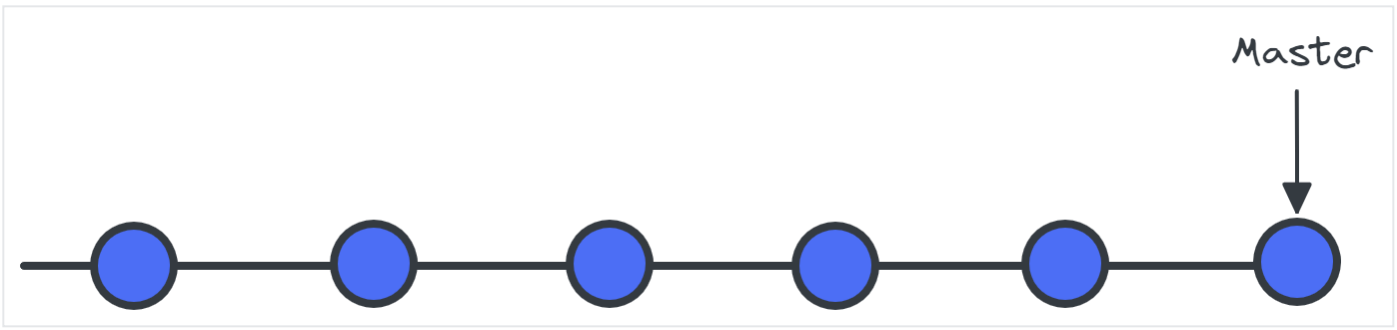
What is Git?

Git is a distributed, open-source version control system. It enables developers and data scientists to track code, merge changes and revert to older versions **- AWS**. It allows you to sync changes with a remote server. Due to its flexibility and popularity, Git has become an industry standard as it supports almost all development environments, command-line tools, and operating systems.

How does Git work?

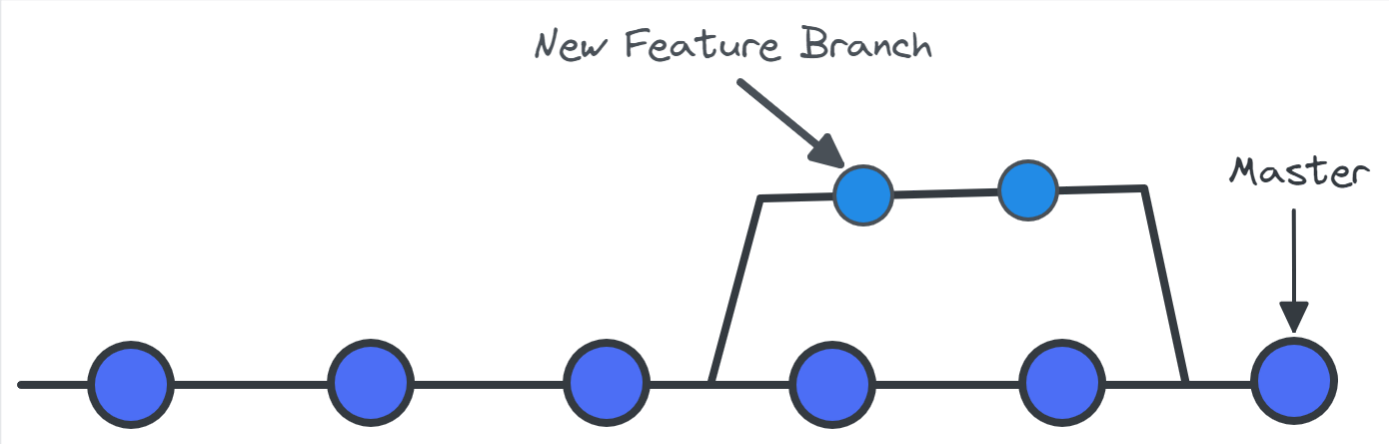
Git stores your files and their development history in a local repository. Whenever you save changes you have made, Git creates a commit. A commit is a snapshot of current files. These commits are linked with each other, forming a development history graph, as shown below. It allows us to revert back to the previous commit, compare changes, and view the progress of the development project **- Azure DevOps**. The commits are identified by a unique hash which is used to compare and revert the changes made.



A graph of the development history

Branches

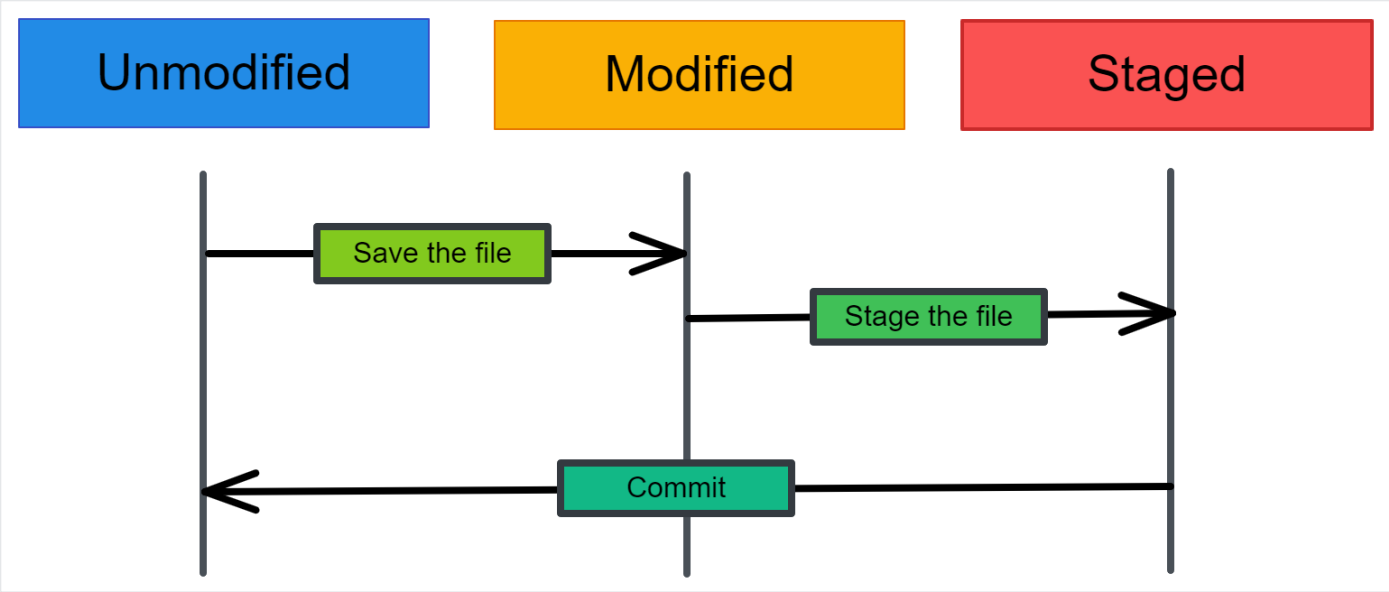
The branches are copies of the source code that works parallel to the main version. To save the changes made, merge the branch into the main version. This feature promotes conflict-free teamwork. Each developer has his/her task, and by using branches, they can work on the new feature without the interference of other teammates. Once the task is finished, you can merge new features with the main version (master branch).



