sourcemind

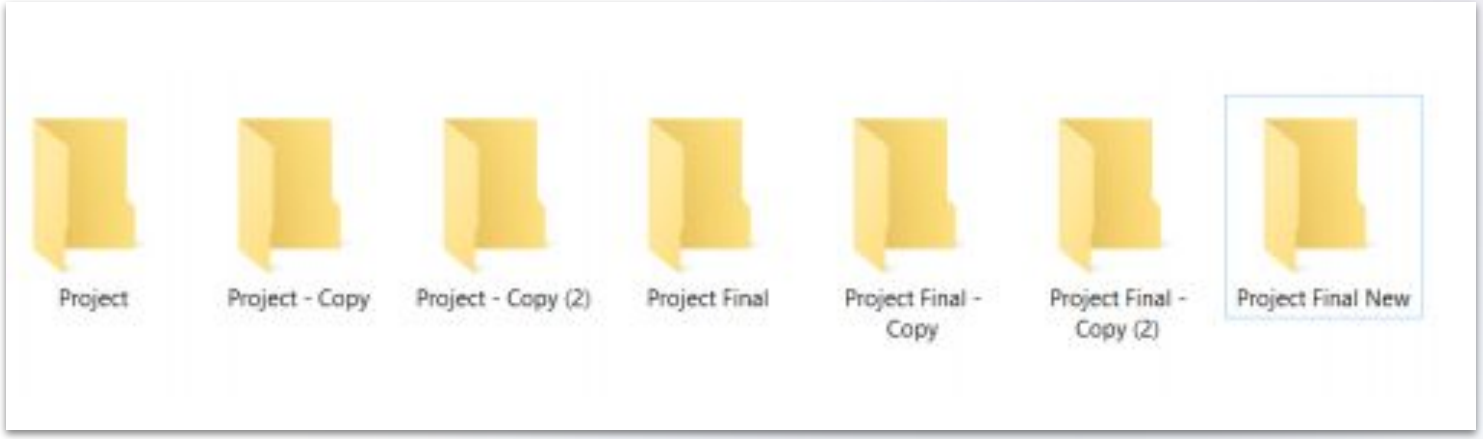
Git Basics

Artur Davtyan
DevOps Engineer
Webb Fontaine Armenia

sourcemind

How do you track the changes to project ?

- Make a million copies by backing up every time you feel you are going to change something big.
- Try to CTRL-Z to the times of dinosaurs until you revert all your changes.
- What if you must maintain multiple versions of a product?



sourcemind

What is Version Control ?

Version Control Allows you to

- ➔ *Restore a specific version*
- ➔ *Compare the changes over time*
- ➔ *Collaborate*
- ➔ *See who modified something*
- ➔ *Tag and version*

sourcemind

Version Control System Types

Version Control are available of three types

1. *Local Version Control System*
2. *Centralized Version Control System*
3. *Distributed Version Control System*

sourcemind

Local Version Control Systems

Everything is stored on local computers. Developers can copy folders and attach timestamps to them or develop simple databases to keep track of changes to the files.

Advantage:

- ➔ *Very simple*

Disadvantage:

- ➔ *Error Prone*
- ➔ *No backups or manual backups:*
- ➔ *No collaboration mechanisms*

sourcemind

Centralized Version Control Systems

These systems have a single server that contains all the versioned files and several clients that check out files from that central place. Examples of centralized VCS are CVS, Subversion and Perforce.

Advantage:

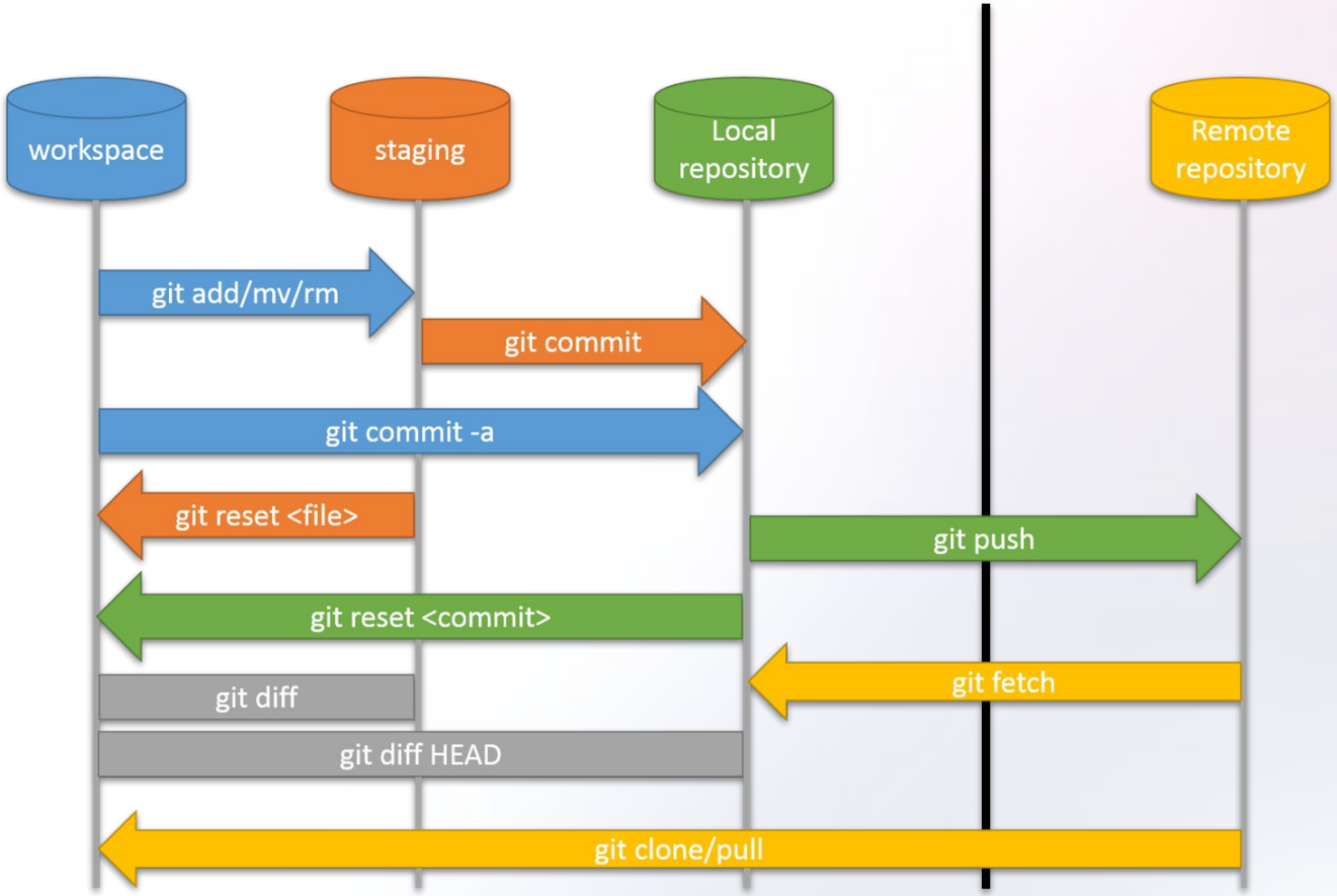
- ➔ *Everyone knows to a certain degree what everyone else on the project is doing.*
- ➔ *Fine-grained control over who can do what.*
- ➔ *The system does not force you to download the whole history, so if the project has a huge history it won't take much time to download it.*
- ➔ *The system does not force you to download all the files so if the project contains many big files, the user can choose which ones to download*

