Lab 6 – Git and GitHub

1 NET2008 – DevOps – Fall 2025 - Lab 6

2 Git and Github

2.1 Step 0: Install git and create a GitHub account

The first two things you'll want to do are install git and create a free GitHub account.

Follow the instructions here to install git (if it's not already installed). Note that for this tutorial we will be using git on the command line only. While there are some great git GUIs (graphical user interfaces), I think it's easier to learn git using git-specific commands first and then to try out a git GUI once you're more comfortable with the command.

Once you've done that, create a GitHub account here. (Accounts are free for public repositories and personal private repositories.)

2.2 Prep Task

git settings are found in the .gitconfig file in your home directory. This file is an INI-style file, and you can edit it either using any text editor or using the git config command. Here, we'll show you how to use the git config command to set this information.

Github changed the default branch from `master` to `main`. Make sure to use `main` instead of `master`.

2.2.1 Changing the default branch from master to main

```
$ git config --global init.defaultBranch main
```

2.2.2 Setting git default

```
$ git config --global user.name "John Smith" $ git config --global user.email
"john.smith@gmail.com"
```

With the –global flag, git config modifies the .gitconfig file in your home directory; without it, git config modifies the .git/config file of the current repository.

2.3 Step 1: Create a local git repository

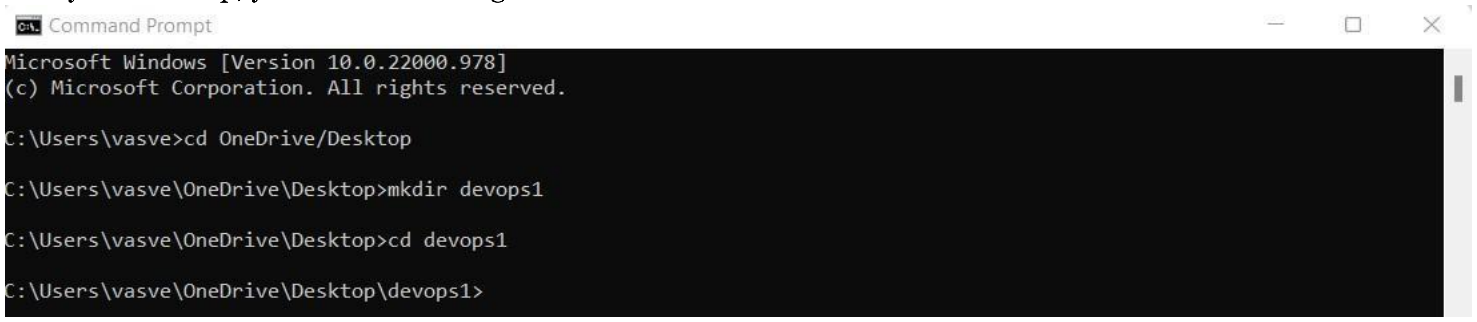
When creating a new project on your local machine using git, you'll first create a new **repository** (or often, **repo**, for short).

To use git we'll be using the terminal. If you don't have much experience with the terminal and basic commands, check out the following tutorials:

1. **Windows OS**
2. **Mac OS**

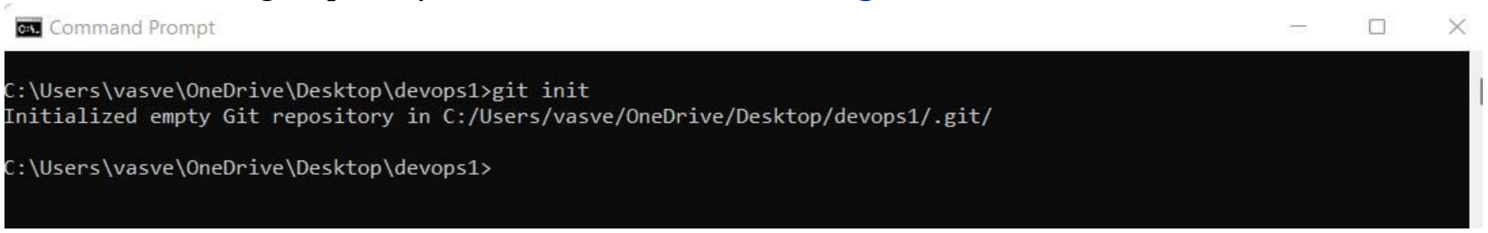
3. Linux OS

To begin, open up a terminal and move to where you want to place the project on your local machine using the `cd` (change directory) command. For example, if you have a 'projects' folder on your desktop, you'd do something like:


Command Prompt

```
Microsoft Windows [Version 10.0.22000.978]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vasve>cd OneDrive/Desktop

C:\Users\vasve\OneDrive\Desktop>mkdir devops1

C:\Users\vasve\OneDrive\Desktop>cd devops1

C:\Users\vasve\OneDrive\Desktop\devops1>
```

To initialize a git repository in the root of the folder, run the **git init** command:

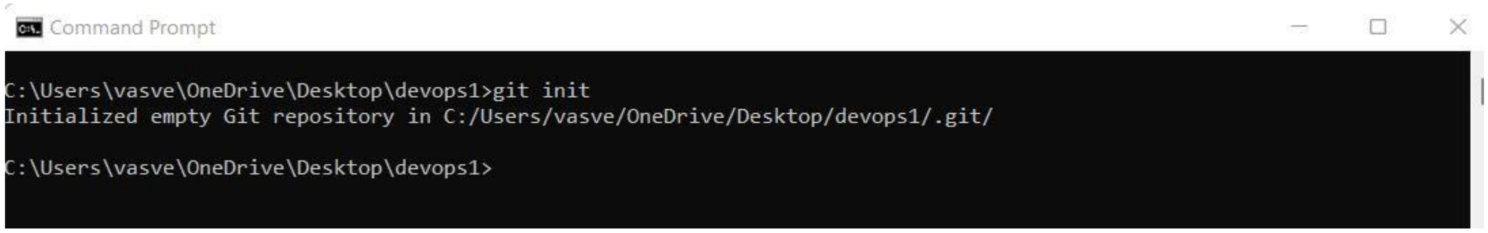

Command Prompt

```
C:\Users\vasve\OneDrive\Desktop\devops1>git init
Initialized empty Git repository in C:/Users/vasve/OneDrive/Desktop/devops1/.git/

C:\Users\vasve\OneDrive\Desktop\devops1>
```

2.4 Step 2: Add a new file to the repo

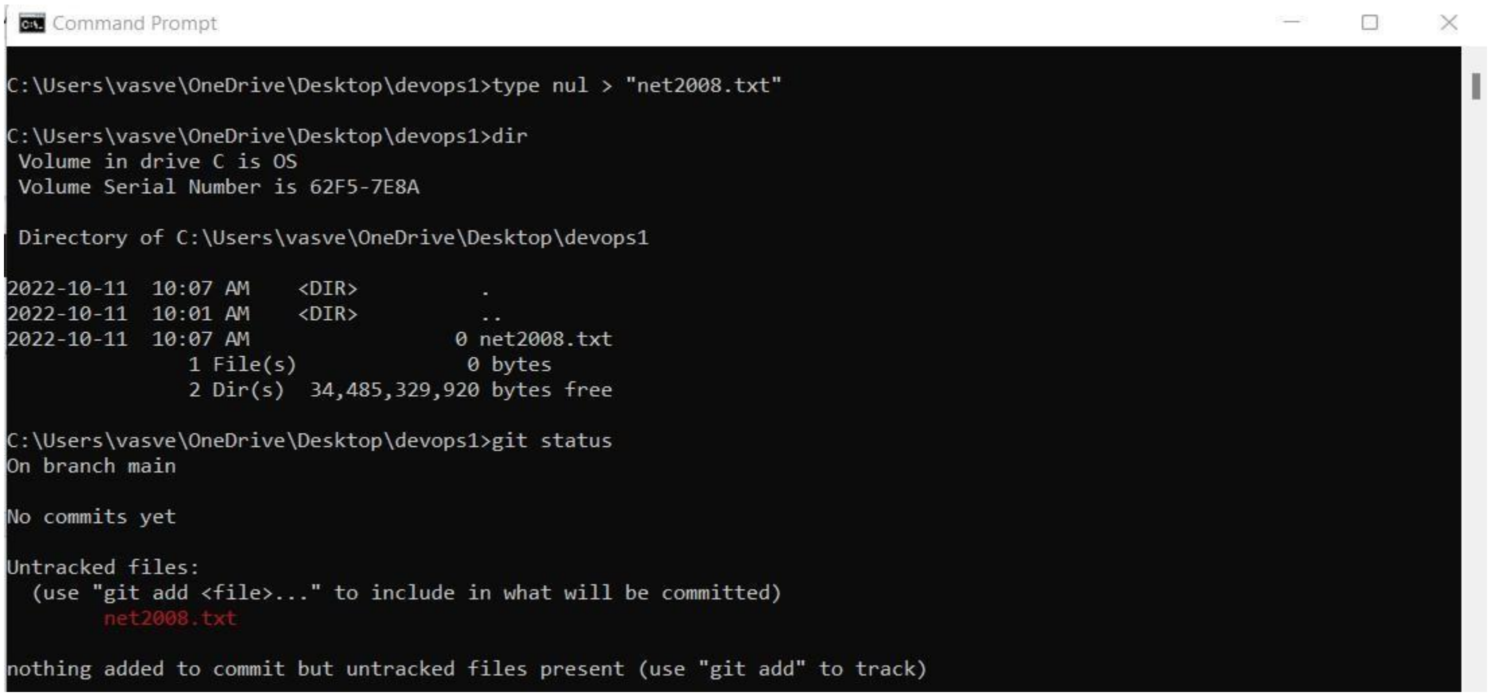
Go ahead and add a new file to the project, using any text editor you like or running a touch command (For Mac OS)or running type command (For Windows OS).

