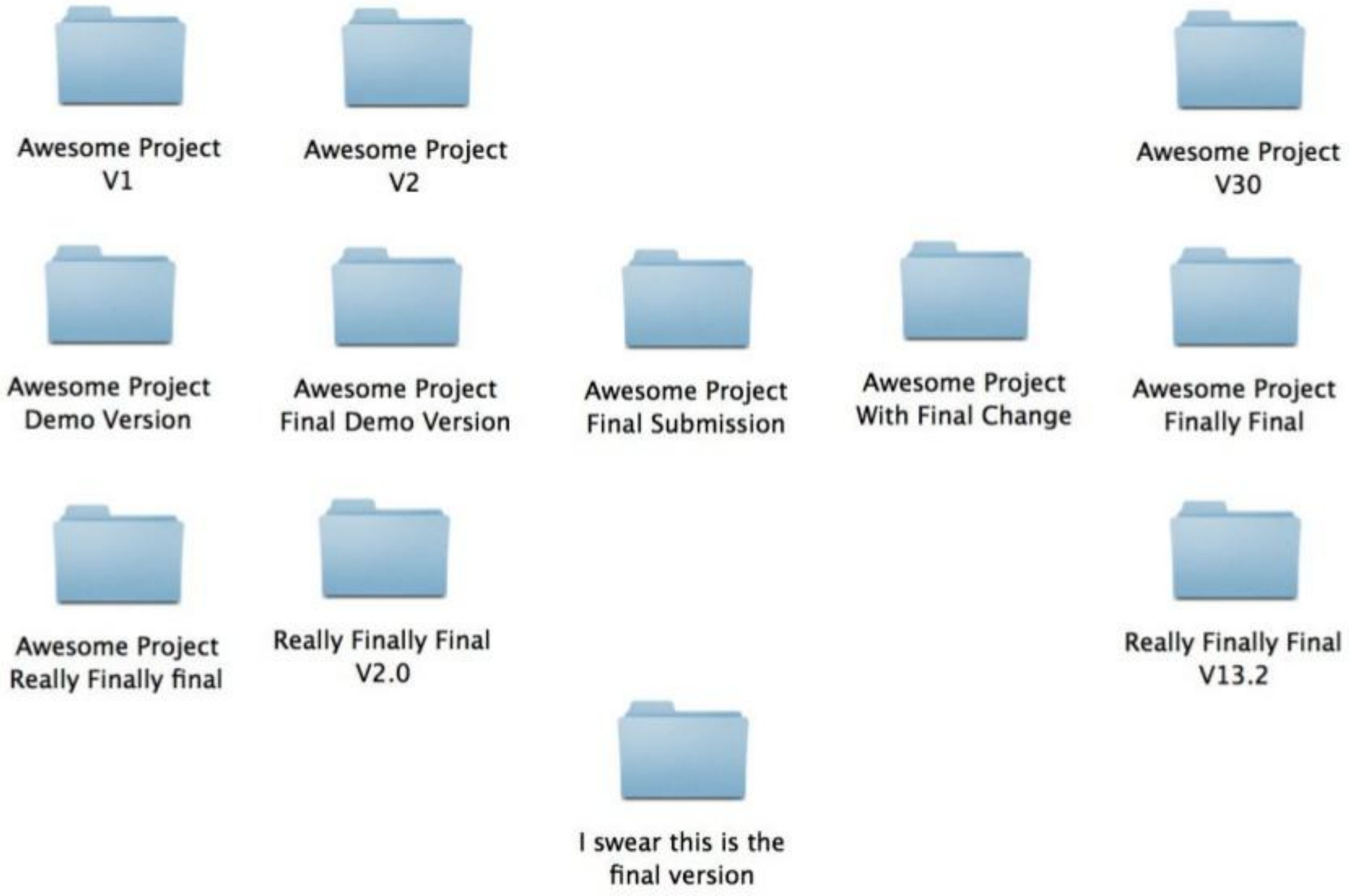
Lecture 18: Git!

CSE 374: Intermediate Programming Concepts and Tools

Administrivia

- HW 3 posted later today -> Extra credit due date Wednesday Nov 18th @ 9pm
- HW 4 out Monday Nov 16th -> Extra credit due date Wednesday Nov 25th @ 9pm
- HW 5 out Monday November 30th -> Extra credit due date Wednesday Dec 5th @ 9pm
- HW 6 out Monday Dec 7th, Due end of quarter
- Review Assignment #2 out Friday Nov 13th, Due Friday Nov 20th @ 9pm
- Review Assignment #3 out Wednesday Dec 9th , Due end of quarter
- **End of quarter due date Wednesday December 16th @ 9pm**

Does your file system look like this?


Awesome Project V1
Awesome Project V2
Awesome Project V30
Awesome Project Demo Version
Awesome Project Final Demo Version
Awesome Project Final Submission
Awesome Project With Final Change
Awesome Project Finally Final
Awesome Project Really Finally final
Really Finally Final V2.0
Really Finally Final V13.2
I swear this is the final version

Issues faced

- Multiple copies of the same files
- Changes leading to scripts being broken with no way to go back
- Accidental deletion of files
- Lost computer leads to loss of files
- Collaboration on a project can be difficult if each collaborator has their own copy