Disadvantage:

- ➔ *If server goes down, no one can collaborate.*
- ➔ *If the disk on which the central database is stored becomes corrupted and there are no backups, you lose absolutely everything.*
- ➔ *You need a connection to the central server for all the action*

sourcemind

Git Architecture



sourcemind

Git Installation And Environment Set Up

Installing git:

➢ *Open https://git-scm.com/downloads and find the version for your system*

For Ubuntu:

```
adavtyan@artur-lpt:~$ sudo apt install -y git
...
adavtyan@artur-lpt:~$ git --version to verify
git version 2.25.1
```

sourcemind

Post Installation And Configuration git

- ➔ git config --global user.name “Artur Davtyan"
- ➔ git config --global user.email arturdavtyan1995@gmail.com
- ➔ sudo git config --system core.editor vim
- ➔ git config –l

- ➔ --global for editing user-specific configurations. Makes changes to ~/.gitconfig file.
- ➔ --system for editing system-specific configurations. Makes changes to /etc/gitconfig file.
- ➔ --local for editing repository-specific configurations. Makes changes to .git/config file.

sourcemind

What is Bitbucket ?

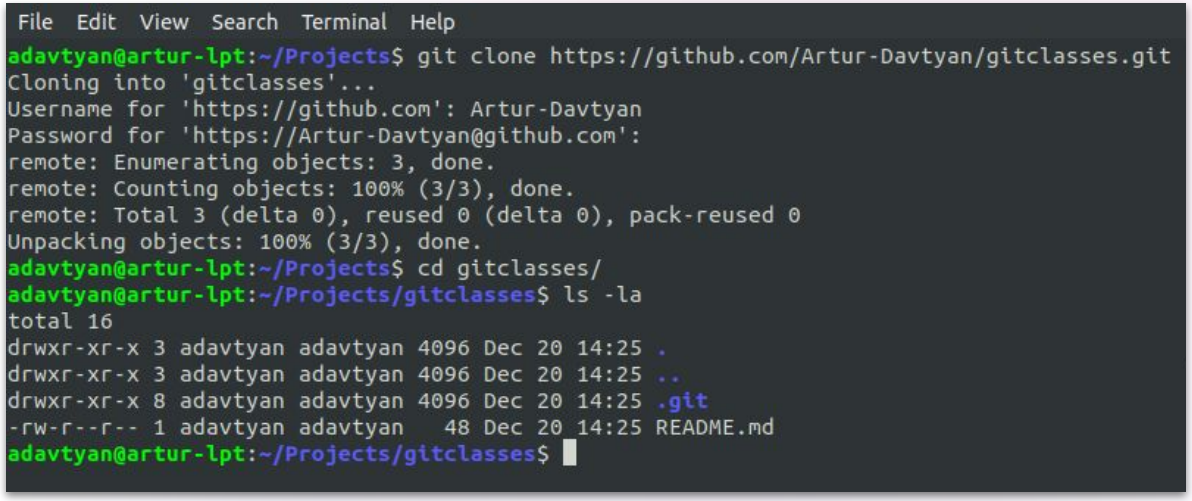
- GitHub is a service for that hosts git repositories.
- It provides web-based GUI for interacting with the repositories.
- Some of the famous alternatives are BitBucket and GitLab.

sourcemind

Cloning a bitbucket Repository and Status

Git clone:

- Open the terminal and type git clone
- Git will create a new directory, name it as the remote repository (this is going to be the working directory) and download the remote repository there (making it your local repository).
- Cd to that directory to see that all the files from remote repository are now in that directory+



Git status:

