justin.adam.hunt@gmail.com> 1761146885 -0700    checkout: moving from features to main
868d83eb2c02fd229b44f6a34d83a6b2cbcc2922 dfa382297fbcdff14857c41fea3d7e8b1da1a6be Justin Hunt <justin.adam.hunt@gmail.com> 1761146936 -0700    merge features: Fast-forward
dfa382297fbcdff14857c41fea3d7e8b1da1a6be cbf805cf7e306dd883b1f956f35d7de241413d71 Justin Hunt <justin.adam.hunt@gmail.com> 1761171062 -0700    commit: Updated README
cbf805cf7e306dd883b1f956f35d7de241413d71 afbfa097da7c4d47ef9915703a7d99abf543d854 Justin Hunt <justin.adam.hunt@gmail.com> 1761171237 -0700    commit: Docs/cheatsheet
afbfa097da7c4d47ef9915703a7d99abf543d854 ebdf32a8b3514483d15461624677c5e1b052007a Justin Hunt <justin.adam.hunt@gmail.com> 1761171340 -0700    commit: upload
ebdf32a8b3514483d15461624677c5e1b052007a 1994215bd21a31ea82a4a9ed0f937781ac109743 Justin Hunt <justin.adam.hunt@gmail.com> 1761190490 -0700    commit: + glossary
```

Permission
Octal

The Git workflow (Local)

- Initialize a new repo
- Create content
- Index (add) changes
  - Branch
  - Merge
- Commit changes
- Remote VC

We begin our content with a the quintessential README.md file.

The README.md file is a common entrant in git projects, as a working documentation text file formatted in markdown.

Upon viewing a repo from the github web ui, the README.md (if present) is displayed by default as the repo's greeter content.

```
echo "# First Repo" > README.md
```

Confirm the file creation with: `ls`

and it's contents with: `cat README.md`

The Git workflow (Local)

- Initialize a new repo
- Create content
- Index (add) changes
  - Branch
  - Merge
- Commit changes
- Remote VC

We've now have a fundamental change to our repo that has created what's known as a diff.

A diff (difference) is a deviation from the previous version. git is instantly aware of this deviation and report such deviations if asked with the following command:

```
git status
```

```
[jhunt@fedora myRepo]$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
```

Current branch

```
[jhunt@fedora myRepo]$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
```

previous commits

unstaged deviations (diffs).

Working Tree status

Our next step is to **index** (or add) this file.

...

Indexing the file includes it in the next **commit**, and moves it to a status we call **staged**.

The Git workflow (Local)

- Initialize a new repo
- Create content
- Index (add) changes
  - Branch
  - Merge
- Commit changes
- Remote VC

Let's add (index) our changes so they're included on the next commit.

```
git add .
```

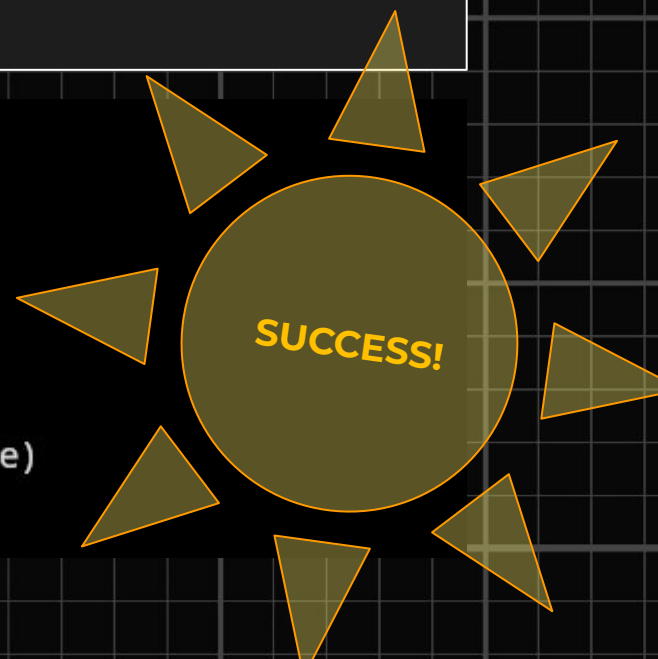
If there are no errors, this provides no feedback, so don't worry. Let's do some verification

```
git status
```

```
[jhunt@fedora myRepo]$ git add .
[jhunt@fedora myRepo]$ git status
On branch main

No commits yet

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



The Git workflow (Local)

- Initialize a new repo
- Create content
- Index (add) changes
  - Branch
  - Merge
- Commit changes
- Remote VC

Finally, it's time for our first **commit**!