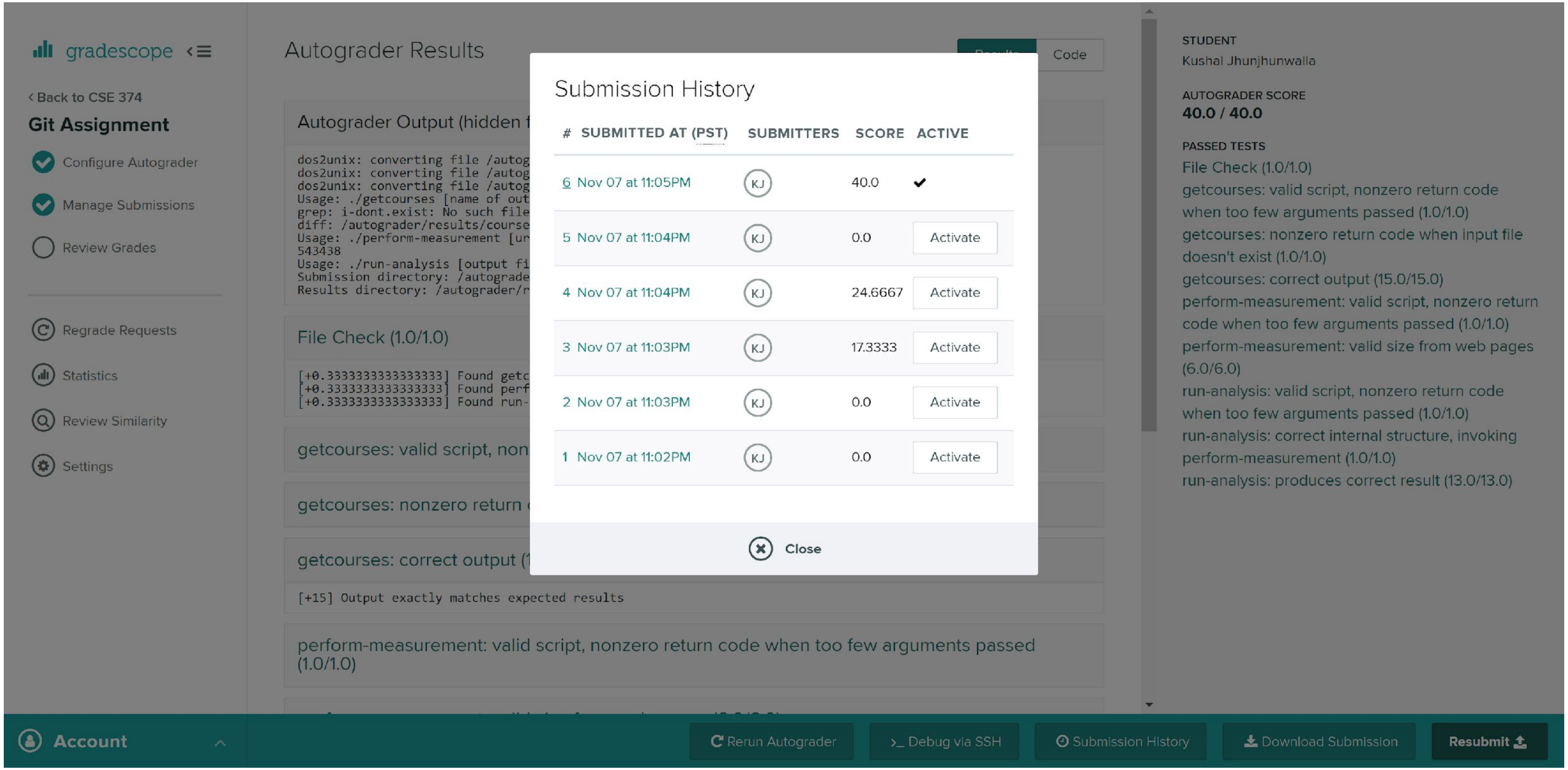
Version Control to the rescue!

- Version control is software that keeps track of changes to a set of files

- Version control enables multiple people to simultaneously work on a single project.

- Places we see version control:
  - Microsoft Word allows us to undo our most recent changes (go back to a previous version)
  - In Google Docs you can see the version history of files, see who made these changes and revert back
  - You (hopefully) back up your computer to a cloud storage like Google Drive or Dropbox
  - Gradescope also allows you to look at your previous submission

Version Control in Gradescope



Autograder Results

Autograder Output (hidden f

```
dos2unix: converting file /autog
dos2unix: converting file /autog
dos2unix: converting file /autog
Usage: ./getcourses [name of out
grep: i-dont.exist: No such file
diff: /autograder/results/course
Usage: ./perform-measurement [ur
543438
Usage: ./run-analysis [output fi
Submission directory: /autograde
Results directory: /autograder/r
```

File Check (1.0/1.0)

```
[+0.3333333333333333] Found getc
[+0.3333333333333333] Found perf
[+0.3333333333333333] Found run-
```

getcourses: valid script, non

getcourses: nonzero return

getcourses: correct output (1

```
[+15] Output exactly matches expected results
```

perform-measurement: valid script, nonzero return code when too few arguments passed (1.0/1.0)

Submission History

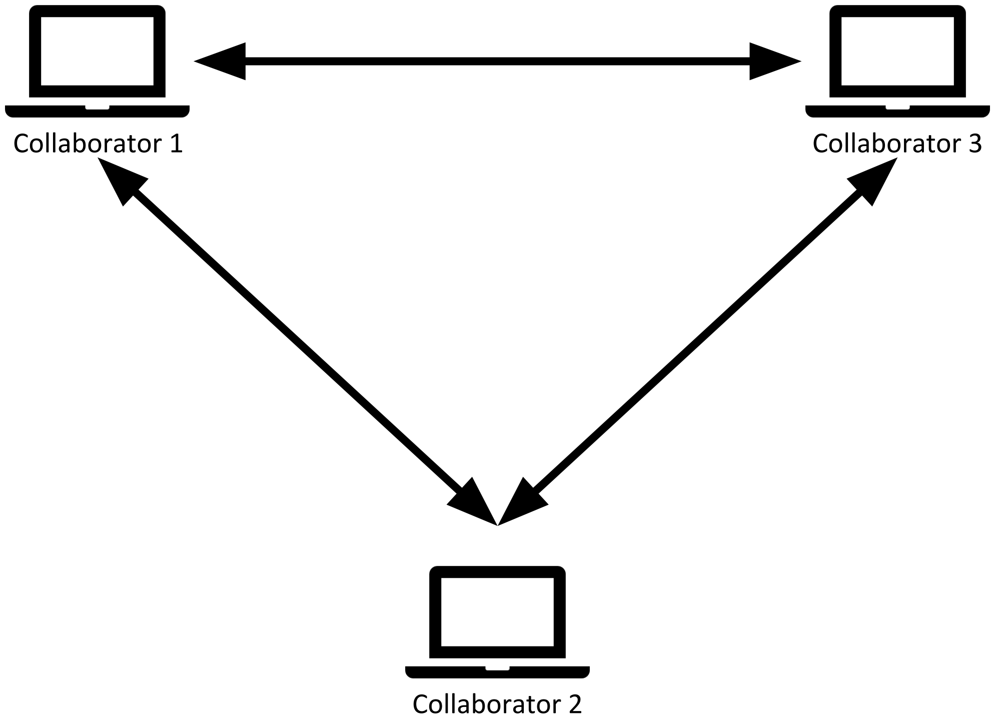
#	SUBMITTED AT (PST)	SUBMITTERS	SCORE	ACTIVE
6	Nov 07 at 11:05PM	KJ	40.0	✓
5	Nov 07 at 11:04PM	KJ	0.0	Activate
4	Nov 07 at 11:04PM	KJ	24.6667	Activate
3	Nov 07 at 11:03PM	KJ	17.3333	Activate
2	Nov 07 at 11:03PM	KJ	0.0	Activate
1	Nov 07 at 11:02PM	KJ	0.0	Activate