Adding new feature to repository

Commits

There are three states of files in Git: modified, staged, and commit. When you make changes in a file, the changes are saved in the local directory. They are not part of the Git development history. To create a commit, you need to first stage changed files. You can add or remove changes in the staging area and then package these changes as a commit with a message describing the changes.



Three states of files in Git

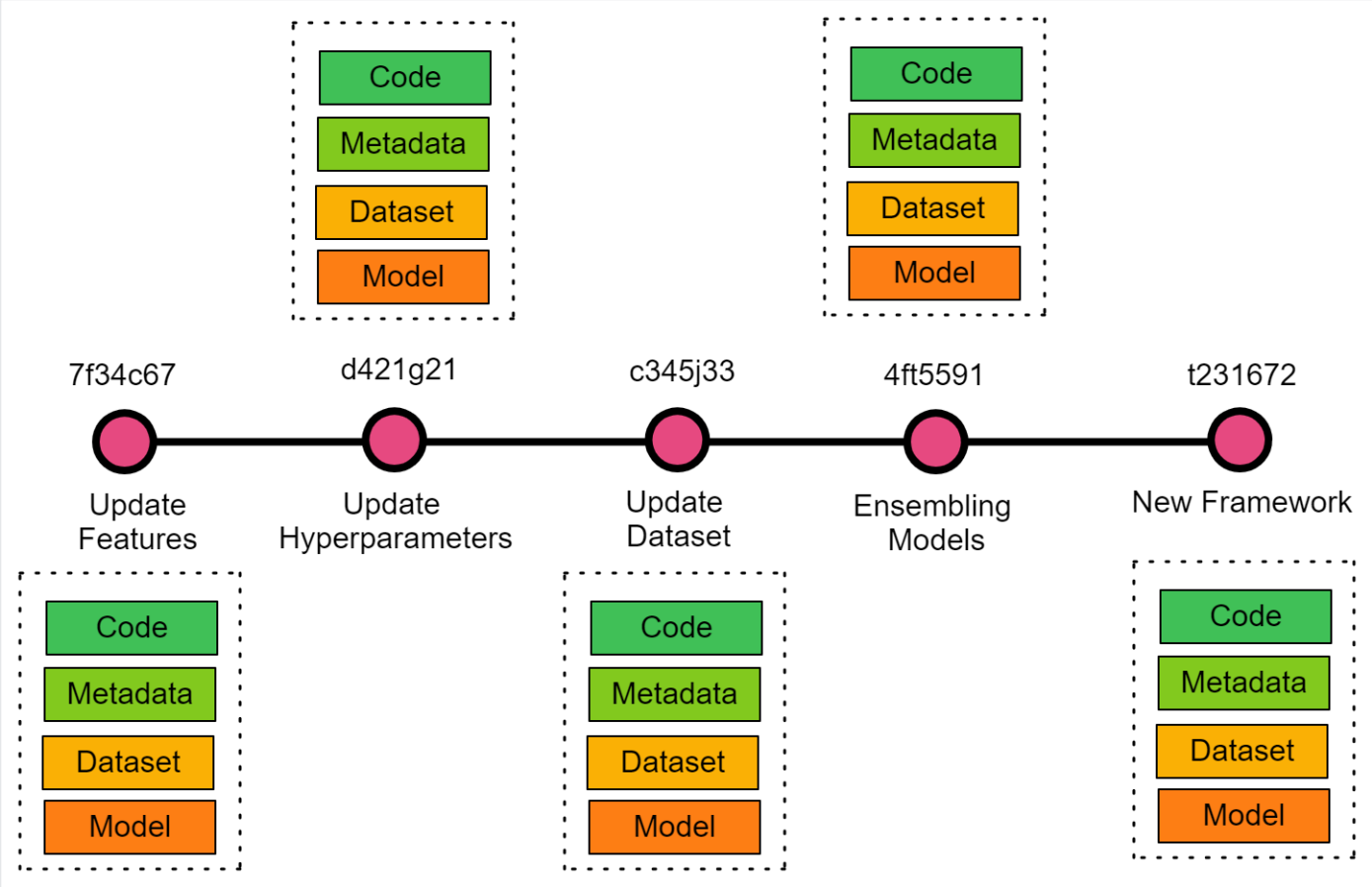
What are the benefits of Git?

- Track changes: It allows developers to view historical changes. Development history makes it easy to identify and fix bugs.
- IDE Integration: Due to its popularity, Git integration is available in all development environments, for example VSCode and JupyterLab.
- Team collaboration: A developer team can view their progress, and by using branches, they can work individually on a task and merge changes with the main version. Pull requests, resolving merge conflicts, and code review promote team collaboration.
- Distributed VSC: In a distributed system, there is no centralized file storage. There are multiple backups for the same project. This approach allows developers to work offline and commit changes.

Git for Data Science Projects

Git provides version control for scripts, metrics, data, and models. By using Git extension **git-lfs**, you can store and version a large database and machine learning models. In a typical data science project, you have a Jupyter notebook, dataset, model, metadata, and model metrics. The metadata includes files containing meta-information about the machine learning model, features, model parameters, and automation files. All of these are necessary for monitoring the progress of AI applications and resolving issues.

Track data science experiments help scientists revert accidental changes, select the best experiment based on the performance metric, and collaborate with other teammates. The diagram below shows how changes to data or code affect the metadata and output of the model. Tracking these changes can also help other teammates come up with a better solution. Learn **all about Git** in the latest blog from Summer Worsley.