This is a BIG deal, this means something is ready to be committed to our project! Let's do it!

```
git commit -m "Our first commit!"
```

msg flag

Commit message

This command instantiates the commit. Commits **require** a commit message! (as they should)

Commits and their messages track the life of the repo, and follow the project.

Repo

Ref

```
[jhunt@fedora myRepo]$ git commit -m "first commit"
[main (root-commit) 141a6b7] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

commit msg

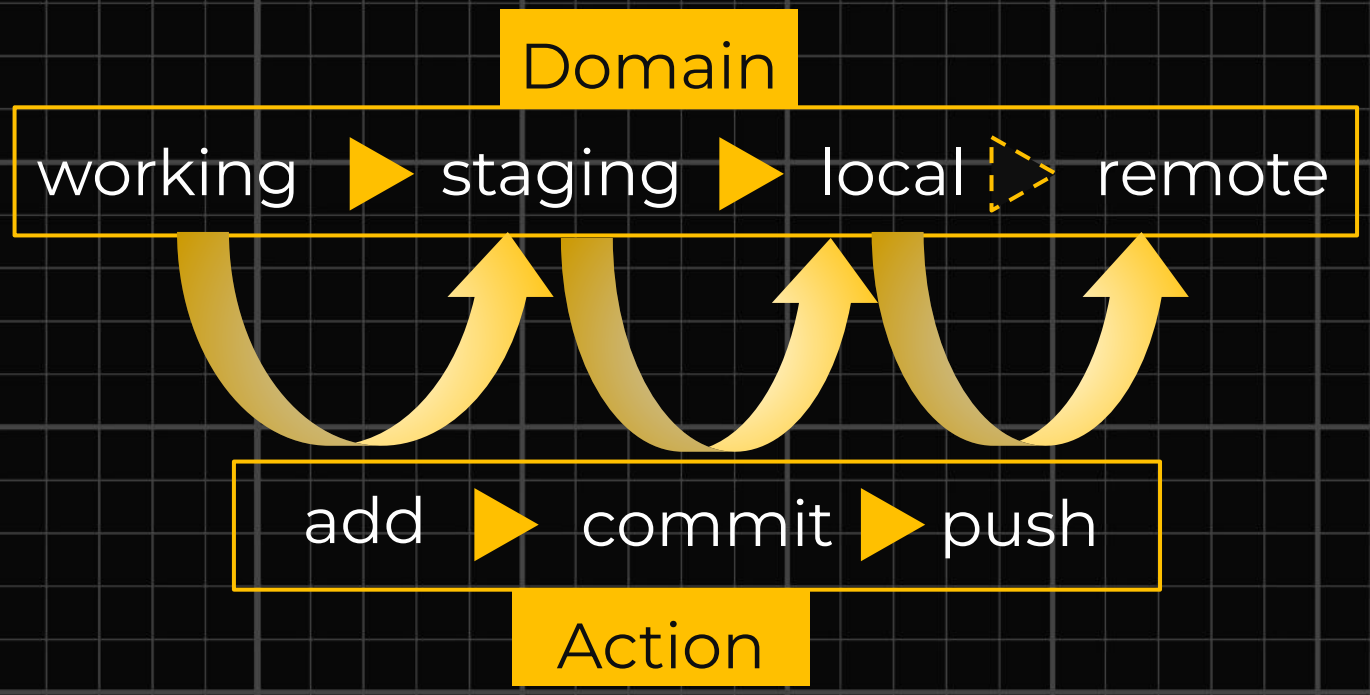
Change qty / type

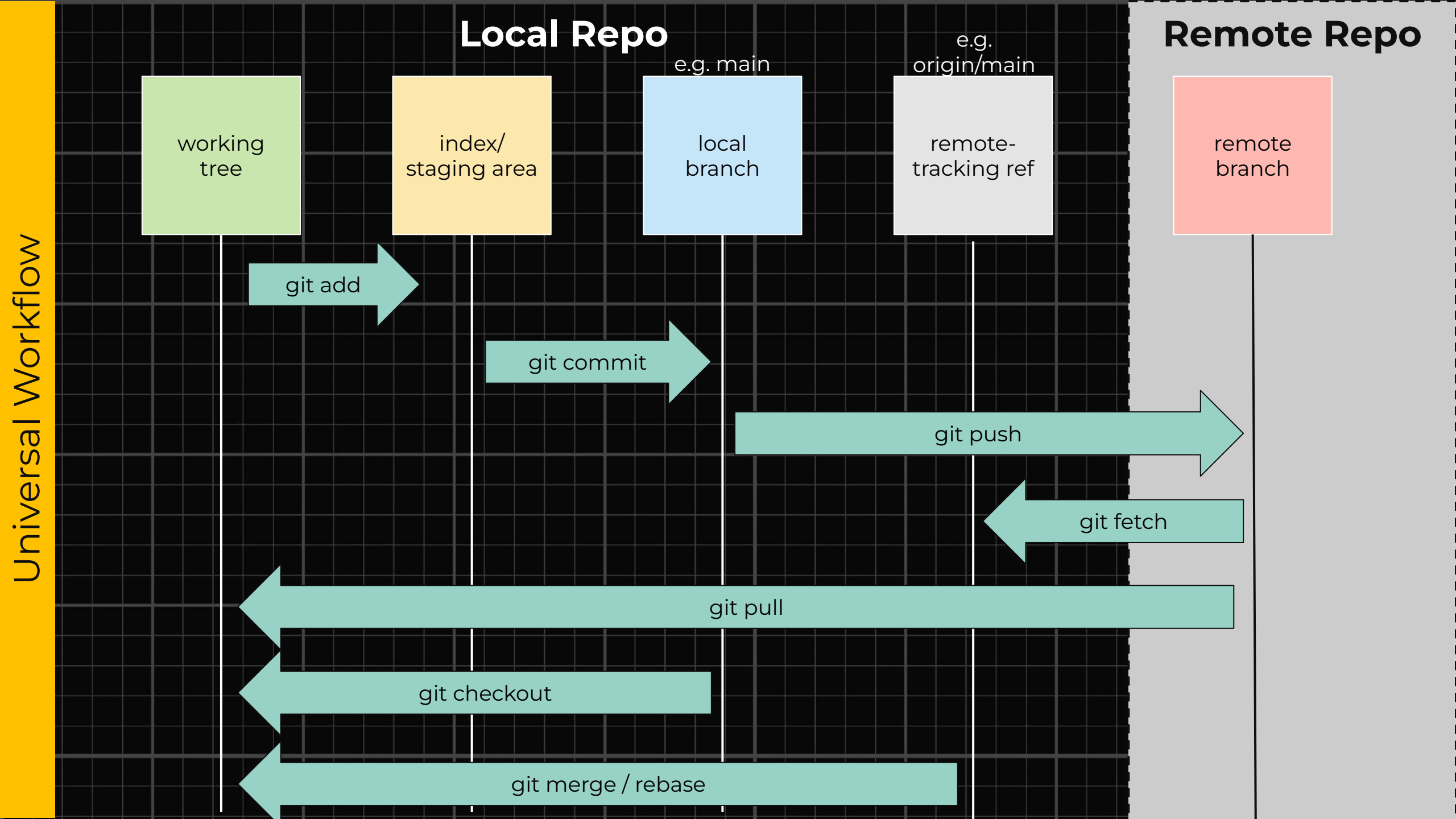
Operation Mode
Permissions:
    100 - Regular File
    644 - Permission Octal
Affected files

The Git workflow (Local)

- Initialize a new repo
- Create content
- Index (add) changes
  - Branch