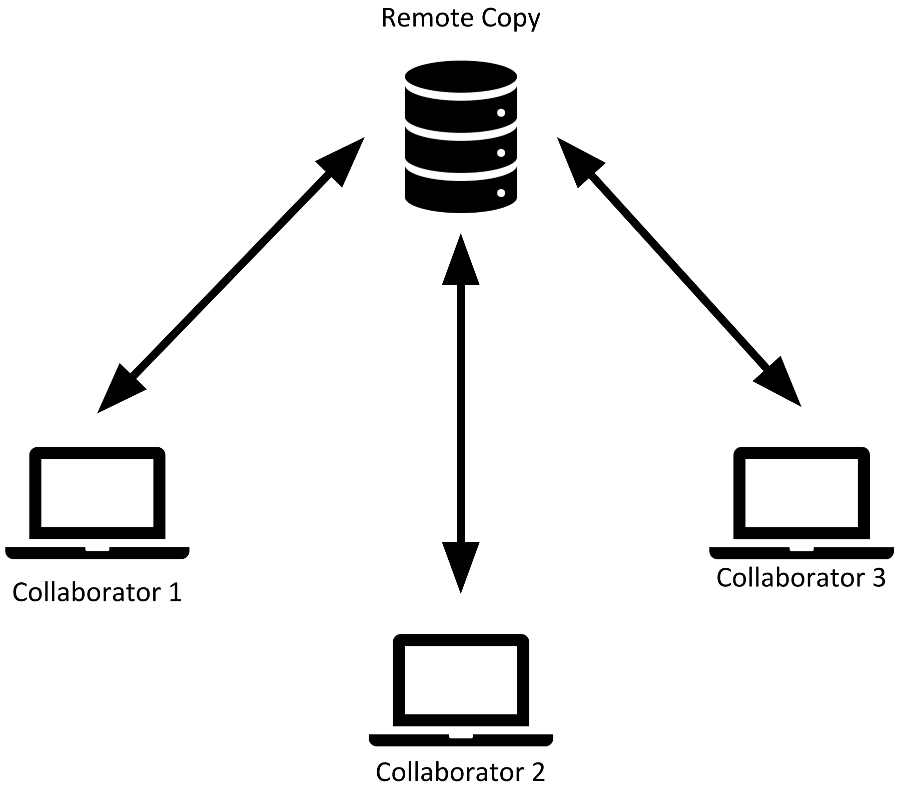
STUDENT
Kushal Jhunjhunwalla

AUTOGRADER SCORE
40.0 / 40.0

PASSED TESTS
File Check (1.0/1.0)
getcourses: valid script, nonzero return code when too few arguments passed (1.0/1.0)
getcourses: nonzero return code when input file doesn't exist (1.0/1.0)
getcourses: correct output (15.0/15.0)
perform-measurement: valid script, nonzero return code when too few arguments passed (1.0/1.0)
perform-measurement: valid size from web pages (6.0/6.0)
run-analysis: valid script, nonzero return code when too few arguments passed (1.0/1.0)
run-analysis: correct internal structure, invoking perform-measurement (1.0/1.0)
run-analysis: produces correct result (13.0/13.0)

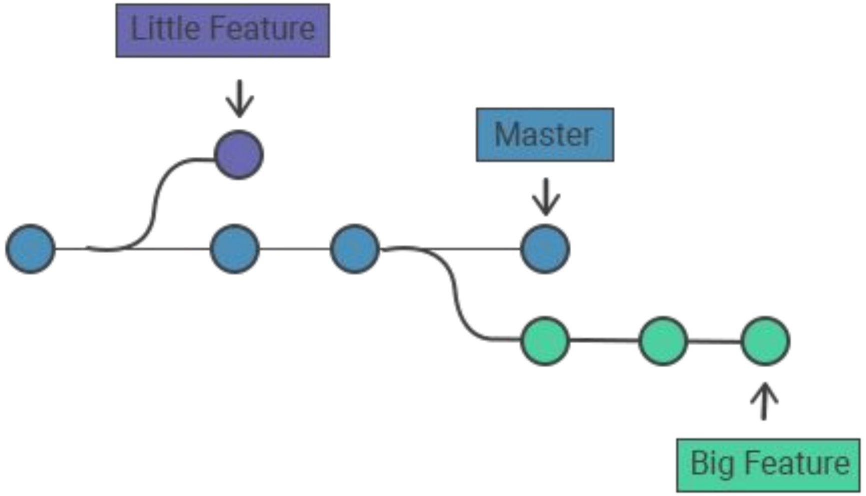
Decentralized vs centralized storage





Intro to Git

- Git is a version control system optimized for text-based files
- Git ≠ GitHub
- "origin" copy of the repo is stored on a Git server
  - The remote repository is the *defacto* central repository
  - Remote repositories are hosted on services like GitHub, Gitlab, or Bitbucket
- Everyone shares changes by pushing their changes and pulling others' changes



Repository

- A repository, commonly referred to as a **repo**, is a location where a project lives
- Each collaborator has a copy of a repository for the project which they sync with the central copy
- What should be inside of a repository?
  - Source code files (i.e. .c files, .java files, etc)
  - Build files (Makefiles, build.xml)
  - Images, general resources files
- What should not be inside of a repository (generally)
  - Object files (i.e. .class files, .o files)
  - Executables
  - Huge media files (e.g. videos)
  - Any application credentials
  - System files (e.g. .DS_Store in Mac)

A repository's history is a series of "commits"