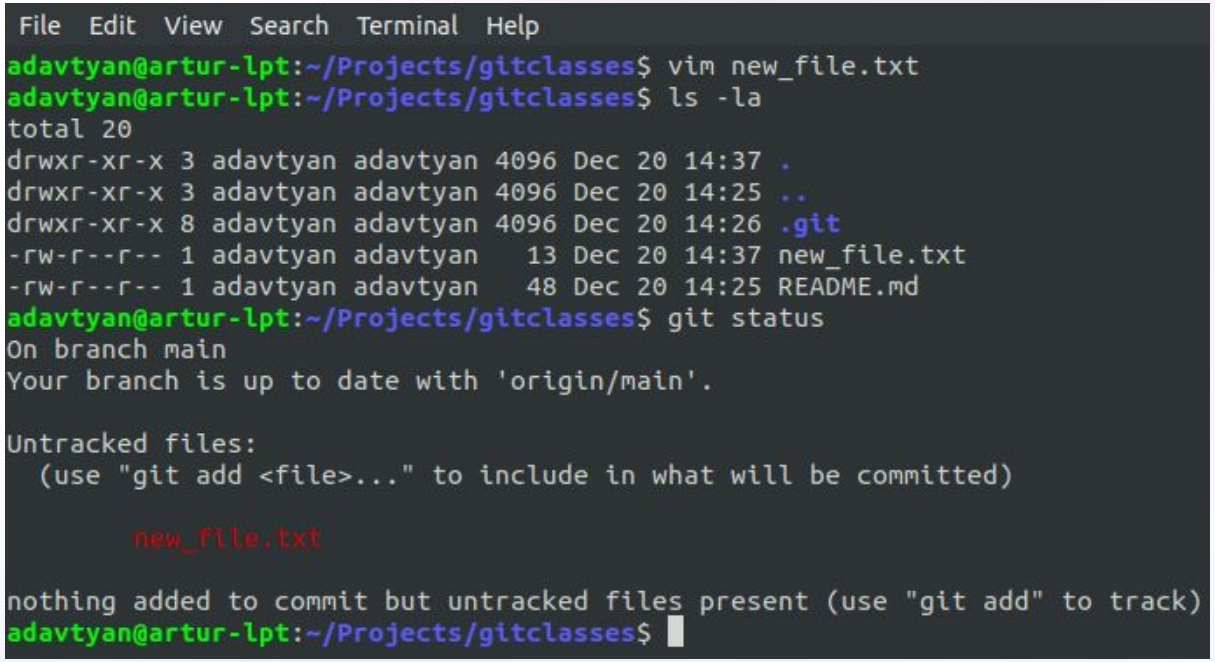
- The state of the working directory and the staging area
- Current branch
- The state compared to the remote repository

sourcemind

Adding new Files

Adding new file:

- Open your favorite text editor and add a new file with some content in it. Save and go back to the terminal and run ***git status***.
- Git detected a new file in the ***working directory***, but it is not added to the ***local repository***, that's why it is listed in ***Untracked files*** section.

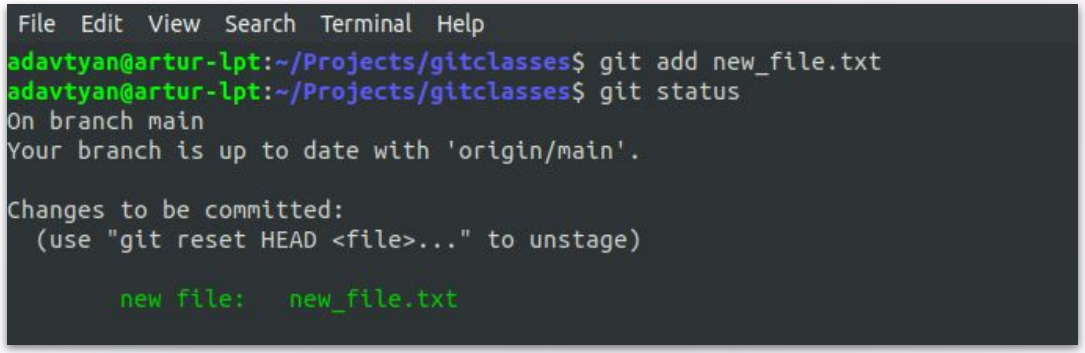


sourcemind

Staging a new file

Staging a new file:

- To tell git that we want to add the file to the ***local repository*** we first need to ***add*** the file to the ***staging area.***
- Run ***git add << file name>>*** and then ***git status*** again.
- You can see that the file is green now. That means that it is added to the ***staging area.***



sourcemind

Commit a new file

Committing a new file:

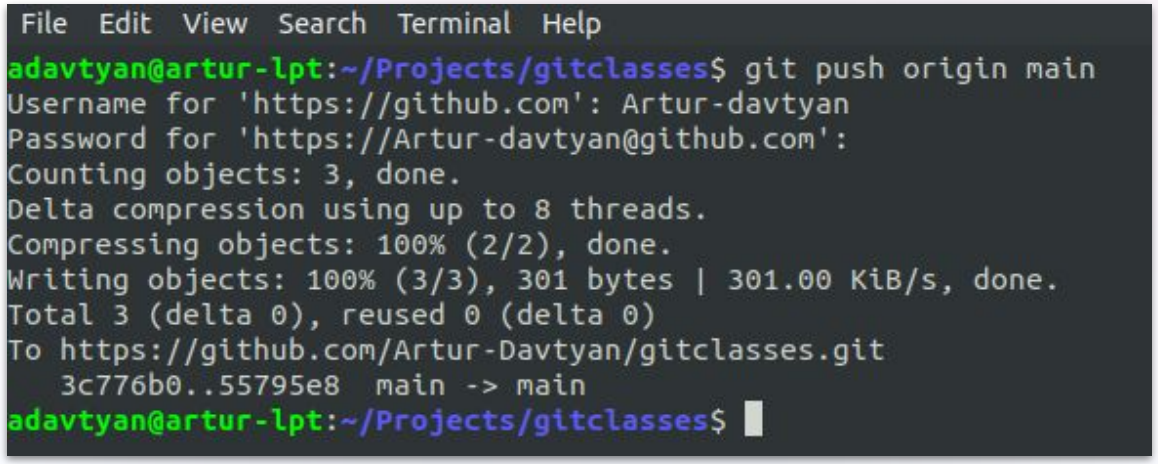
- To tell git that we want to add the staged changes to the local repository we need to commit the changes.
- Every commit should have a message/description associated with it. It helps us see the history of changes in a human-readable way.
- Run git commit -m “” and then git status again.
- You can see that we have nothing to commit. And now our local repository is ahead of ‘origin/master’ by 1 commit.
- The default name for the remote repository is origin. And the default branch name is master. We will come back to branches later