Once you've added or modified files in a folder containing a git repo, git will notice that changes have been made inside the repo. But, git won't officially keep track of the file (that is, put it in a commit - we'll talk more about commits next) unless you explicitly tell it to.


Command Prompt

```
C:\Users\vasve\OneDrive\Desktop\devops1>git init
Initialized empty Git repository in C:/Users/vasve/OneDrive/Desktop/devops1/.git/

C:\Users\vasve\OneDrive\Desktop\devops1>
```

After creating the new file, you can use the **git status** command to see which files git knows exist.



```
C:\Users\vasve\OneDrive\Desktop\devops1>type nul > "net2008.txt"

C:\Users\vasve\OneDrive\Desktop\devops1>dir
 Volume in drive C is OS
 Volume Serial Number is 62F5-7E8A

 Directory of C:\Users\vasve\OneDrive\Desktop\devops1

2022-10-11  10:07 AM    <DIR>          .
2022-10-11  10:01 AM    <DIR>          ..
2022-10-11  10:07 AM                 0 net2008.txt
               1 File(s)              0 bytes
               2 Dir(s)  34,485,329,920 bytes free

C:\Users\vasve\OneDrive\Desktop\devops1>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        net2008.txt

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

What this basically says is, “Hey, we noticed you created a new file called `net2008.txt`, but unless you use the ’`git add`’ command we aren’t going to do anything with it.”

2.4.1 An interlude: The staging environment, the commit, and you

One of the most confusing parts when you’re first learning git is the concept of the staging environment and how it relates to a commit.

A **commit** is a record of what files you have changed since the last time you made a commit. Essentially, you make changes to your repo (for example, adding a file or modifying one) and then tell git to put those files into a commit.

Commits make up the essence of your project and allow you to go back to the state of a project at any point.

So, how do you tell git which files to put into a commit? This is where the **staging environment** or **index** come in. As seen in Step 2, when you make changes to your repo, git notices that a file has changed but won’t do anything with it (like adding it in a commit).

To add a file to a commit, you first need to add it to the staging environment. To do this, you can use the **git add** command (see Step 3 below).

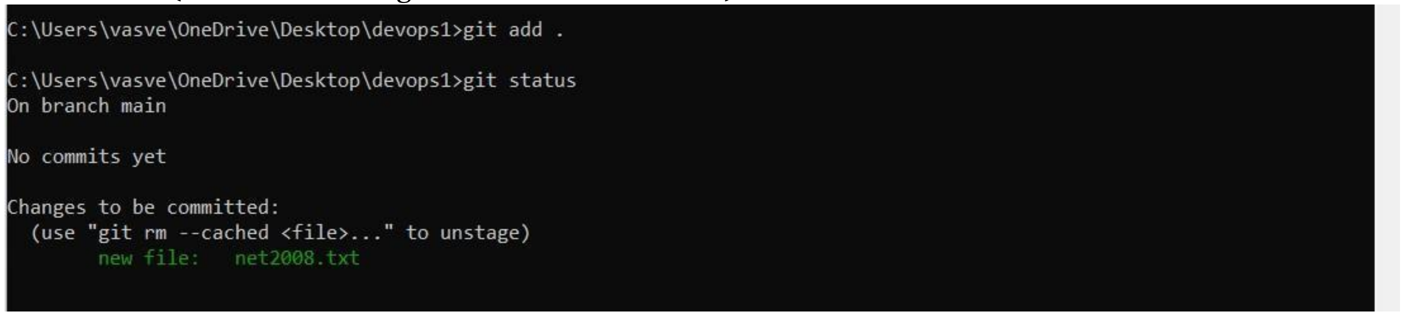
Once you’ve used the git add command to add all the files you want to the staging environment, you can then tell git to package them into a commit using the **git commit** command.

Note: The staging environment, also called ‘staging’, is the new preferred term for this, but you can also see it referred to as the ‘index’.

2.5 Step 3: Add a file to the staging environment

Add a file to the staging environment using the **git add** command.

If you rerun the git status command, you'll see that git has added the file to the staging environment (notice the "Changes to be committed" line).



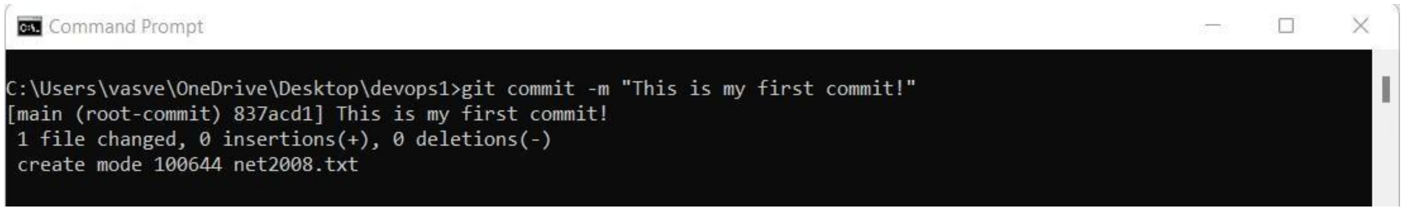
```
C:\Users\vasve\OneDrive\Desktop\devops1>git add .

C:\Users\vasve\OneDrive\Desktop\devops1>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   net2008.txt
```

To reiterate, the file has **not** yet been added to a commit, but it's about to be.

2.6 Step 4: Create a commit

It's time to create your first commit!

Run the command `git commit -m "This is my first commit!"`