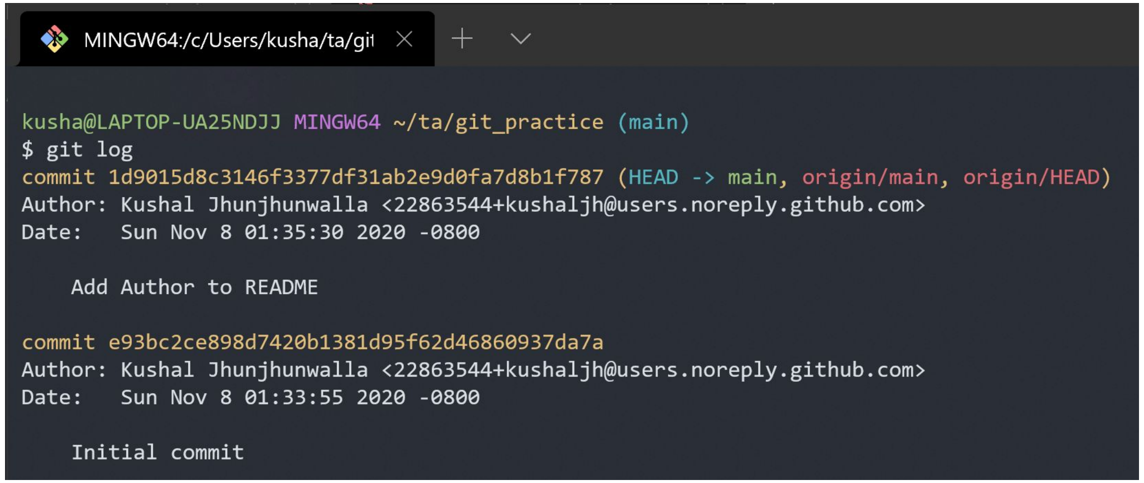
- Each commit makes changes to the files in the repo

- Commit history serves as a log of the changes everyone made

- Commits are easy, and free! Commit early and often.
  - Ever accidentally deleted something and forgot what it was?
  - If you make a mistake, you can look at the changes since the last commit

What is a “commit”?

- A "**commit**" is a single set of changes made to your repository
- Essentially a **version** of the project you want to save
- Also records:
  - The name of the author
  - The date and time
  - A "**commit message**": short sentence describing what that commit did
- Identified by an ID, or "**SHA**"

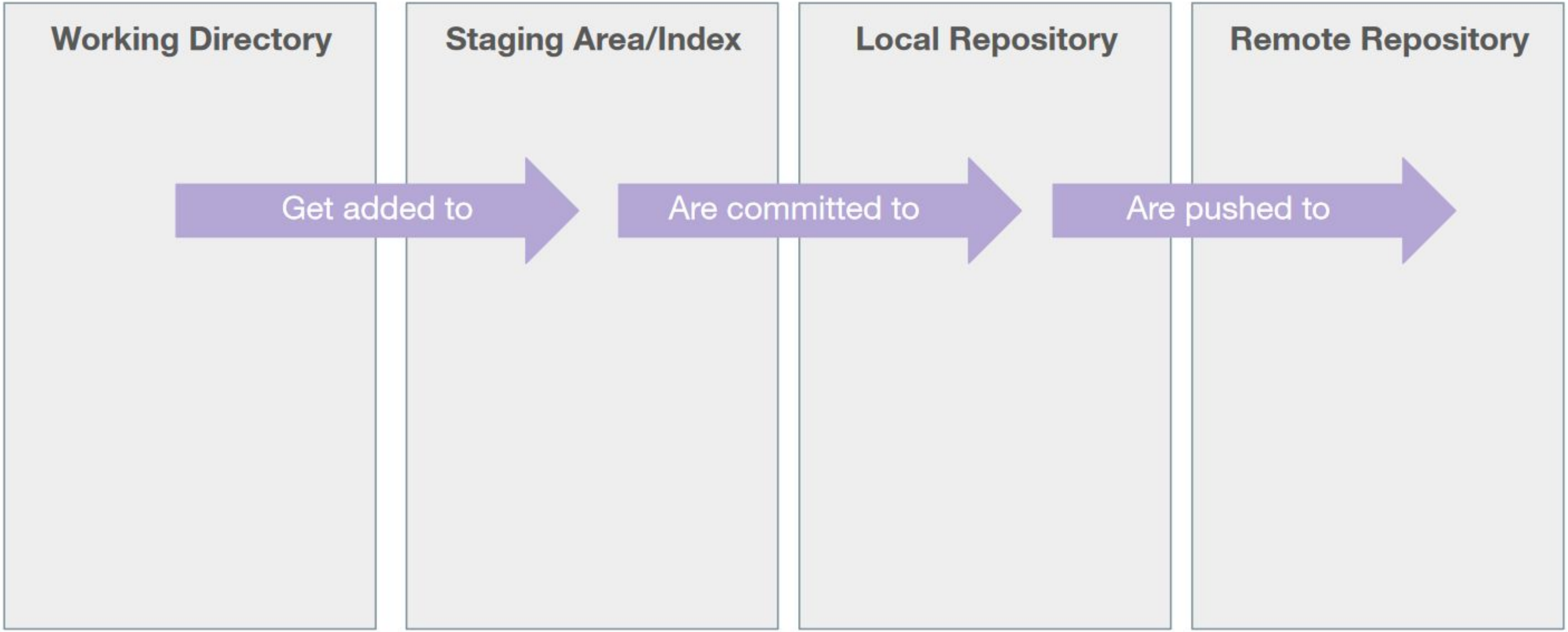


```
kusha@LAPTOP-UA25NDJJ MINGW64 ~/ta/git_practice (main)
$ git log
commit 1d9015d8c3146f3377df31ab2e9d0fa7d8b1f787 (HEAD -> main, origin/main, origin/HEAD)
Author: Kushal Jhunjhunwalla <22863544+kushaljh@users.noreply.github.com>
Date:   Sun Nov 8 01:35:30 2020 -0800

    Add Author to README

commit e93bc2ce898d7420b1381d95f62d46860937da7a
Author: Kushal Jhunjhunwalla <22863544+kushaljh@users.noreply.github.com>
Date:   Sun Nov 8 01:33:55 2020 -0800

    Initial commit
```