sourcemind

Pushing changes

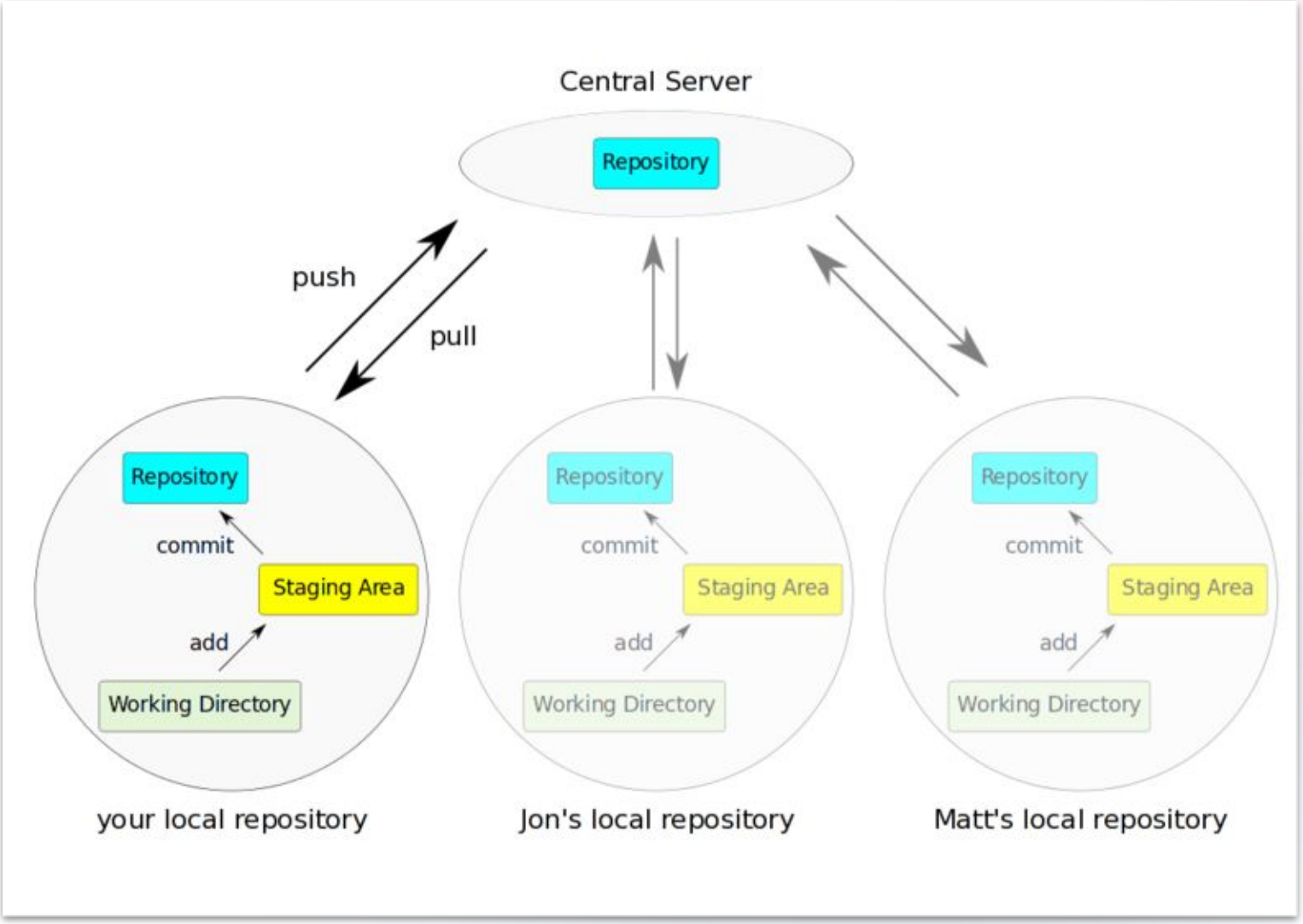
Pushing new Change:

- To tell git that we want to share our changes with everyone, we should push our local repository to the remote repository.
- Run git push origin master and then git status.
- Our local repository is now up to date with the remote.