Git for a Data Science Project

Collaboration with GitHub

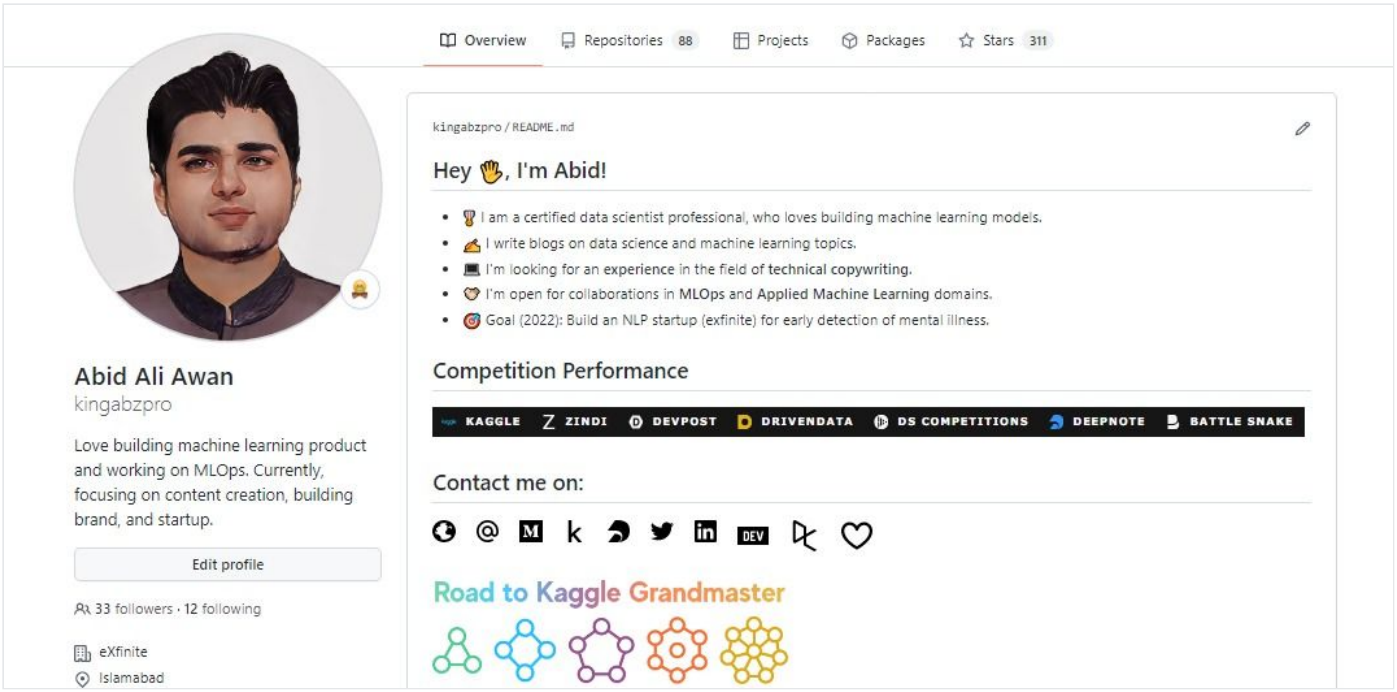
GitHub is a cloud software development platform. It is commonly used for saving files, tracking changes, and collaborating on development projects. In recent years, GitHub has become the most popular social platform for software development communities. Individuals can contribute to open-source projects and bug reports, discuss new projects and discover new tools.

Data scientists and machine learning engineers are following the path of software developers and integrating the workflow with GitHub. By doing this, they can share their research work, allow community contribution, and collaborate with data teams. You can find all kinds of data science and machine learning projects, guides, tutorials, and resources on

this platform. For students, the platform has become an opportunity to gain work experience and eventually land a job in a prestigious company.

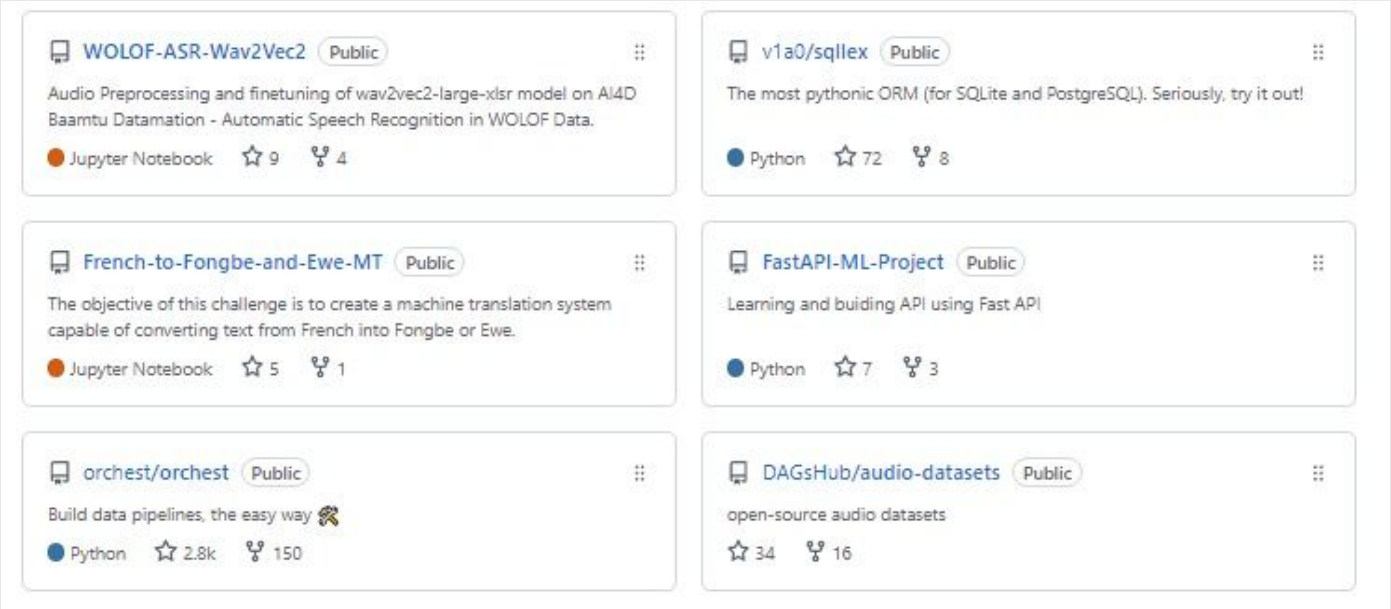
Portfolio

Most technical recruiters will ask for the portfolio projects or GitHub profile. This helps them determine whether a candidate is a good fit for their company. It is highly recommended to create a GitHub profile and update it regularly. Hiring managers are always on the lookout for candidates that are highly experienced in software development and contribute to open-source projects. Being able to analyze the GitHub portfolio helps them prepare questions for technical interview sessions.