Phases of Git

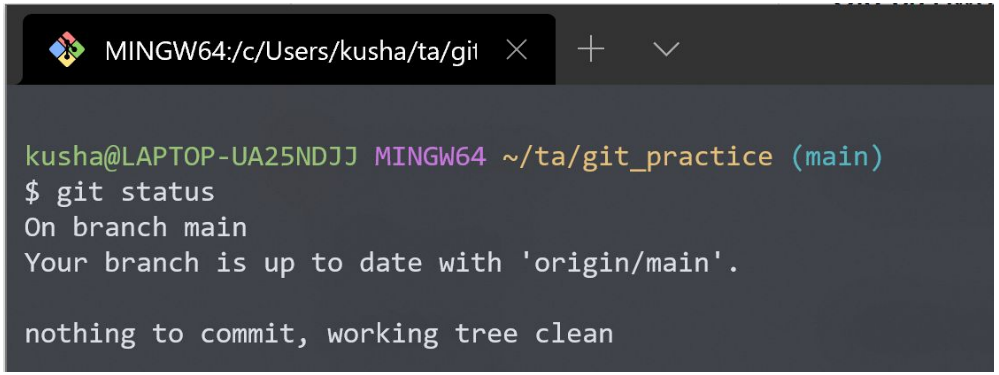
Working Directory	Staging Area/Index	Local Repository	Remote Repository
Working changes	Changes you’re preparing to commit	Local copy of the repository with your committed changes	Remote shared repository (Usually stored with a platform like GitHub/Gitlab)

Phases of Git

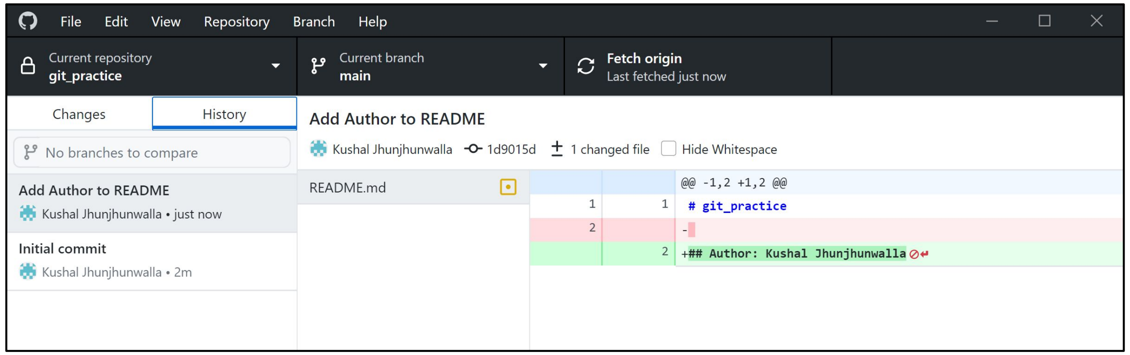


"git" command

- The "git" command is the primary way of interacting with git
- You must "cd" into the folder where your **repo** is stored, or any subfolder within it
- Used like any other shell command!
- There is a GUI available for most companies (e.g. GitHub Desktop)
- Must be installed on your computer (already installed on **klaatu**)



```
kusha@LAPTOP-UA25NDJJ MINGW64 ~/ta/git_practice (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```



Add Author to README
Kushal Jhunjhunwalla • just now

Initial commit
Kushal Jhunjhunwalla • 2m

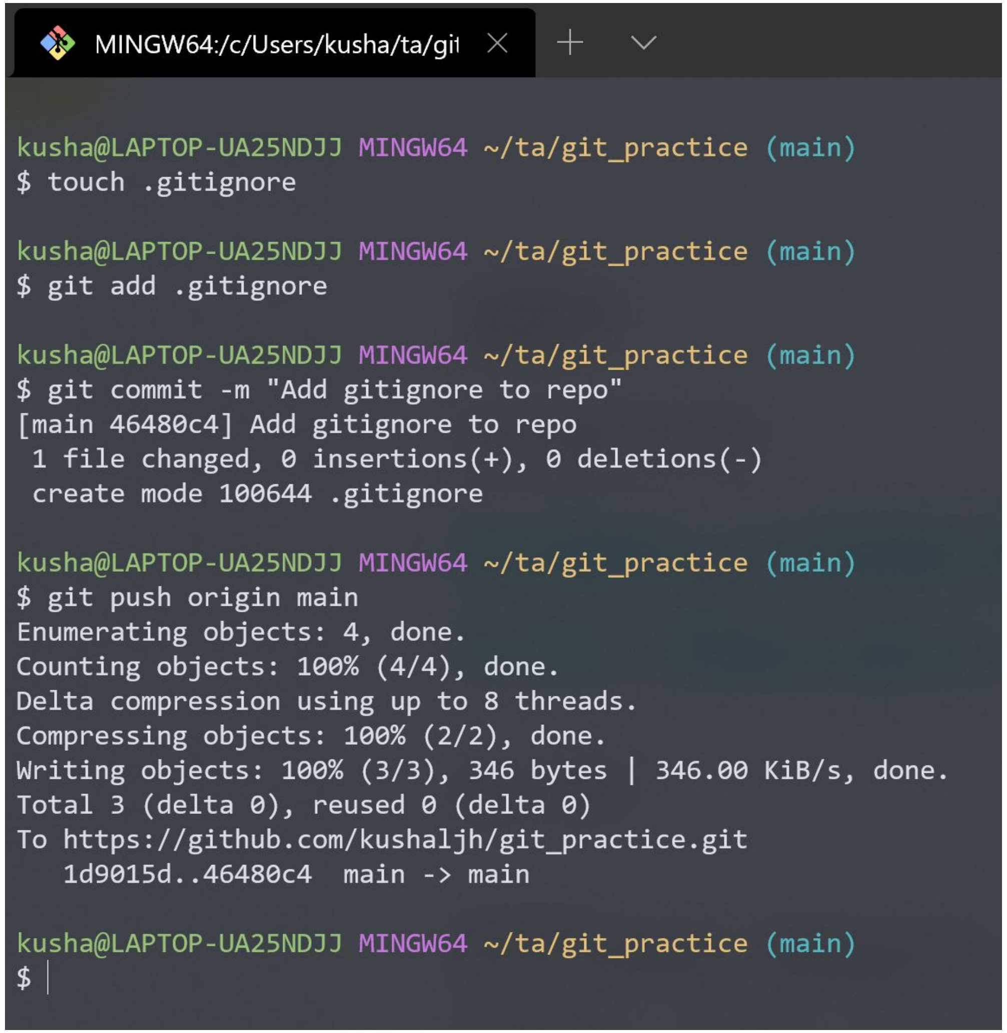
Add Author to README
Kushal Jhunjhunwalla 1d9015d 1 changed file □ Hide Whitespace

README.md