sourcemind

Pushing changes

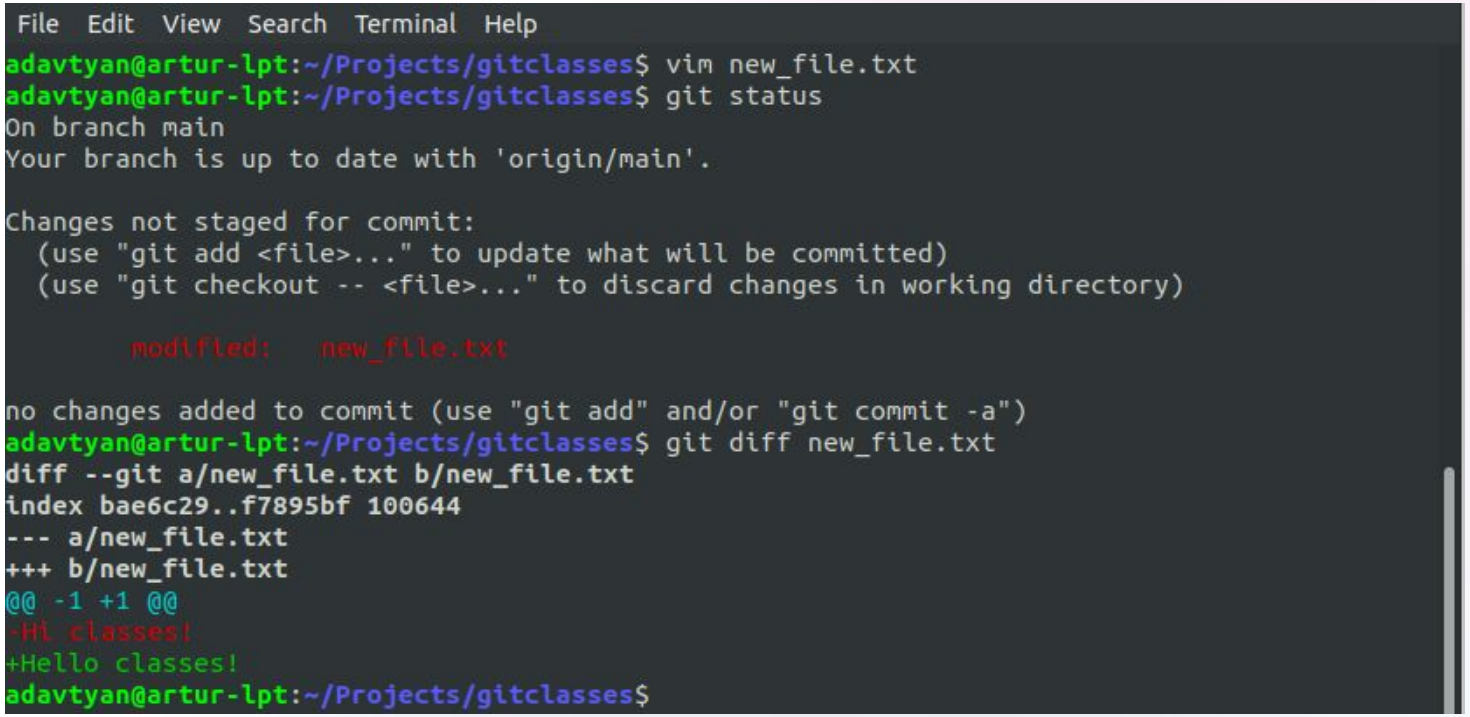


sourcemind

Making changes to files

Making changes to tracked files:

- Open your favorite editor and change the contents of the newly created file.
- Save, go back in terminal and run git status.
- You can see the changes with git diff command.
- The file is now tracked, and git shows it as modified. We should add the file to staging area again to stage the changes.

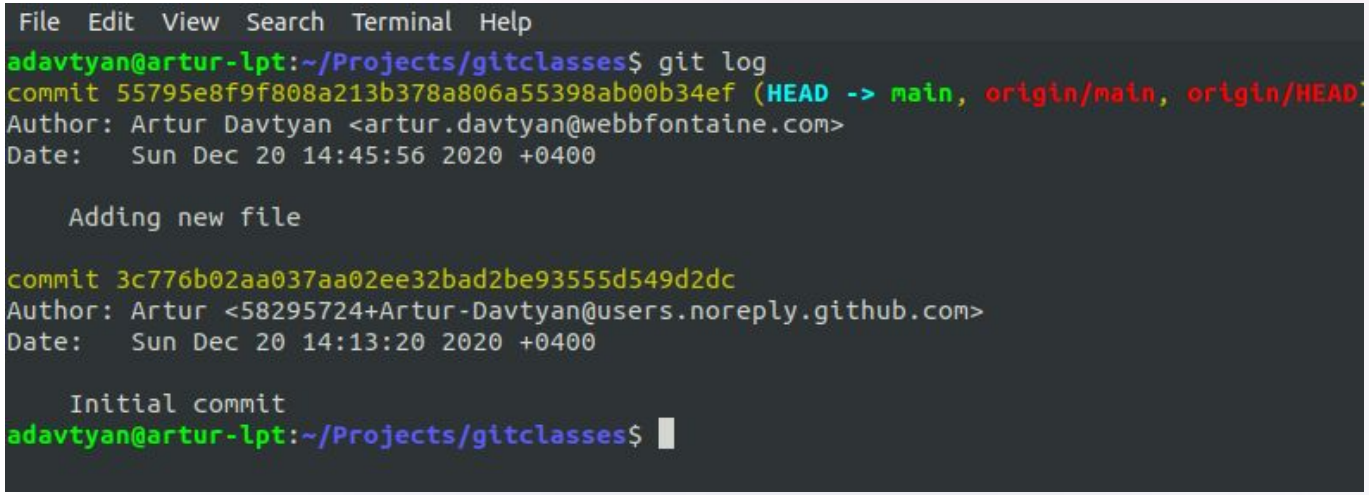


sourcemind

See the commit history

Git log:

- git log show the history of commits.