GitHub Profile

GitHub enables data scientists to showcase their projects, and it can also count as work experiences on your resume. Showcasing portfolio projects also creates opportunities to work together, launch a startup, and research work.



Portfolio Projects

Features

GitHub also provides various other features that are as important as showcasing a portfolio. It is necessary to learn about each feature so that you can incorporate them into your data science projects.

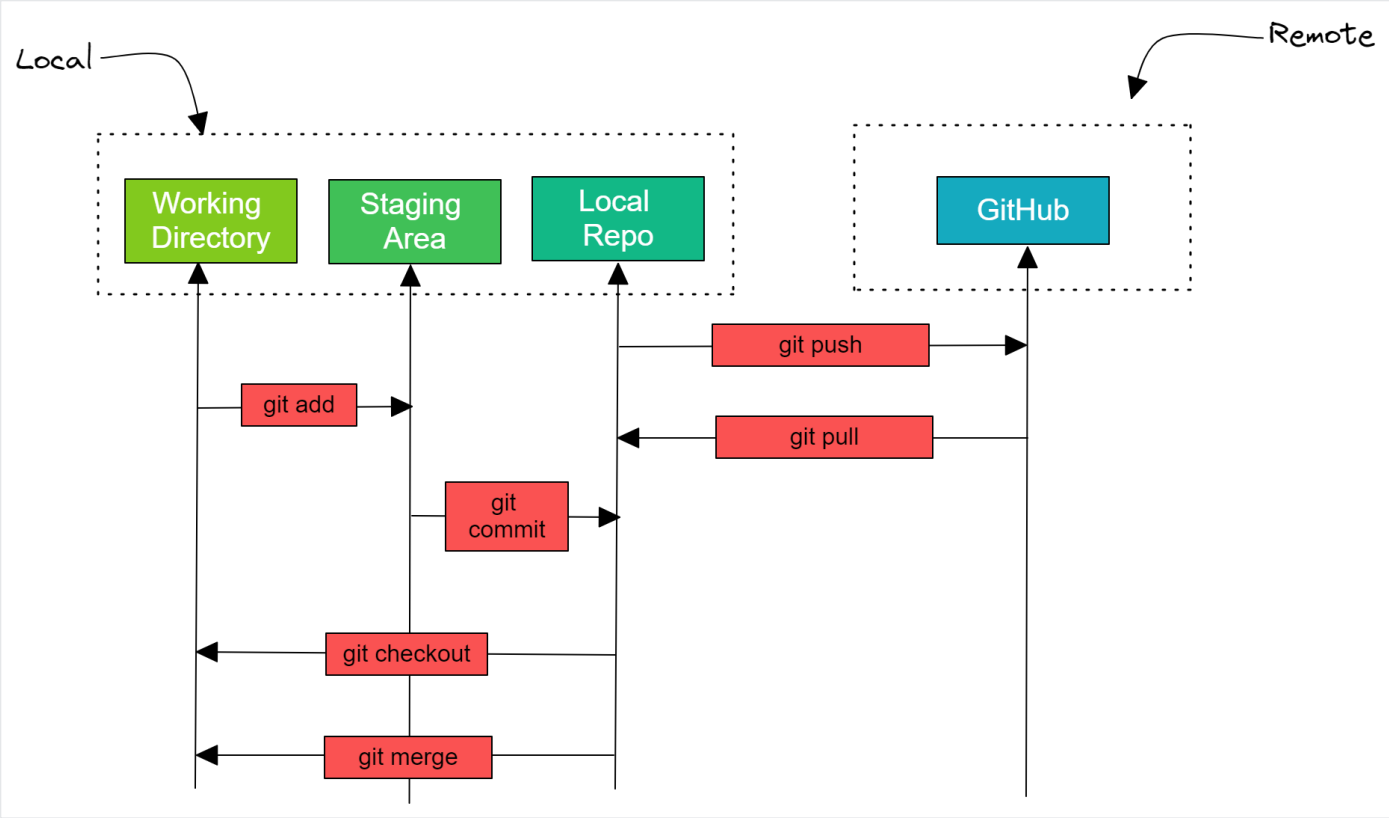
- Open-source: GitHub provides a complete ecosystem for open-source projects. You can sponsor maintainers, contribute to a project, use the open-source tool in your existing project, and promote your work.
- Community Collaboration: GitHub has become a community platform where issues, feature requests, code, and documentation contributions can be discussed.
- Explore: GitHub Explore tab helps you discover new projects, trending tools, and developer events.
- GitHub Gists: You can share the snippet of your code or embed it in a blog or website.
- GitHub CLI: It allows you to perform merge requests, review code, check issues, and monitor progress from the command line program.
- Free Storage: unlimited private and public repositories storage.
- Web hosting: You can publish your portfolio site or documentation. GitHub pages provide easy to build and deploy website experience.
- Codespace: a cloud development environment integrated with your GitHub repository.

- Project: a customizable, flexible tool for planning and tracking the work on GitHub.
- Automation: GitHub Action automates development workflow such as build, test, publish, release, and deployment.
- Sponsor: You can support your favorite open-source project or developers by paying a monthly or one-time fee. It also allows developers to use third-party payment platforms such as ko-fi.

Basic Commands

Before we jump into managing data science projects, let's learn about the most common Git commands that you will be using in every data science project. The basic commands include initializing the Git repository, saving changes, checking logs, pushing the changes to the remote server, and merging.

- **git init** create a Git repository in a local directory.
- **git clone** <remote-repo-address>: copy the entire repository from a remote server to remote directory. You can also use it to copy local repositories.
- **git add** <file.txt>: add a single file or multiple files and folders to the staging area.
- **git commit** –m “Message”: create a snapshot of changes and save it in the repository.
- **git config** use to set user-specific configurations like email, username, and file format.
- **git status** shows the list of changed files or files that have yet to be staged and committed.
- **git push** <remote-name> <branch-name>: send local commits to remote branch of repository.
- **git checkout** -b <branch-name>: creates a new branch and switches to a new branch.
- **git remote** –v: view all remote repositories.
- **git remote add** <remote-name> <host-or-remoteURL>: add remote server to local repository.
- **git branch** –d <branch-name>: delete the branch.
- **git pull** merge commits to a local directory from a remote repository.
- **git merge** <branch-name>: after resolving merge conflicts the command blends selected branch into the current branch.
- **git log** show a detailed list of commits for the current branch.