```
@@ -1,2 +1,2 @@
1 1  # git_practice
2    -
  2  +## Author: Kushal Jhunjhunwalla
```

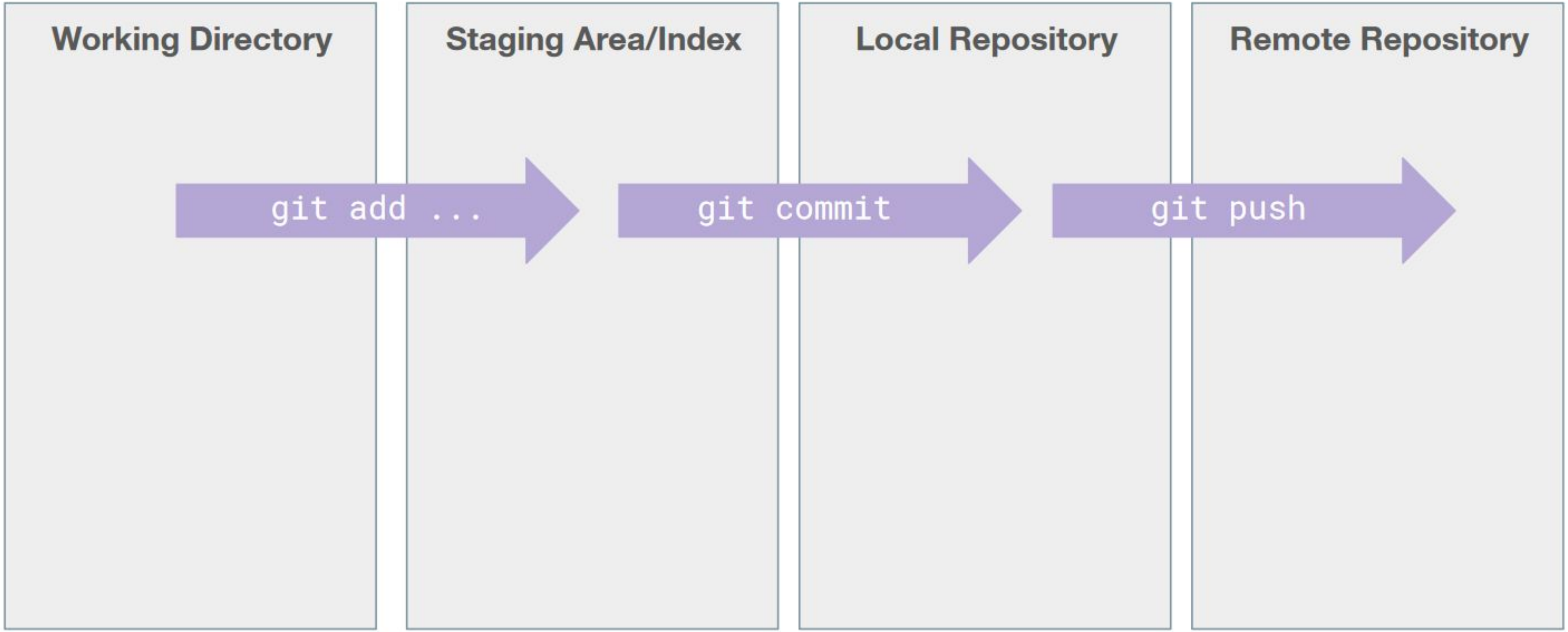
Some "git" commands

- `git init`
  - Create a new empty git repo or convert an existing folder to a git repo
- `git add`
  - Preparing edited files to be saved (committed) to a repo
- `git commit`
  - Records (saves) changes to a repo
  - Accompanied by a short descriptive message
- `git push`
  - Update the remote copy of the repo with the local changes and commits



```
kusha@LAPTOP-UA25NDJJ MINGW64 ~/ta/git_practice (main)
$ touch .gitignore

kusha@LAPTOP-UA25NDJJ MINGW64 ~/ta/git_practice (main)
$ git add .gitignore

kusha@LAPTOP-UA25NDJJ MINGW64 ~/ta/git_practice (main)
$ git commit -m "Add gitignore to repo"
[main 46480c4] Add gitignore to repo
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .gitignore

kusha@LAPTOP-UA25NDJJ MINGW64 ~/ta/git_practice (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 346 bytes | 346.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/kushaljh/git_practice.git
   1d9015d..46480c4  main -> main

kusha@LAPTOP-UA25NDJJ MINGW64 ~/ta/git_practice (main)
$ 
```

Phases of Git



NOTE: There are way more git commands than what is listed here - this is a simplified model to get us started.

Staging and committing overview

Staging and committing overview