```
C:\Users\vasve\OneDrive\Desktop\devops1>git commit -m "This is my first commit!"
[main (root-commit) 837acd1] This is my first commit!
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 net2008.txt
```

The message at the end of the commit should be something related to what the commit contains - maybe it's a new feature, maybe it's a bug fix, maybe it's just fixing a typo. Don't put a message like "asdfadsf" or "foobar". That makes the other people who see your commit sad. Very, very, sad.

2.7 Step 5: Create a new branch

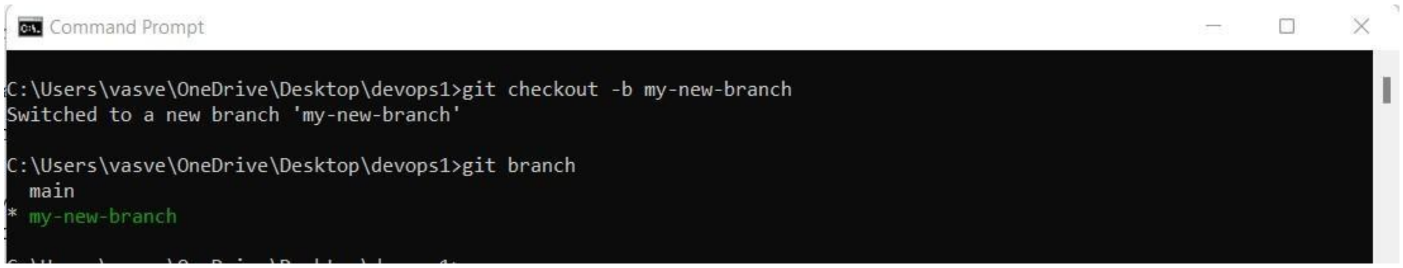
Now that you've made a new commit, let's try something a little more advanced.

Say you want to make a new feature but are worried about making changes to the main project while developing the feature. This is where **git branches** come in.

Branches allow you to move back and forth between 'states' of a project. For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you're done with the page, you can **merge** your changes from your branch into the main branch. When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files.

Let's say you are on the main branch and want to create a new branch to develop your web page. Here's what you'll do: Run **git checkout -b my-new-branch**. This command will automatically create a new branch and then 'check you out' on it, meaning git will move you to that branch, off of the main branch.

After running the above command, you can use the **git branch** command to confirm that your branch was created:



```
C:\Users\vasve\OneDrive\Desktop\devops1>git checkout -b my-new-branch
Switched to a new branch 'my-new-branch'

C:\Users\vasve\OneDrive\Desktop\devops1>git branch
  main
* my-new-branch
```

The branch name with the asterisk next to it indicates which branch you're pointed to at that given time.

Now, if you switch back to the main branch and make some more commits, your new branch won't see any of those changes until you **merge** those changes onto your new branch.

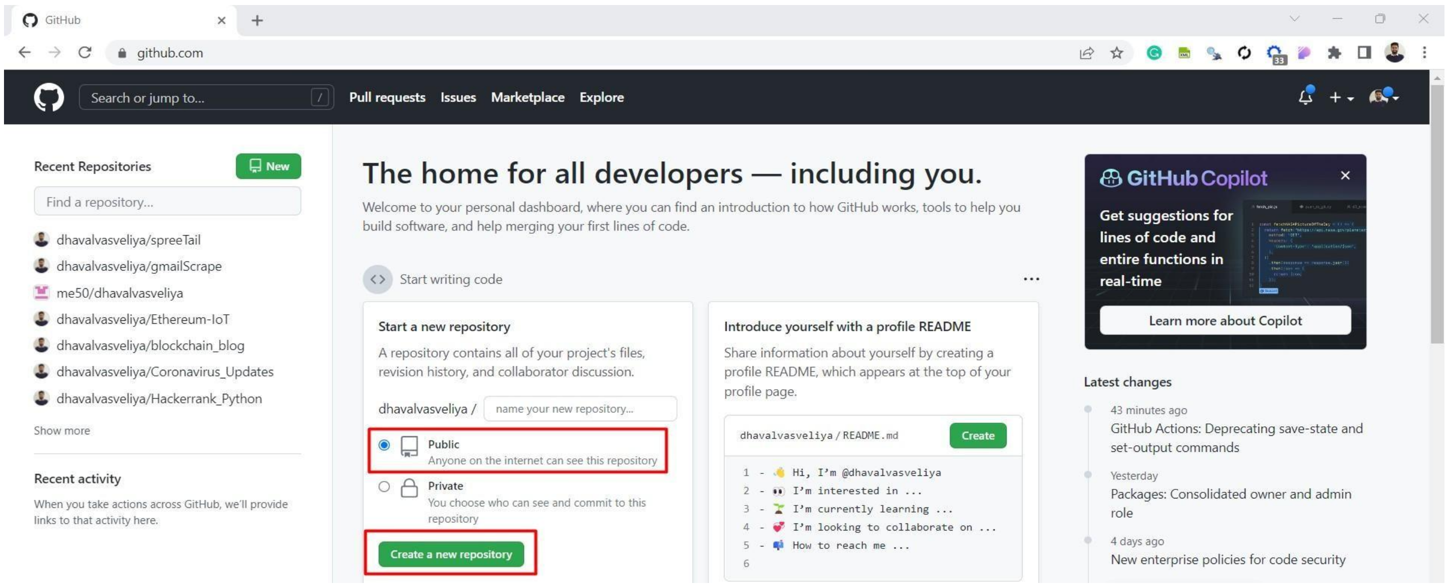
2.8 Step 6: Create a Personal Access Token (PAT)

https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

2.9 Step 7: Create a new repository on GitHub

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code.

To create a new repo on GitHub, log in and go to the GitHub home page. You should see a green 'Create a new' button:



The home for all developers — including you.

Welcome to your personal dashboard, where you can find an introduction to how GitHub works, tools to help you build software, and help merging your first lines of code.

A repository contains all of your project's files, revision history, and collaborator discussion.

Anyone on the internet can see this repository

You choose who can see and commit to this repository

Share information about yourself by creating a profile README, which appears at the top of your profile page.

```
- 👋 Hi, I'm @dhavalvasveliya
- 👀 I'm interested in ...
- 🌱 I'm currently learning ...
- 💞️ I'm looking to collaborate on ...
- 📫 How to reach me ...
```

When you take actions across GitHub, we'll provide links to that activity here.

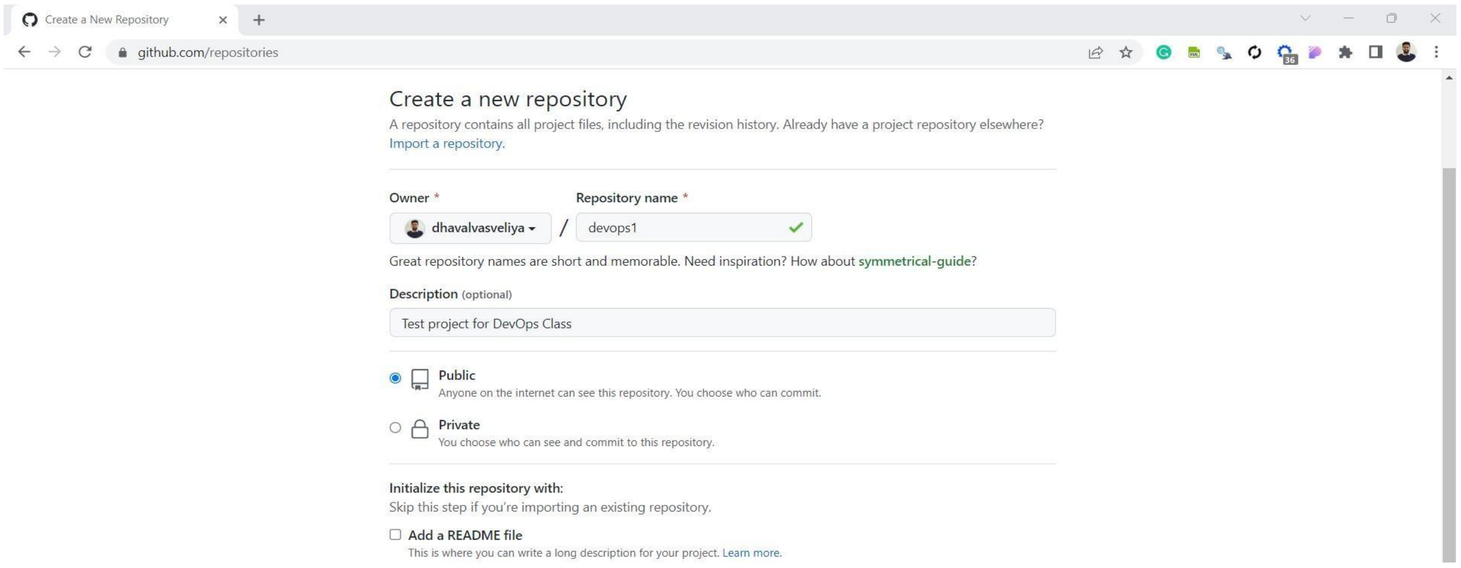
Get suggestions for lines of code and entire functions in real-time

43 minutes ago
GitHub Actions: Deprecating save-state and set-output commands

Yesterday
Packages: Consolidated owner and admin role

4 days ago
New enterprise policies for code security

After clicking the button, GitHub will ask you to name your repo and provide a brief description:


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: dhavalvasveliya / Repository name: devops1

Great repository names are short and memorable. Need inspiration? How about symmetrical-guide?

Description (optional): Test project for DevOps Class

Public [✓]
Anyone on the internet can see this repository. You choose who can commit.

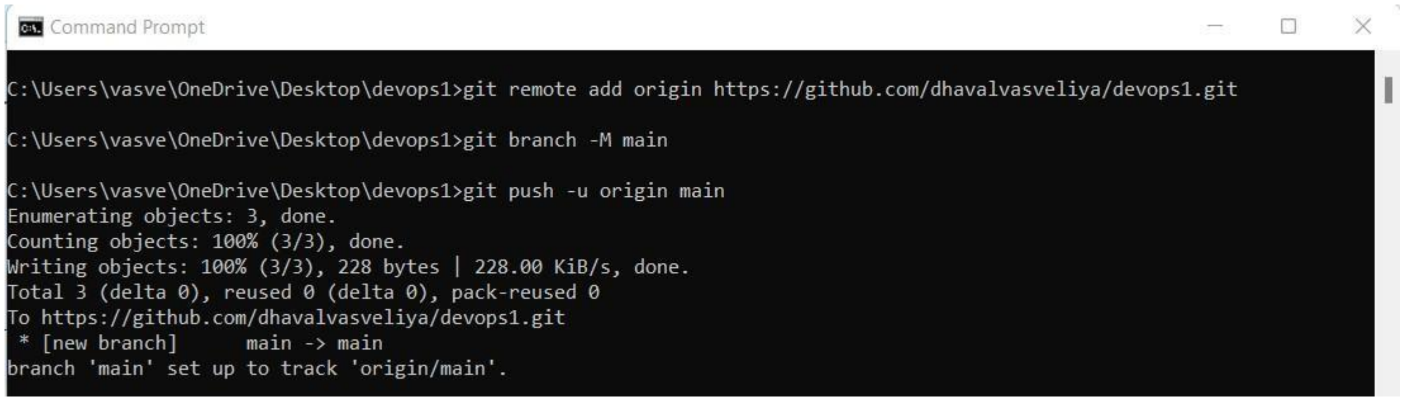
Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

When you're done filling out the information, press the **Create repository** button to make your new repo.

GitHub will ask if you want to create a new repo from scratch or if you want to add a repo you have created locally. In this case, since we've already created a new repo locally, we want to push that onto GitHub so follow the **'....or push an existing repository from the command line'** section:


```
C:\Users\vasve\OneDrive\Desktop\devops1>git remote add origin https://github.com/dhavalvasveliya/devops1.git

C:\Users\vasve\OneDrive\Desktop\devops1>git branch -M main

C:\Users\vasve\OneDrive\Desktop\devops1>git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 228 bytes | 228.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dhavalvasveliya/devops1.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

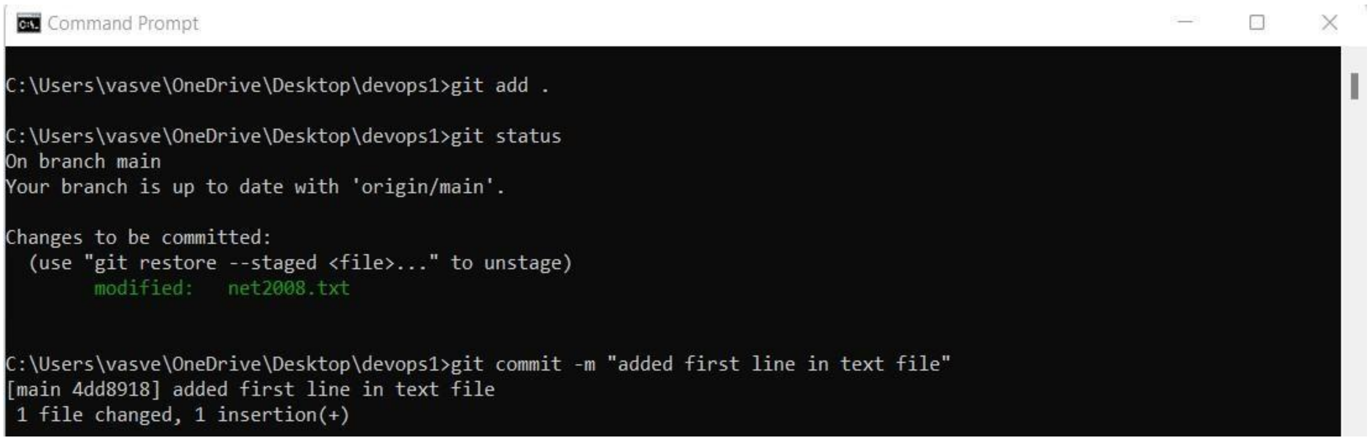
Change the URL in the first command line to what GitHub lists in this section since your GitHub username and repo name are different.

2.10 Step 8: Push a branch to GitHub

Before we can **push** a branch, we need to make some local changes.

For example: 1. Make some modifications to your file net2008.txt in the local repo. 2. **commit** these changes

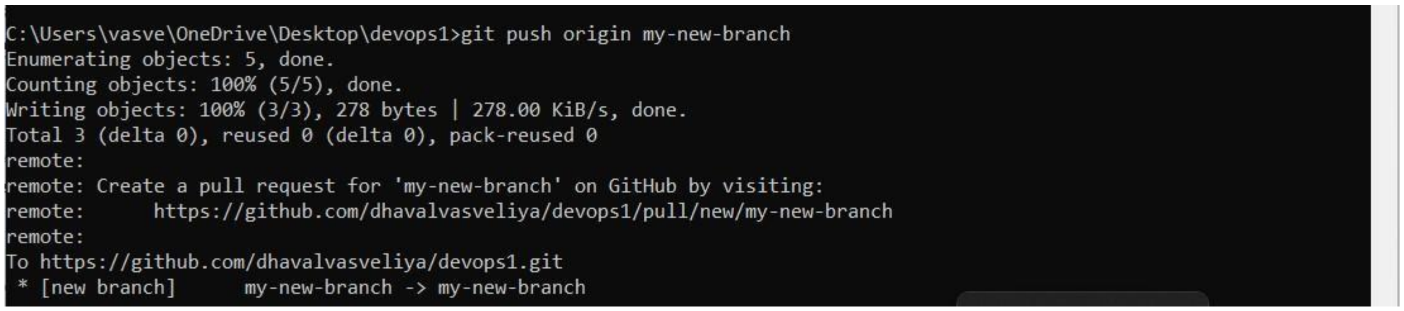
Now these changes are ready to be **pushed**



```
C:\Users\vasve\OneDrive\Desktop\devops1>git add .

C:\Users\vasve\OneDrive\Desktop\devops1>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   net2008.txt


C:\Users\vasve\OneDrive\Desktop\devops1>git commit -m "added first line in text file"
[main 4dd8918] added first line in text file
 1 file changed, 1 insertion(+)
```

So, let us **push** the commit in your branch to your new GitHub repo. This allows other people to see the changes you've made. If they're approved by the repository's owner, the changes can then be merged into the main branch.