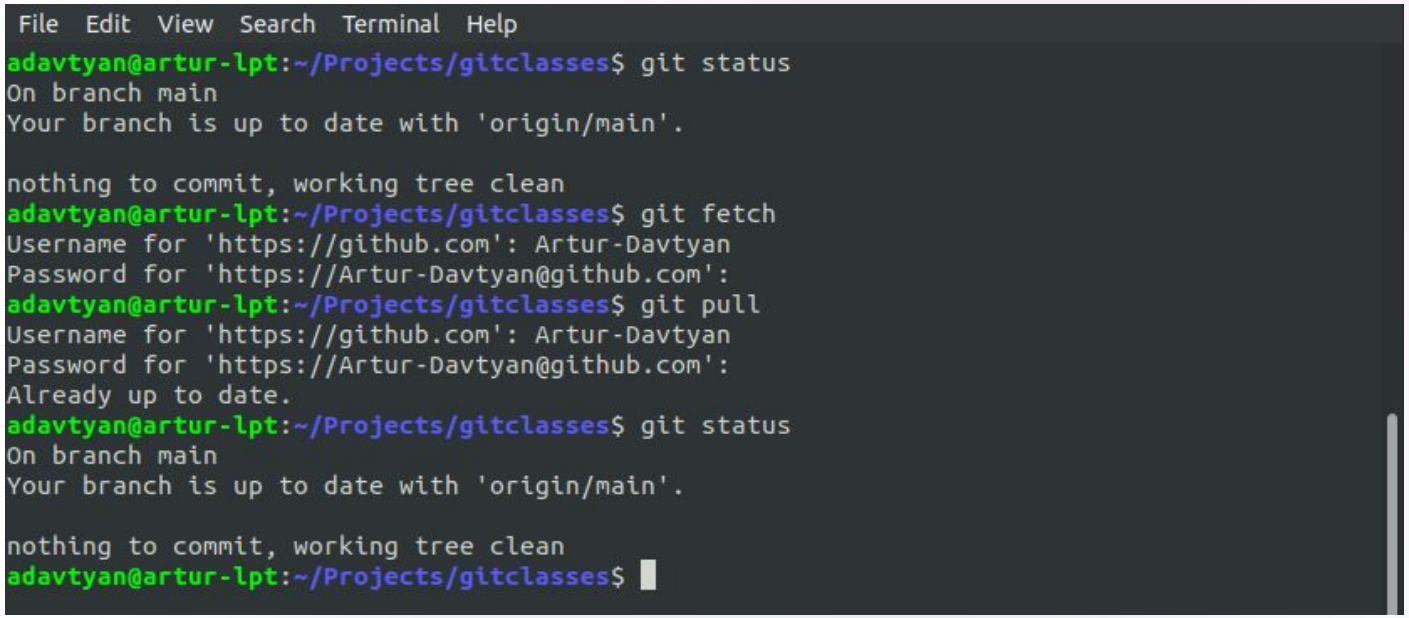


sourcemind

Pulling the changes from the remote repostirory

Fetching and pulling:

- There are changes in the remote that we don't have in our local repo.
- ***git status*** won't show anything as it doesn't know about remote changes yet.
- ***git fetch*** will download the remote changes but not merge them with your local repository.
- ***git pull*** will download the remote changes and merge them with your local repository

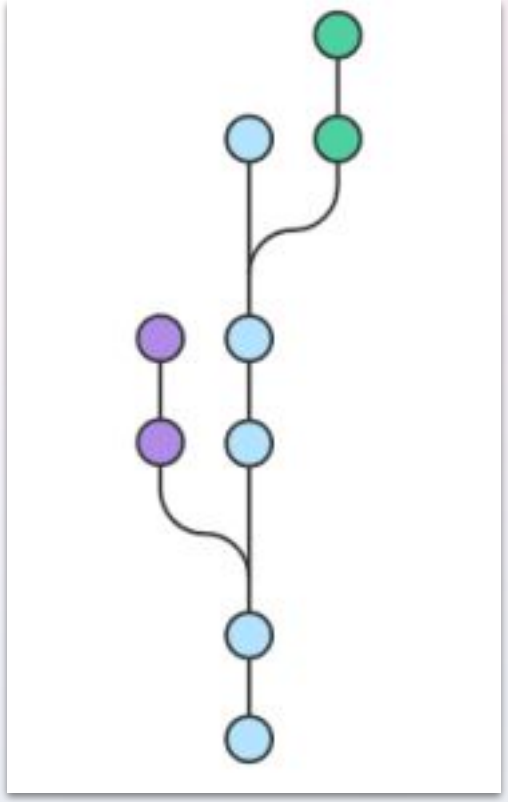


sourcemind

Branching

Git branches:

- A branch represents an independent line of development.
- You can think of them as a way to request a brand-new working directory, staging area, and project history.
- New commits are recorded in the history for the current branch, which results in a fork in the history of the project.
- You can merge your branch back to the source branch.