  - Merge
- Commit changes
- Remote VC

When adding to our repo, anything we create will be in working. The work flow is as such:




Universal Workflow
Local Repo
Remote Repo
working tree
index/ staging area
e.g. main
local branch
e.g. origin/main
remote-tracking ref
remote branch
git add
git commit
git push
git fetch
git pull
git checkout
git merge / rebase

Connecting to remote

Prerequisites:

- `git` - git commands via cli
- `gh` - github cli interface
- Login to the GitHub web ui

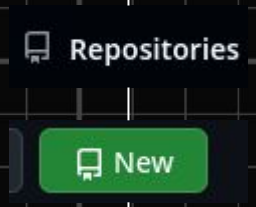
```
sudo apt install git gh
```

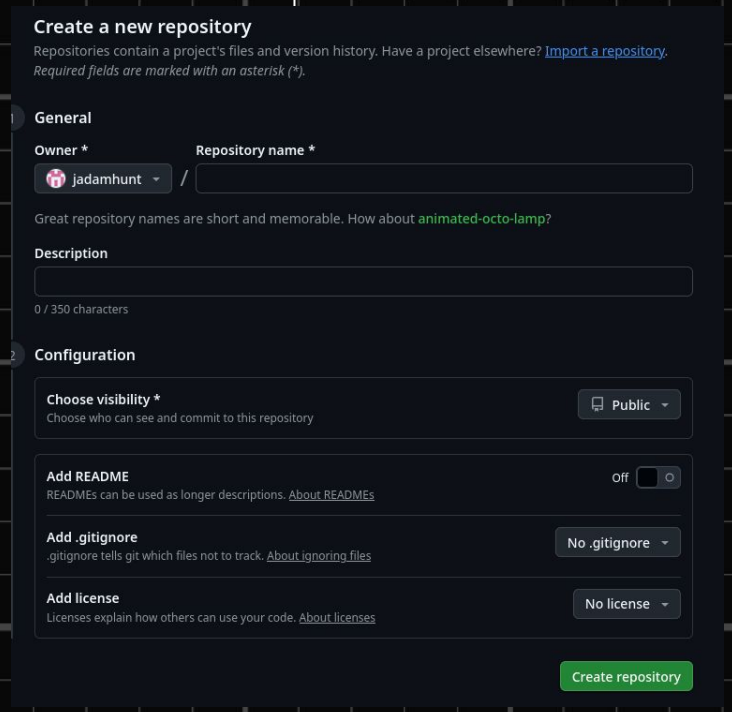
```
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser
```

Your first fresh Repo!

To create a repo from scratch we have two options:

Create from web UI





create from CLI