To push changes onto a new branch on GitHub, you'll want to run `git push origin yourbranchname`. GitHub will automatically create the branch for you on the remote repository:

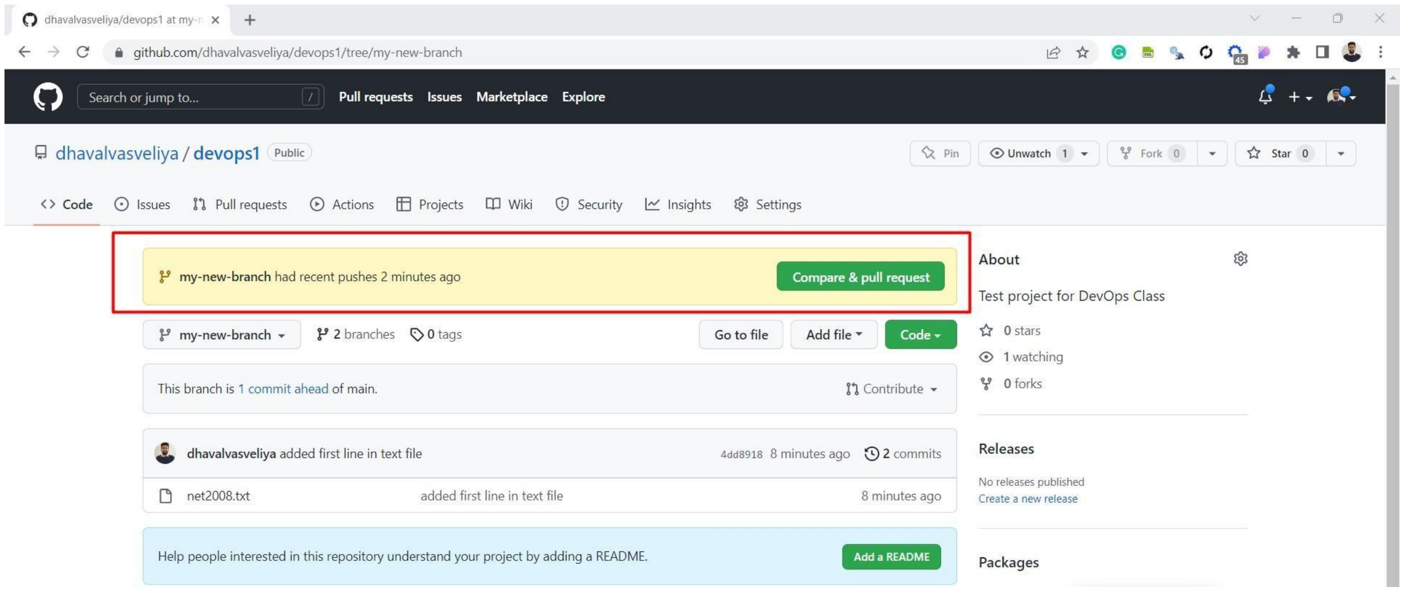


```
C:\Users\vasve\OneDrive\Desktop\devops1>git push origin my-new-branch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 278 bytes | 278.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'my-new-branch' on GitHub by visiting:
remote:      https://github.com/dhavalvasveliya/devops1/pull/new/my-new-branch
remote:
To https://github.com/dhavalvasveliya/devops1.git
 * [new branch]      my-new-branch -> my-new-branch
```

You might be wondering what that "origin" word means in the command above. What happens is that when you clone a remote repository to your local machine, git creates an **alias** for you. In nearly all cases this alias is called "**origin**." It's essentially shorthand for the remote repository's URL. So, to push your changes to the remote repository, you could've used either the command: **git push git@github.com:git/git.git yourbranchname** or **git push origin yourbranchname**

It will prompt you to log in with your GitHub username and password. Since August 2021, GitHub requires that you use the Personal Access Token instead of passwords.

If you refresh the GitHub page, you'll see note saying a branch with your name has just been pushed into the repository. You can also click the 'branches' link to see your branch listed there.


dhavalvasveliya/devops1 at my-n

github.com/dhavalvasveliya/devops1/tree/my-new-branch

dhavalvasveliya / devops1 Public

my-new-branch had recent pushes 2 minutes ago

my-new-branch 2 branches 0 tags

This branch is 1 commit ahead of main.

dhavalvasveliya added first line in text file 4dd8918 8 minutes ago 2 commits

net2008.txt added first line in text file 8 minutes ago

Help people interested in this repository understand your project by adding a README.

About

Test project for DevOps Class

0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

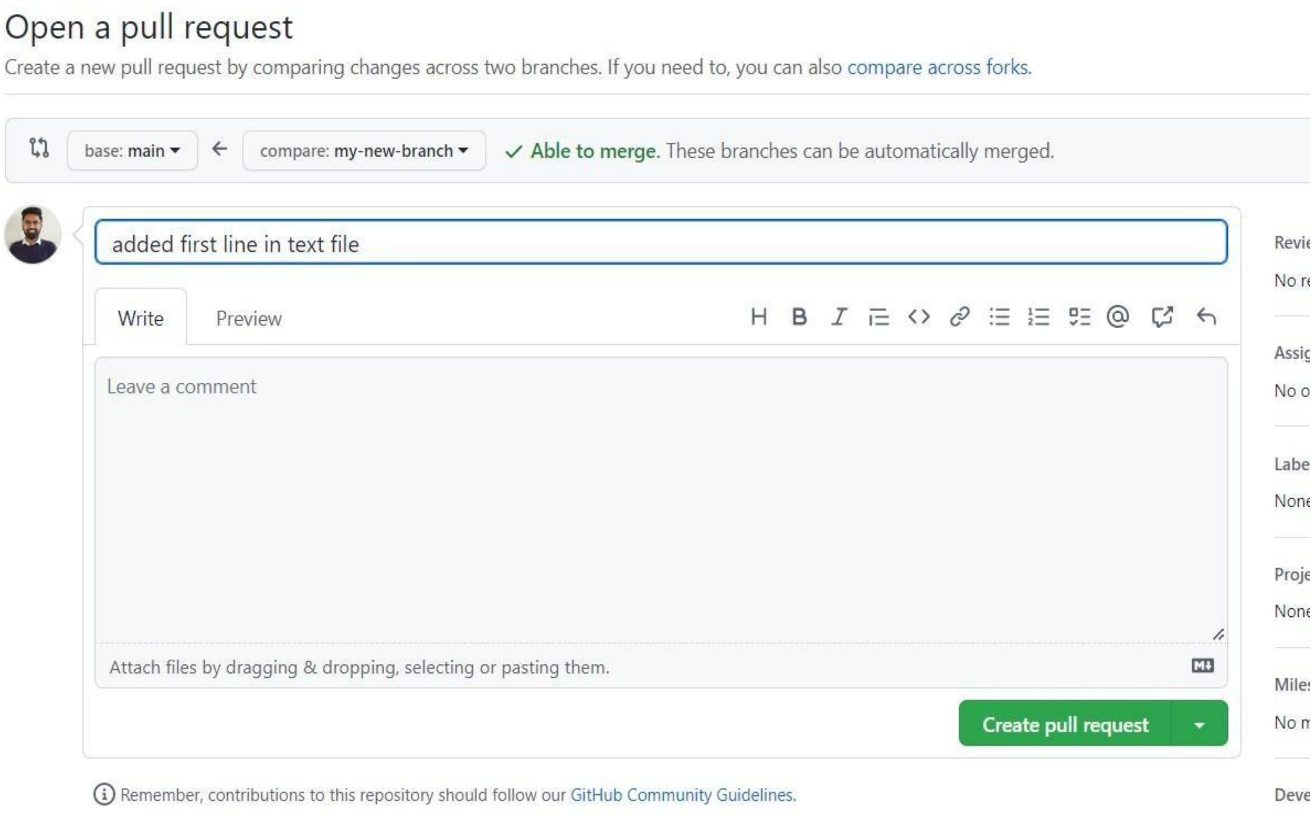
Packages

Now click the green button in the screenshot above. We're going to make a **pull request**!

2.11 Step 9: Create a Pull Request (PR)

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the main branch.