```
git init
```

Staging and committing overview

```
git init
```

git repo

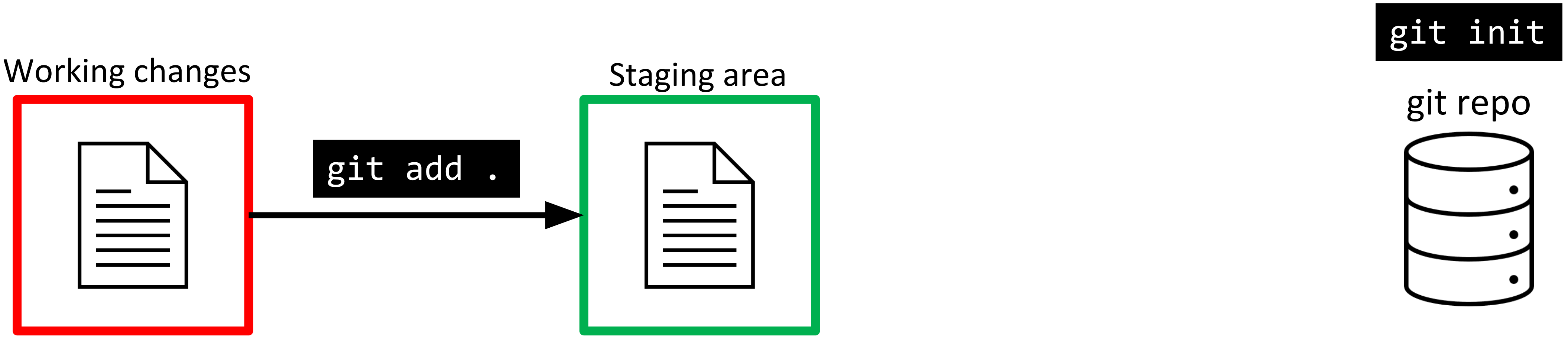
Staging and committing overview

Working changes

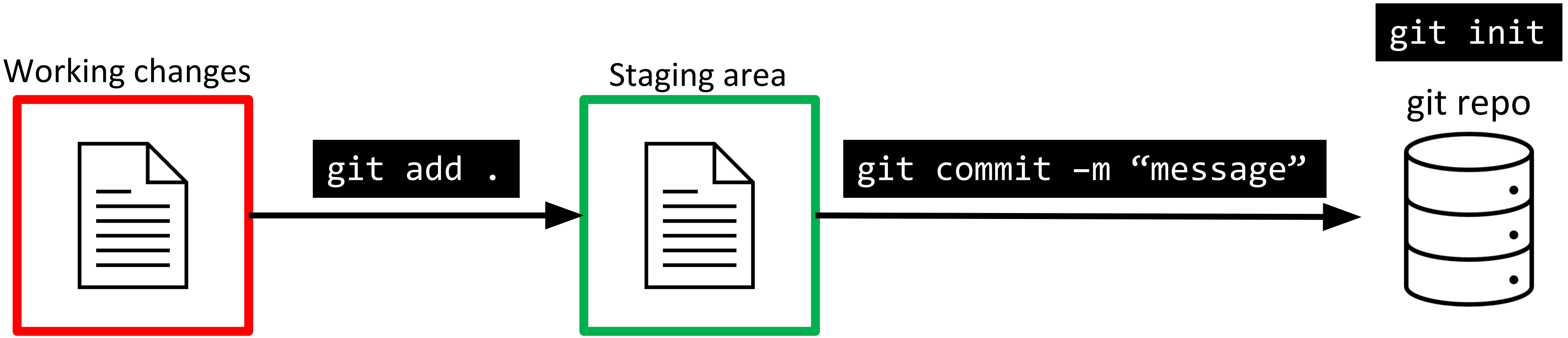
```
git init
```

git repo

Staging and committing overview



Staging and committing overview



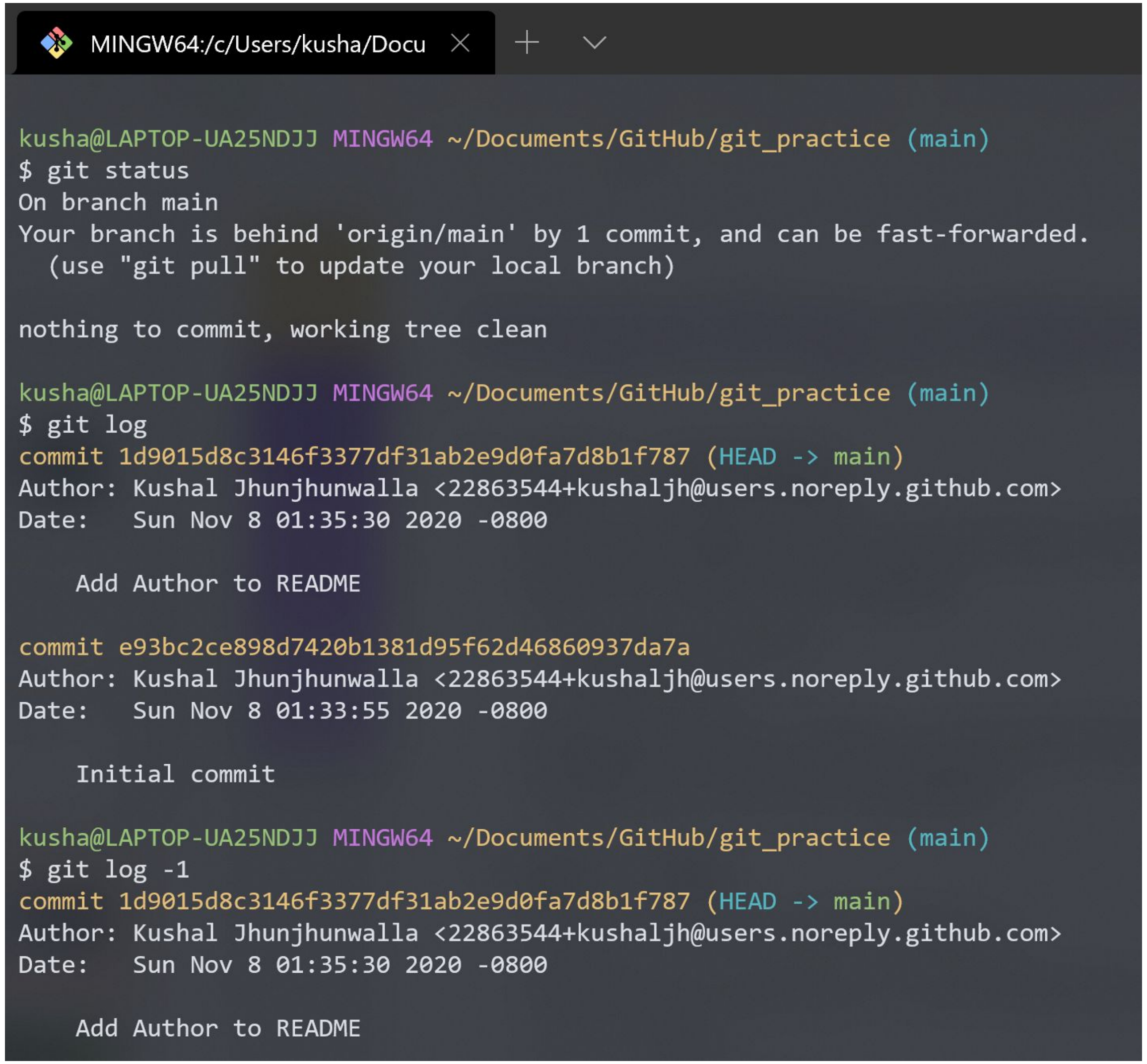
```
git add .
```