```
gh repo create
```

```
? What would you like to do? Push an existing local repository to github.com
? Path to local repository .
? Repository name myRepo
? Description myRepo
? Visibility Public
✓ Created repository jadamhunt/myRepo on github.com
  https://github.com/jadamhunt/myRepo
? Add a remote? Yes
? What should the new remote be called? origin
✓ Added remote https://github.com/jadamhunt/myRepo.git
? Would you like to push commits from the current branch to "origin"? Yes
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 229 bytes | 229.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/jadamhunt/myRepo.git
 * [new branch]      HEAD -> main
branch 'main' set up to track 'origin/main'.
✓ Pushed commits to https://github.com/jadamhunt/myRepo.git
```

```
gh repo create
```

Create a new repository on the command line

```
echo "# test01" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/{user}/{repo}.git
git push -u origin main
```

Push an existing repo from the command line.

```
git remote add origin https://github.com/{user}/{repo}.git
git branch -M main
git push -u origin main
```

git add README.md

```
┌(~/repos/test01)──────────────────────(jhunt@Alpine:pts/0)┐
└(07:42:38 on main ★)──> git add README.md
┌(~/repos/test01)──────────────────────(jhunt@Alpine:pts/0)┐
└(07:42:47 on main +)──> git commit -m "New commit"
[main (root-commit) 0ec3069] New commit
 1 file changed, 7 insertions(+)
 create mode 100644 README.md
```

Now, to push to a remote repo:
git push // This should provide an error the 1st time.

git remote add {branch_name} {remote repo} #github
git push # !! Another error we need to set an upstream branch.

```
git push --set-upstream {project} {branch}
```

mkdir scripts
cd scripts , create and edit hello.sh script
set permissions, run script
git add .
git status
git commit -m “Created scripts”
git status
git push, confirm on webui
git remote -v
create goodbye script, add, commit, push #!create flowchart!#
New Branch
git branch features
git checkout features
create dir / add feature content /
add, add commit, push
confirm features NOT in webui
git checkout main, ## NOTE! branch is ahead of main by 1 commit.
git merge features, git push
#confirm features present webuit

```
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```