This is what the PR page looks like before you've submitted it:


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

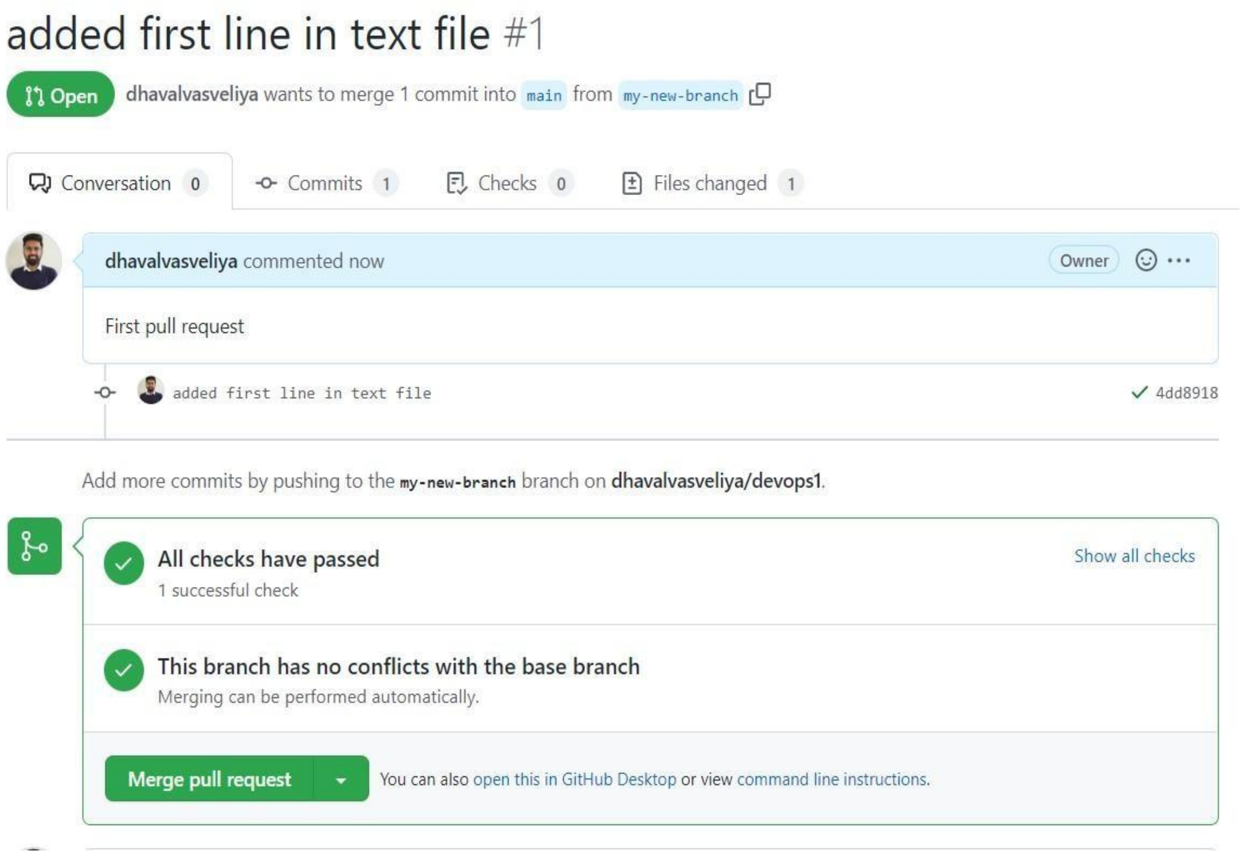
✓ Able to merge. These branches can be automatically merged.

added first line in text file

Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Create pull request

Remember, contributions to this repository should follow our GitHub Community Guidelines.

And this is what it looks like once you've submitted the PR request:


added first line in text file #1

Open dhavalvasveliya wants to merge 1 commit into main from my-new-branch

dhavalvasveliya commented now

First pull request

added first line in text file ✓ 4dd8918

Add more commits by pushing to the my-new-branch branch on dhavalvasveliya/devops1.

All checks have passed
1 successful check

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

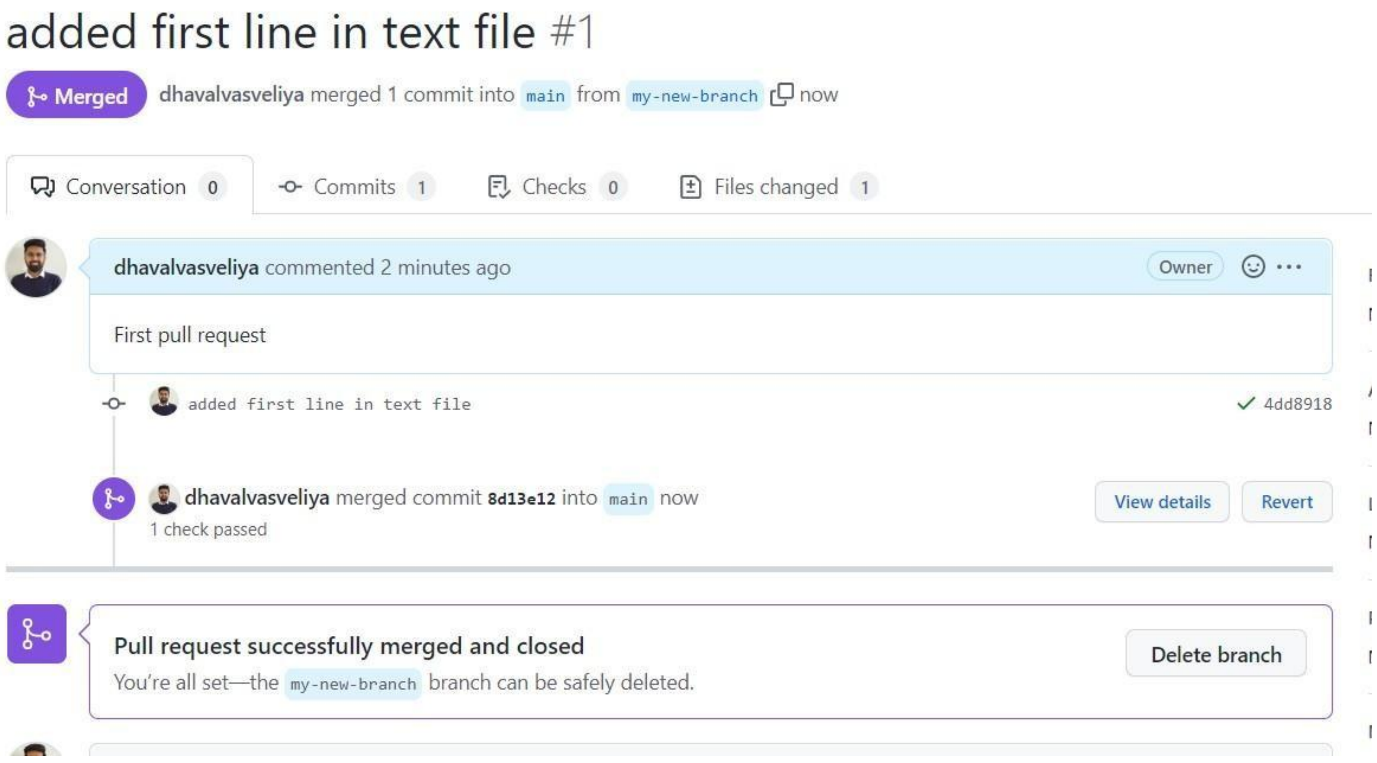
You might see a big green button at the bottom that says ‘Merge pull request’. Clicking this means you’ll merge your changes into the main branch.

Note that this button won’t always be green. In some cases it’ll be grey, which means you’re faced with a **merge conflict**. This is when there is a change in one file that conflicts with a change in another file and git can’t figure out which version to use. You’ll have to manually go in and tell git which version to use.

Sometimes you’ll be a co-owner or the sole owner of a repo, in which case you may not need to create a PR to merge your changes. However, it’s still a good idea to make one so you can keep a more complete history of your updates and to make sure you always create a new branch when making changes.