Complete Development with GitHub

If you are interested in learning more commands, check out the **Git cheat sheet** by Gitlab.

Getting started

In this section, we are going to use Git to track a data science project and GitHub as a remote server. We will learn how to install Git, create and clone a repository from GitHub, run machine learning experiments, and push changes (notebook, model, data) to GitHub using Windows PowerShell 7.

Installing Git

Git supports all operating systems. You can install it using command-line tools or directly download and install the setup.

Linux

For Debian/Ubuntu-based operating systems use `apt-get install git`, and if you are using another Linux-based system, check out the complete list of installing commands **here**.

macOS

If you have **homebrew** installed, use this command to download and install Git: `brew install git`. You can also download the binary **installer** and run the setup.

Windows

Installing Git on Windows is hassle-free. Just go to the download **page**, click on the specific Windows version, and download and install the setup. If you have a **winget tool**, you can install it by typing `winget install --id Git.Git -e --source winget` in PowerShell.

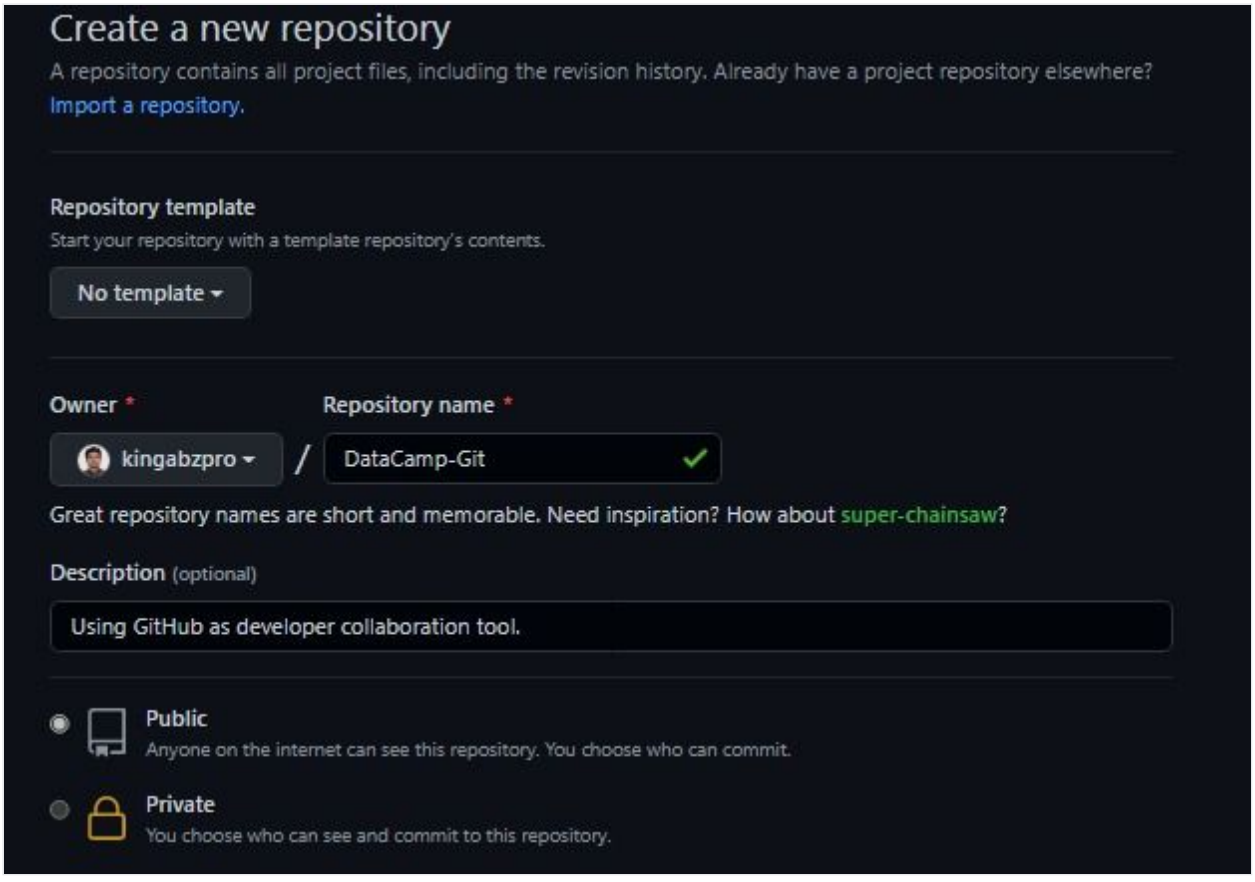
After installing Git, make sure you have configured the user name and email. This information is used to sign the commits.

```
git config --global user.name "your-user-name"
git config --global user.email "your@email.com"
```

For more in-depth information on how to install Git, click **here**.