```
git commit -m “message”
```

```
git init
```

Inspecting a repository

- `git status`
  - Lists the files which you have changed but not yet committed
    - Working directory
    - Staging area
  - Indicates how many commits have made but not yet pushed
- `git log`
  - Shows the commit history
  - Option to see last n commits by adding –<n> flag
  - Press "q" to exit



```
kusha@LAPTOP-UA25NDJJ MINGW64 ~/Documents/GitHub/git_practice (main)
$ git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

kusha@LAPTOP-UA25NDJJ MINGW64 ~/Documents/GitHub/git_practice (main)
$ git log
commit 1d9015d8c3146f3377df31ab2e9d0fa7d8b1f787 (HEAD -> main)
Author: Kushal Jhunjhunwalla <22863544+kushaljh@users.noreply.github.com>
Date:   Sun Nov 8 01:35:30 2020 -0800

    Add Author to README

commit e93bc2ce898d7420b1381d95f62d46860937da7a
Author: Kushal Jhunjhunwalla <22863544+kushaljh@users.noreply.github.com>
Date:   Sun Nov 8 01:33:55 2020 -0800

    Initial commit

kusha@LAPTOP-UA25NDJJ MINGW64 ~/Documents/GitHub/git_practice (main)
$ git log -1
commit 1d9015d8c3146f3377df31ab2e9d0fa7d8b1f787 (HEAD -> main)
Author: Kushal Jhunjhunwalla <22863544+kushaljh@users.noreply.github.com>
Date:   Sun Nov 8 01:35:30 2020 -0800

    Add Author to README
```

Working with `git` locally | DEMO

Reference: Demo commands

```
mkdir test_git
cd test_git
git status
git init
touch file.txt
git status
git add file.txt
git status
git commit -m "Add first file"
git status
```

Working with remote

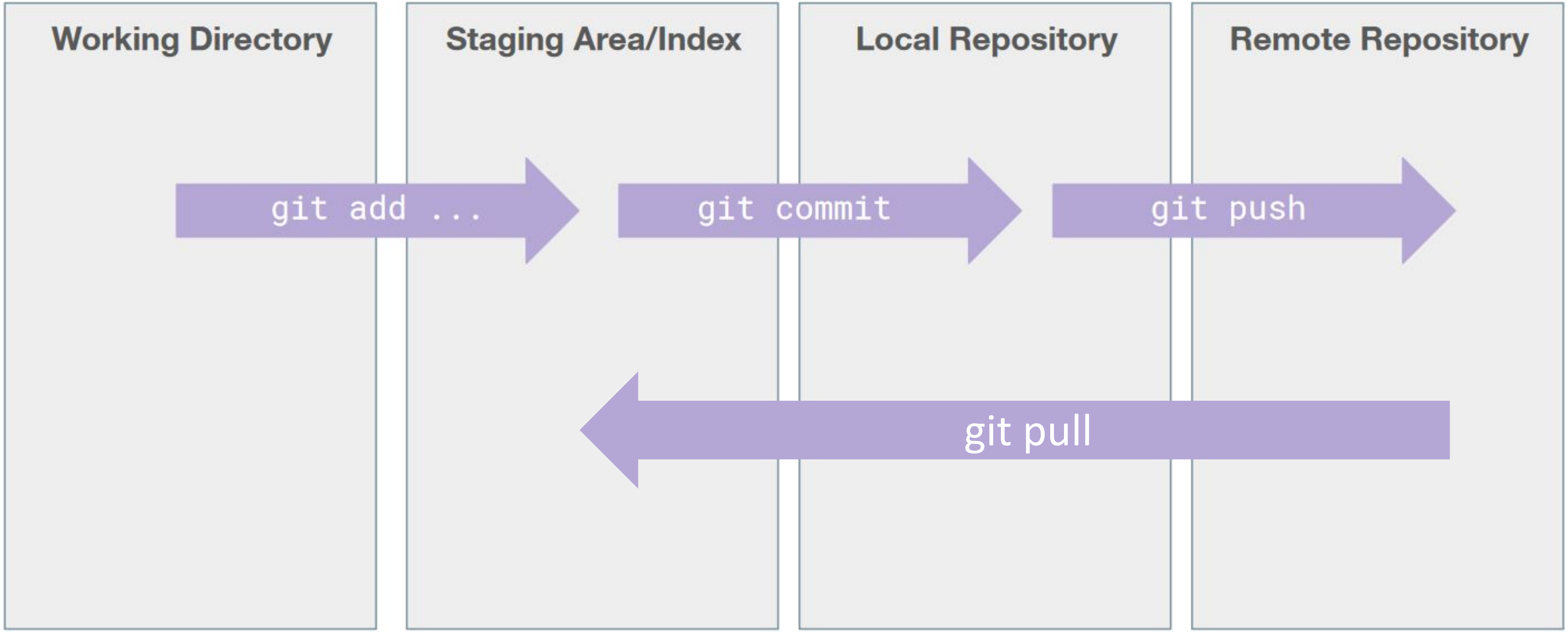
`git` commands for interaction with remote

- **`git clone`**
  - Cloning is the process of creating a working copy of the remote or local repository by passing the following command.
  - **`git clone username@git_server_hostname:/path_of_repository`**

- **`git pull`**
  - If we have already cloned the repository and need to update local (only code) respect to the remote server
  - **`git pull origin main`**

- **`git fetch`**
  - Fetching is the process of updating (only git information) the local git structure and information from remote repository
  - **`git fetch`**

Updating changes using pull



NOTE: There are way more git commands than what is listed here - this is a simplified model to get us started.

What's next!

- Branching, checkout
- Merging, Merge (Pull) Requests
- Conflicts
- Interacting with a Git Server (GitHub / GitLab)

Complete working (including fork)

Useful resources

- Try Git (resources and tutorial)
- The Git Cheat Sheet
- Stack Overflow's definitive guide for beginners
- When you are terribly stuck with git
  - DO NOT PANIC! Even experienced developers get stuck with git issues
  - https://ohshitgit.com/
  - https://stackoverflow.com/questions/tagged/git