2.12 Step 10: Merge a PR

Go ahead and click the green ‘Merge pull request’ button. This will merge your changes into the main branch.


added first line in text file #1

Merged dhavalvasveliya merged 1 commit into `main` from `my-new-branch` now

dhavalvasveliya commented 2 minutes ago

First pull request

`added first line in text file` ✓ 4dd8918

dhavalvasveliya merged commit `8d13e12` into `main` now

1 check passed

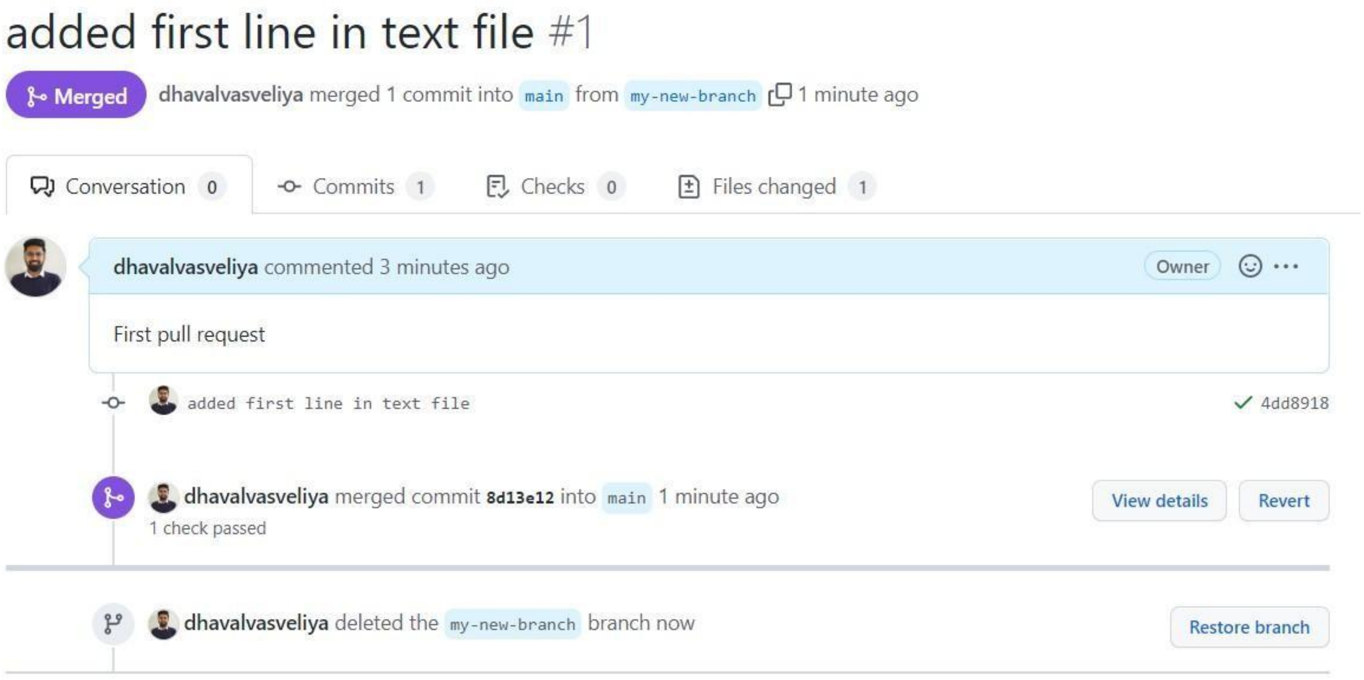
Pull request successfully merged and closed

You're all set—the `my-new-branch` branch can be safely deleted.

When you're done, I recommend deleting your branch (too many branches can become messy), so hit that grey 'Delete branch' button as well.

You can double check that your commits were merged by clicking on the 'Commits' link on the first page of your new repo.

This will show you a list of all the commits in that branch. You can see the one I just merged right up top (Merge pull request #1).


added first line in text file #1

Merged dhavalvasveliya merged 1 commit into `main` from `my-new-branch` 1 minute ago

dhavalvasveliya commented 3 minutes ago

First pull request

added first line in text file ✓ 4dd8918

dhavalvasveliya merged commit 8d13e12 into `main` 1 minute ago

1 check passed

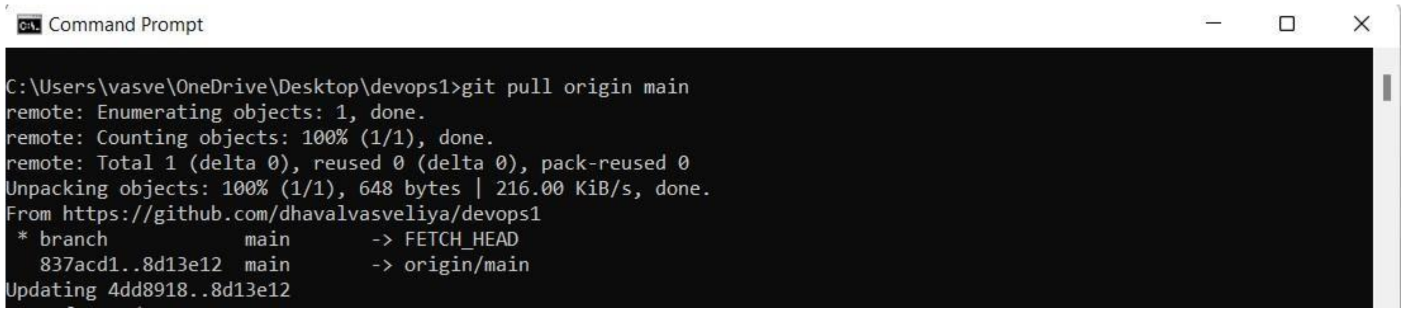
dhavalvasveliya deleted the `my-new-branch` branch now

You can also see the **hash code** of the commit on the right hand side. A hash code is a unique identifier for that specific commit. It's useful for referring to specific commits and when undoing changes `git revert` command to backtrack).

2.13 Step 11: Get changes on GitHub back to your computer

Right now, the repo on GitHub looks a little different than what you have on your local machine. For example, the commit you made in your branch and merged into the main branch doesn't exist in the main branch on your local machine.

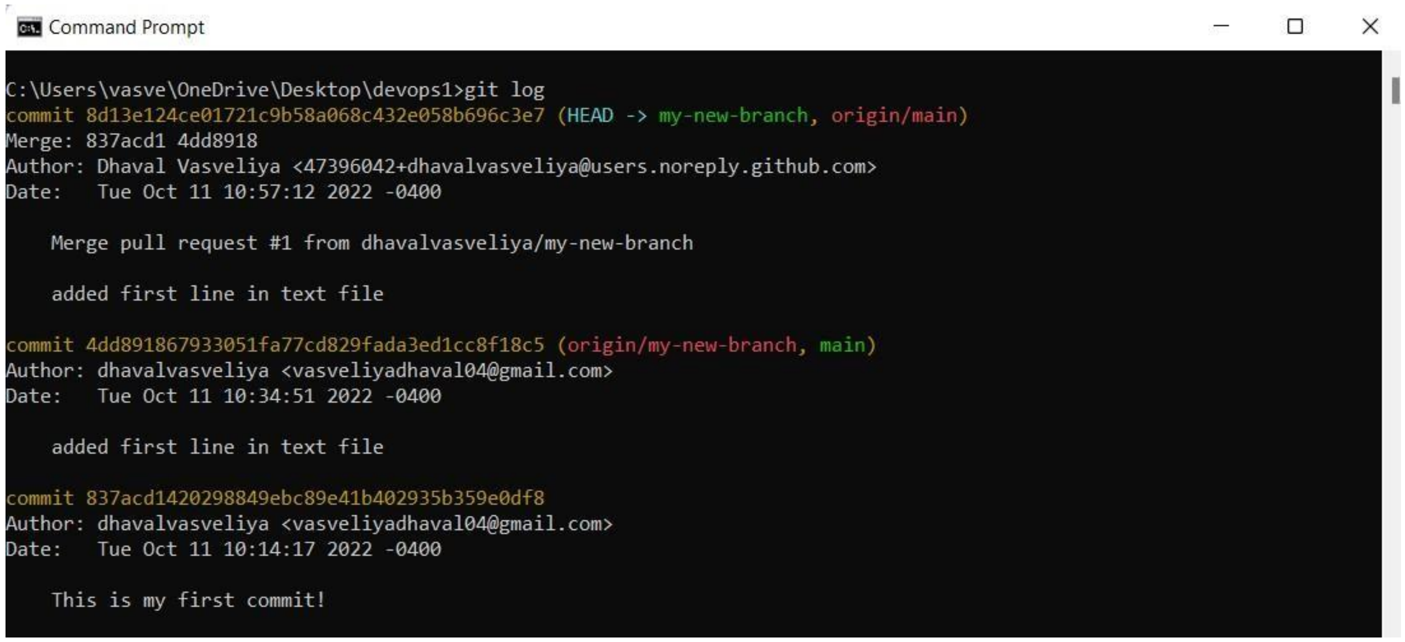
In order to get the most recent changes that you or others have merged on GitHub, use the `git pull origin main` command (when working on the main branch).



```
C:\Users\vasve\OneDrive\Desktop\devops1>git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 648 bytes | 216.00 KiB/s, done.
From https://github.com/dhavalvasveliya/devops1
 * branch            main       -> FETCH_HEAD
   837acd1..8d13e12  main       -> origin/main
Updating 4dd8918..8d13e12
```

This shows you all the files that have changed and how they've changed.