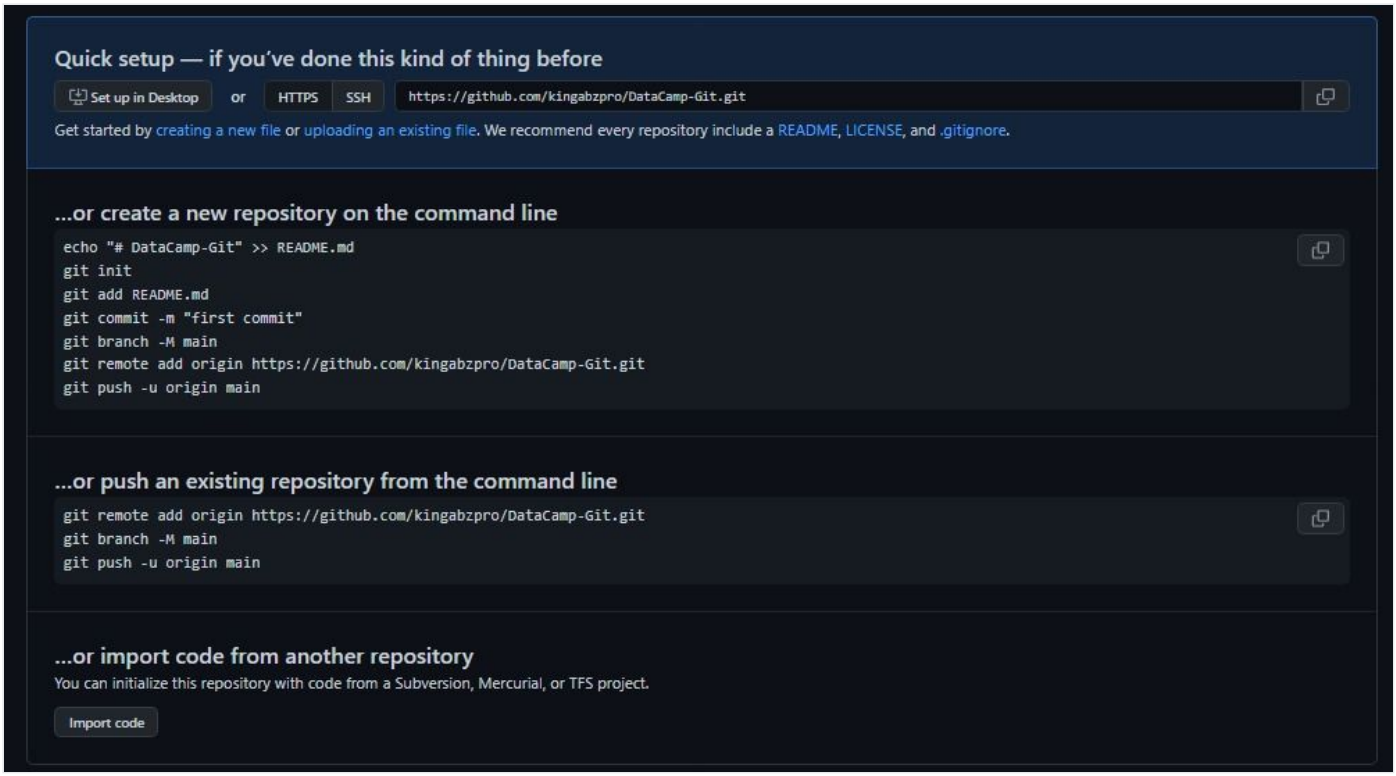
Initializing the Project

If you have a GitHub account, click on the + button and select a new repository. After that, type the repository name and add a simple description. It will create an empty public repository.



Creating project

There are many ways to clone remote repositories to the local directory, and GitHub provides a detailed guide on how to clone, add remote, and initialize a Git project.



GitHub Clone

We can simply clone the repository by providing an HTTPS link. Make sure you are in the working directory using the command prompt or PowerShell.

```
git clone https://github.com/kingabzpro/DataCamp-Git.git

>>> Cloning into 'DataCamp-Git'...
>>> warning: You appear to have cloned an empty repository.

cd .\DataCamp-Git\
```

OR

Create a new directory called “DataCamp-Git” and initialize Git using a simple command. After that, add a connection to the remote repository so that you can sync your work with GitHub.

```
mkdir DataCamp-Git
cd .\DataCamp-Git
git init

>>> Initialized empty Git repository in C:/Repository/GitHub/DataCamp-Git/.git/

git remote add origin https://github.com/kingabzpro/DataCamp-Git.git
```

OR

If you already have a project in a directory, just initialize Git using `git init` and add GitHub remote, as shown above.

Simple Commit

Before we add files to our repository, make sure you are in the correct local directory.

We will start simple and create a README file with the heading DataCamp-Git. Then, we will add it to the staging area by using `git add`.

```
echo "# DataCamp-Git" >> README.md
git add README.md
```

Git status shows that we are on the main branch and the `README.md` file is staged and ready to be committed.

```
git status

>>> On branch main
>>> No commits yet
>>> Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
```

To create our first commit, we will use `git commit` with a message. As we can observe, the first commit is added under the ed9c886 hash.

```
git commit -m "first commit"

>>> [main (root-commit) ed9c886] first commit
```

```
>>> 1 file changed, 1 insertion(+)
>>> create mode 100644 README.md
```

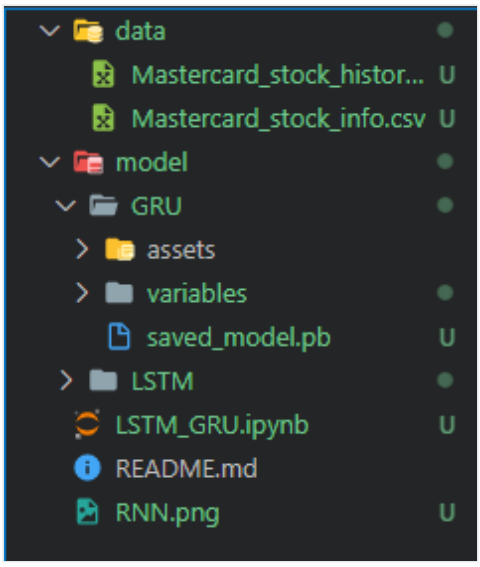
Adding Project Files

We will use the DataCamp workspace **MasterCard Stock Price with LSTM and GRU** and download files. The author of the project has preprocessed the data and training time series data on the LSTM and GRU models. Learn more about the project by reading **Recurrent Neural Network Tutorial (RNN)**.

To save the model file, we have added a new code cell in the project's Jupyter notebook. The new script will create a new directory called “model” and save both LSTM and GRU models.

```
!mkdir -p model
model_lstm.save('model/LSTM')
model_gru.save('model/GRU')
```

As we can see, the Git repository has a data folder containing CSV files, the model folder with the model’s weight and metadata.



We will now stage all the files. You can add any directory, file, or data after the initial command.

```
git add .\data .\model LSTM_GRU.ipynb RNN.png
```

OR

If you want to add all files to the staging area, then use dot.

```
git add .
```

Commit and Push

We will commit all the changes with a simple message, and the output shows all the new files in create mode.

```
git commit -m "project files added"

>>> [main aa3e19a] project files added
>>>  10 files changed, 5020 insertions(+)
>>>  create mode 100644 LSTM_GRU.ipynb
>>>  create mode 100644 RNN.png
>>>  create mode 100644 data/Mastercard_stock_history.csv
>>>  create mode 100644 data/Mastercard_stock_info.csv
>>>  create mode 100644 model/GRU/saved_model.pb
>>>  create mode 100644 model/GRU/variables/variables.data-00000-of-00001
>>>  create mode 100644 model/GRU/variables/variables.index
>>>  create mode 100644 model/LSTM/saved_model.pb
>>>  create mode 100644 model/LSTM/variables/variables.data-00000-of-00001
create mode 100644 model/LSTM/variables/variables.index
```

Syncing with GitHub remote repository requires a remote name and branch name `git push <remote-name> <branch-name>`. If you have only one remote and one branch, then using `git push` will work.