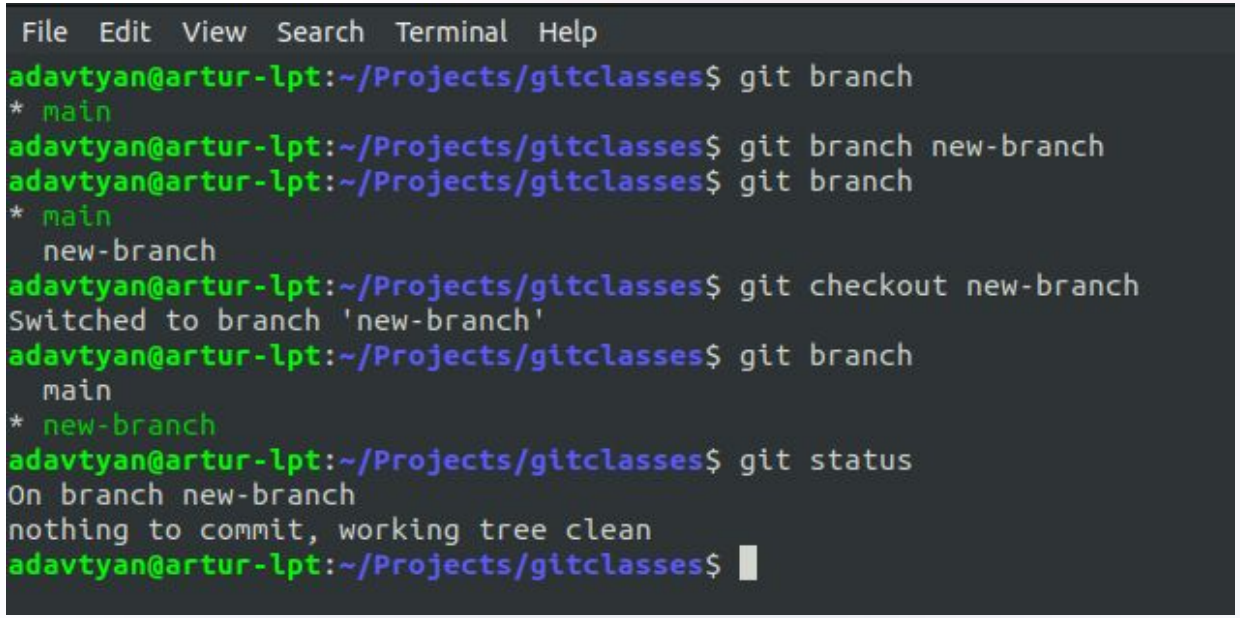


sourcemind

Branching

Git branches - Commands

- ***git branch*** - lists all local branches.
- ***git branch <branch name>*** - Creates new branch
- ***git checkout*** - Switches to the new branch
- ***git checkout -b*** - Creates a branch and switches to it.



sourcemind

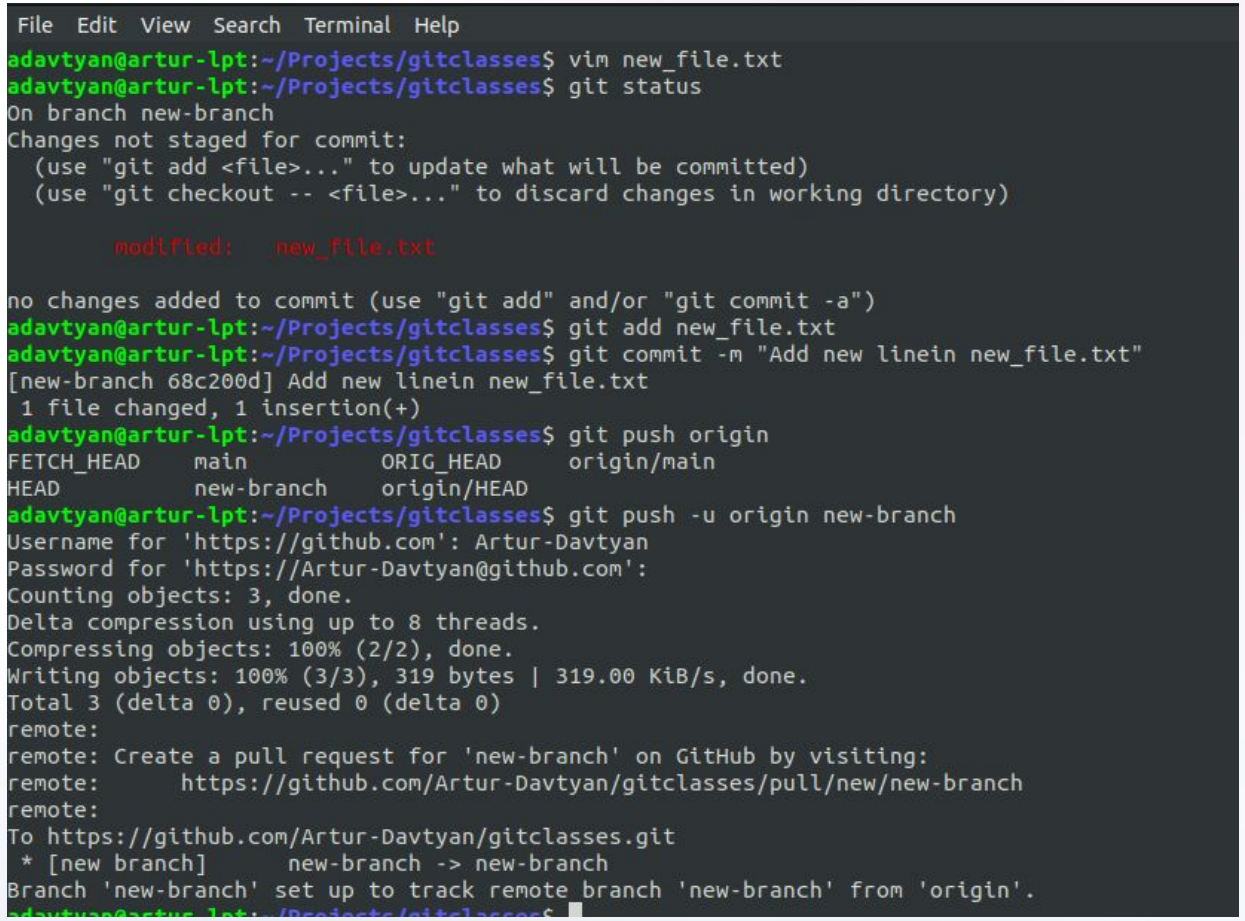
Branching

Making changes in a new branch:

- Open your favorite editor and change the contents of the newly created file.
- Save, go back in terminal, stage and commit the changes

Pushing the new branch to the remote repository:

- When we cloned the repository it only had one branch – ***main*** and it created a local branch ***main*** for our local repository.
- Remote ***main*** branch is the ***upstream*** of our local ***main*** branch.
- Our newly created ***new-branch*** branch does not have an ***upstream*** branch so we tell git to create one with ***-u/--set-upstream*** option.



sourcemind

Good news!