Now we can use the **git log** command again to see all new commits.



```
C:\Users\vasve\OneDrive\Desktop\devops1>git log
commit 8d13e124ce01721c9b58a068c432e058b696c3e7 (HEAD -> my-new-branch, origin/main)
Merge: 837acd1 4dd8918
Author: Dhaval Vasveliya <47396042+dhavalvasveliya@users.noreply.github.com>
Date:   Tue Oct 11 10:57:12 2022 -0400

    Merge pull request #1 from dhavalvasveliya/my-new-branch

    added first line in text file

commit 4dd891867933051fa77cd829fada3ed1cc8f18c5 (origin/my-new-branch, main)
Author: dhavalvasveliya <vasveliyadhaval04@gmail.com>
Date:   Tue Oct 11 10:34:51 2022 -0400

    added first line in text file

commit 837acd1420298849ebc89e41b402935b359e0df8
Author: dhavalvasveliya <vasveliyadhaval04@gmail.com>
Date:   Tue Oct 11 10:14:17 2022 -0400

    This is my first commit!
```

(You may need to switch branches back to the main branch. You can do that using the `git checkout main` command.)

2.14 Step 12 - Create a credentials file

We want to create a file **credentials.yml** that will contain your username and the Personal Access Token. A template is ready for you : **credentials_template.yml**

1. Copy **credentials_template.yml** to **credentials.yml**
2. Edit **credentials.yml** with your username/repo and PAT
3. Save it

We will need this file later. But this kind of file is for private use and we will never want it to be in the repo. Sometimes, a developer will make a mistake and push it to to the repo.

To prevent that from happening, we will use the gitignore feature of git.

2.15 Step 13 - Create a .gitignore and push

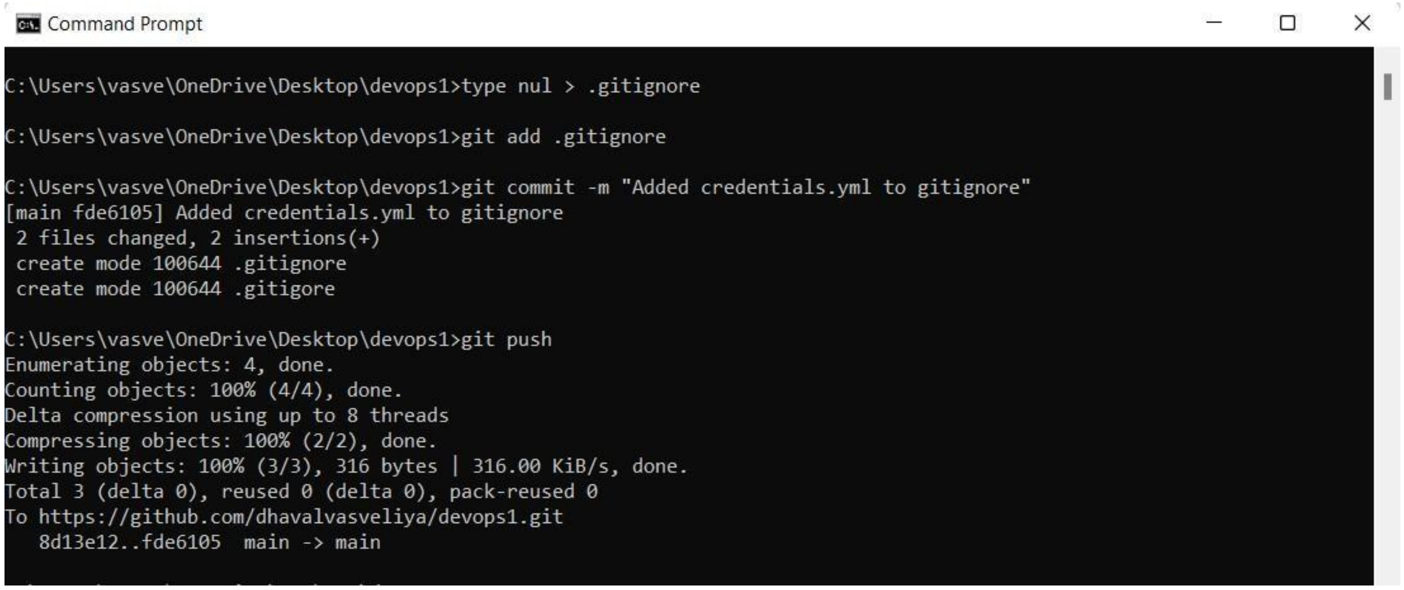
Git provides a couple ways to exclude files from being tracked as part of a repository. There are ways to exclude files on a per-repository basis or on a per-user basis.

The most common way of excluding (or ignoring) files is to use a **.gitignore** file stored in the repository itself. Like any other content in the repository, the **.gitignore** file must be staged into the index and committed to the repository any time changes are made.

The contents of the **.gitignore** file are simply a list of filenames or filename patterns, one on each line. To create your own list of files for Git to ignore, you'd simply create the file named **.gitignore** in the working directory, edit it to add the filenames or filename patterns you want ignored, and then add/commit it to the repository.

So we will create a **.gitignore** file and add crentials.yml to it and push it to git.

Edit .gitignore, using the text editor of your choice, to add credentials.yml on a single line in the file



```
C:\Users\vasve\OneDrive\Desktop\devops1>type nul > .gitignore

C:\Users\vasve\OneDrive\Desktop\devops1>git add .gitignore

C:\Users\vasve\OneDrive\Desktop\devops1>git commit -m "Added credentials.yml to gitignore"
[main fde6105] Added credentials.yml to gitignore
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .gitigore

C:\Users\vasve\OneDrive\Desktop\devops1>git push
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), 316 bytes | 316.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dhavalvasveliya/devops1.git
   8d13e12..fde6105  main -> main
```

Now, Git will ignore the credentials.yml file. For example, here you can see the file exists in the working directory, but git status reports no changes or untracked files.

2.16 Step 14: PyGithub

PyGithub is a Python library to use the Github API v3. With it, you can manage your Github resources (repositories, user profiles, organizations, etc.) from Python scripts.

2.16.1 Create a Github Access Token

Follow this documentation to create your own Github access token.

Update your credentials.yml with the generated token.

2.16.2 Install PyGithub Library

This package is in the Python Package Index, so the following should be enough: `pip3 install PyGithub`

2.16.3 Install YAML Library

We want a yaml parser to read our credentials `pip3 install pyyaml`

2.16.4 Using Python to operate on Github

There are useful examples for your reference to complete the tasks below.

- Get all branches you have created for your public repo

- Get all pull requests you have created • Get a list of commits you have created in your `main` branch.

```
[ ]: from github import Github  import
     yaml

     # credentials.yml contains your usr/repo and PAT created in step 11 above
     #  So  we  load  the  data  into  a  YML  object  data    =
     yaml.safe_load(open('username-credentials.yml'))

     # Extract the user and token from the data object # 0.
     Complete these 2 lines below # user = ...
     # token = ...

     # using an access token g =
     Github(token)    repo    =
     g.get_repo(user)

     ## Complete your tasks from here
```

```
# 1. Get all branches you have created for your public repo

# 2. Get all pull requests you have created

# 3. Get a list of commits you have created in your `main` branch.
```

2.17 Step 15 - Check your work

1. A public repo needs to be created under your Github account.
2. At least 2 branches need to be created
3. At least 1 pull request needs to be created and merged
4. At least 3 commits (including 1 merge commit) need to be in `main` branch.
5. Code and all outputs from Step 10.