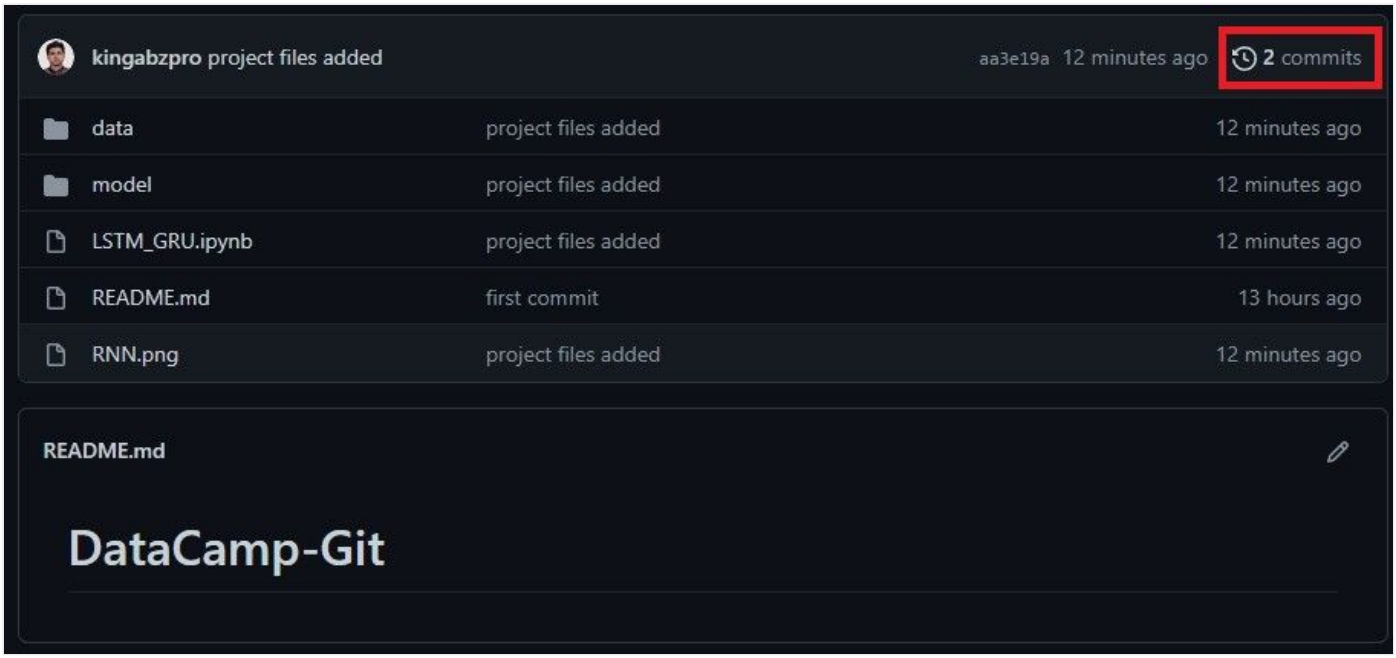
After `git push`, the pop-up window will ask for the credentials, just add your GitHub username or password. You can also generate **Personal access tokens** and add them instead of the password. Learn more by reading the **Git Push and Pull Tutorial**.

```
git push

>>> Enumerating objects: 21, done.
>>> Counting objects: 100% (21/21), done.
>>> Delta compression using up to 4 threads
>>> Compressing objects: 100% (19/19), done.
>>> Writing objects: 100% (21/21), 1.83 MiB | 1.59 MiB/s, done.
>>> Total 21 (delta 2), reused 0 (delta 0), pack-reused 0
>>> remote: Resolving deltas: 100% (2/2), done.
```

```
>>> To https://github.com/kingabzpro/DataCamp-Git.git
>>>  * [new branch]      main -> main
```

We are going to check our GitHub repository **kingabzpro/DataCamp-Git** to see whether we have successfully pushed the changes to remote. The GitHub repository has all the files, data, and models.



Remote push to GitHub

Git Branches

It is recommended to work with branches: for example, if you want to work on project documentation, create a documentation branch using `git checkout` or `git branch`. Make changes in the README file and when you have finalized the changes, merge the branch with the base.

In our case, we have created and switched to a new branch called `readme`.

```
git checkout -b readme
```

Let's edit the README file by adding a description to the project and link the RNN DataCamp workspace and tutorial. After that, we will stage changes and save a snapshot of changes with a message.