1. *git init*
2. *git status*
3. *git log*
4. *git add*
5. *git commit*
6. *git diff*
7. *git rm*
8. *git mv*
9. *git push*
10. *git pull*

85% of the time you'll be using only these commands

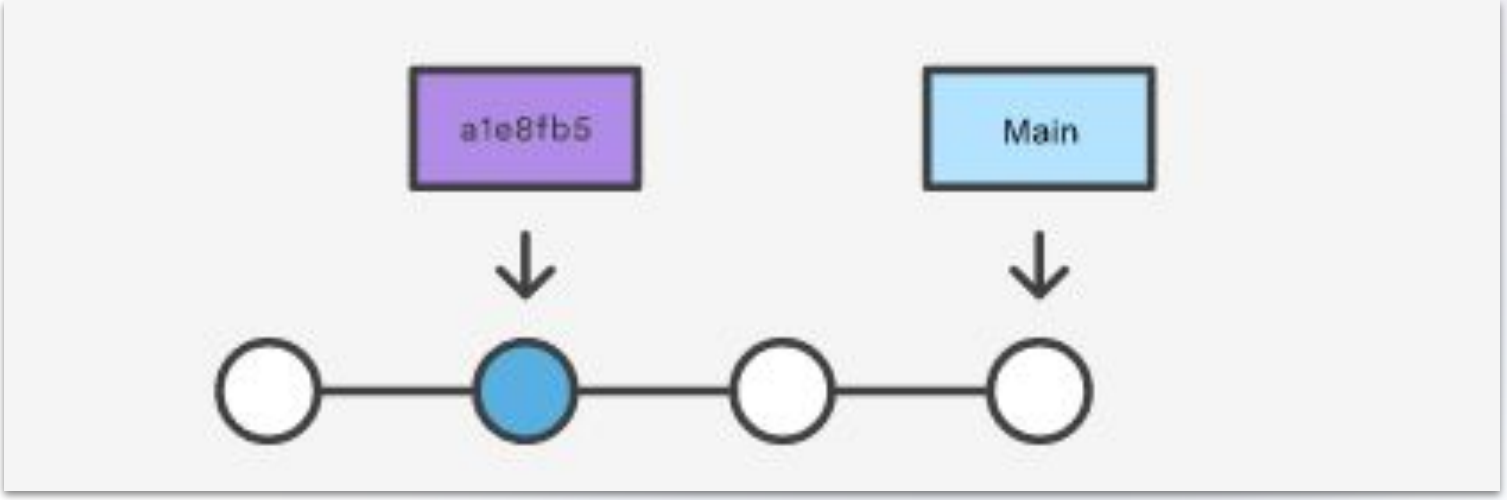
sourcemind

Undoing Commits and Changes

Reviewing old commits

➔ git log --oneline

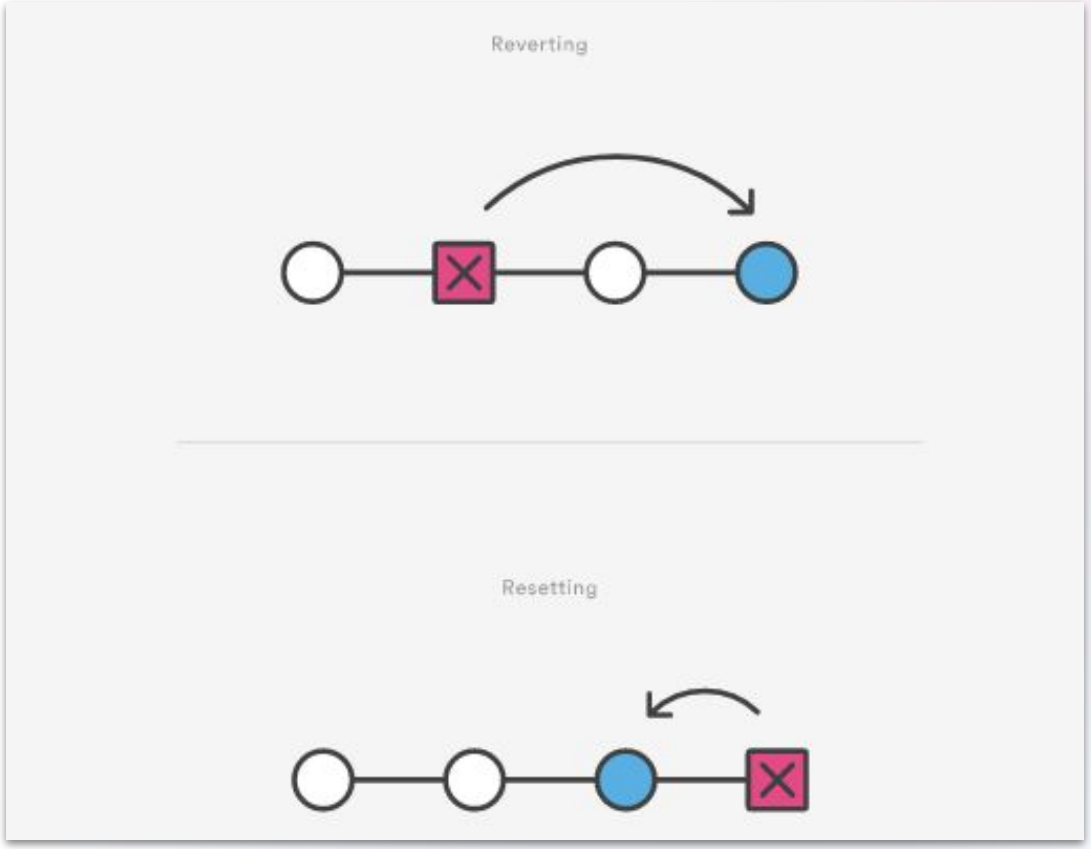
When you have found a commit reference to the point in history you want to visit, you can utilize the *git checkout* command to visit that commit. Git checkout is an easy way to “load” any of these saved snapshots onto your development machine.



sourcemind

Resetting vs Reverting

It's important to understand that git revert undoes a single commit—it does not "revert" back to the previous state of a project by removing all subsequent commits. In Git, this is actually called a git reset, not a revert.



sourcemind

Thank you for your attention !

Q&A