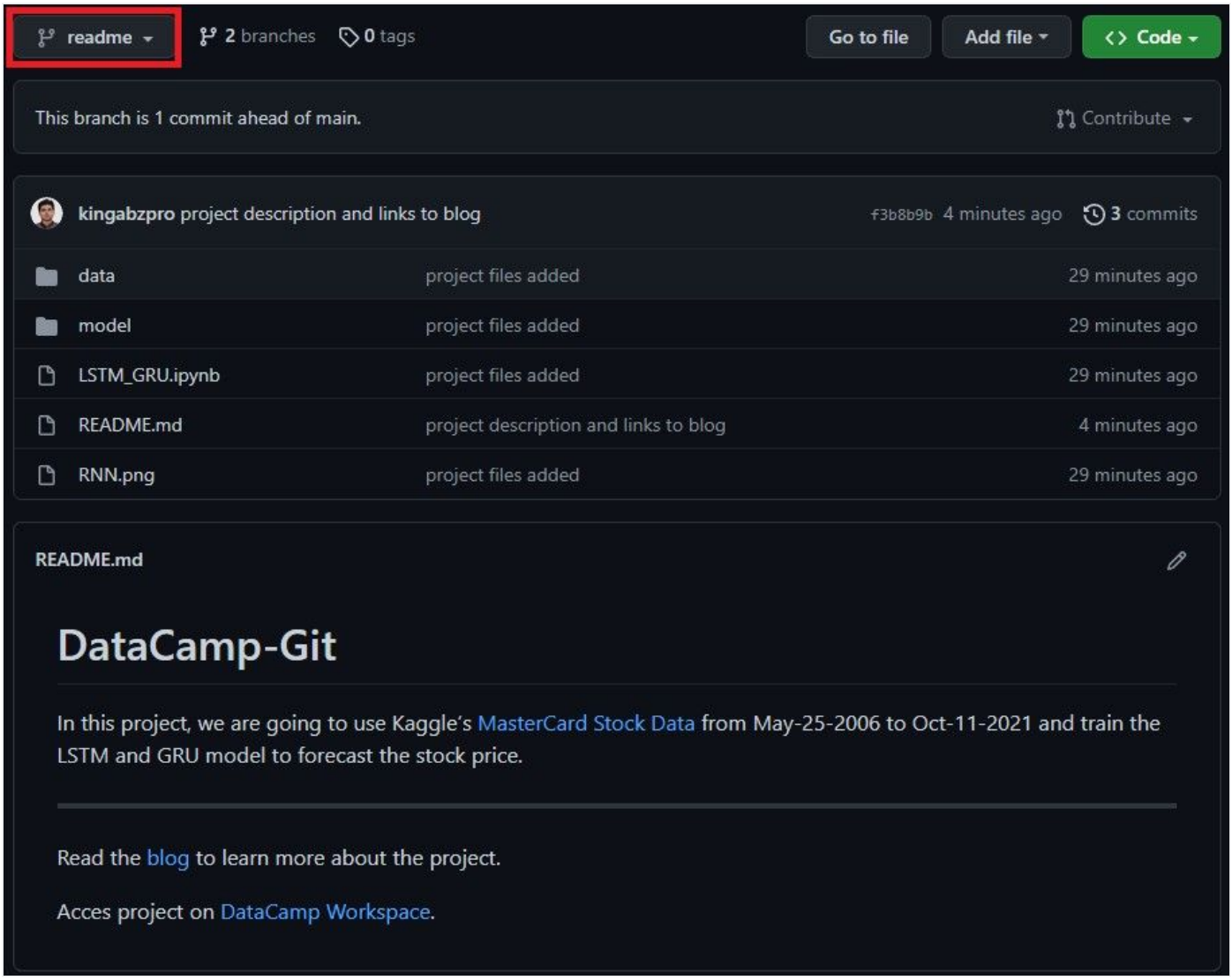
```
git add README.md
git commit -m "project description and links to blog"

>>> [readme f3b8b9b] project description and links to blog
>>>  1 file changed, 8 insertions(+)
```

The remote repository doesn't have a readme branch. To create a new branch and push changes, we will use “readme:readme”. The output of the command shows that new branches have been created and both local and remote `readme` branches are synced.

```
git push origin readme:readme
>>> remote: Resolving deltas: 100% (1/1), completed with 1 local object.
>>> remote: Create a pull request for 'readme' on GitHub by visiting:
>>> remote:  https://github.com/kingabzpro/DataCamp-Git/pull/new/readme
>>> remote:
To https://github.com/kingabzpro/DataCamp-Git.git
>>>  * [new branch]      readme -> readme
```

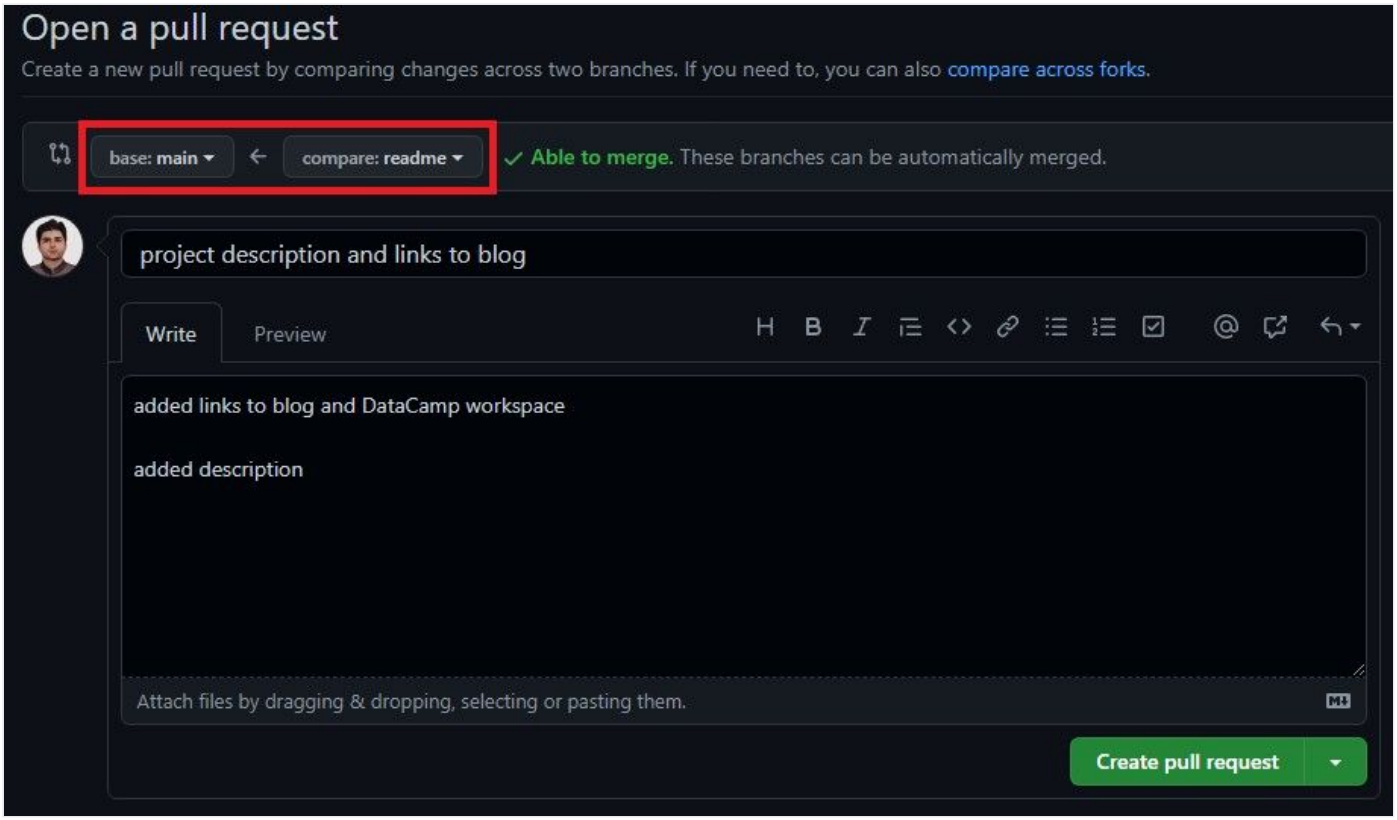
You can observe that we have successfully pushed the local branch to GitHub with a modified version of README.md file.



Readme branch on GitHub

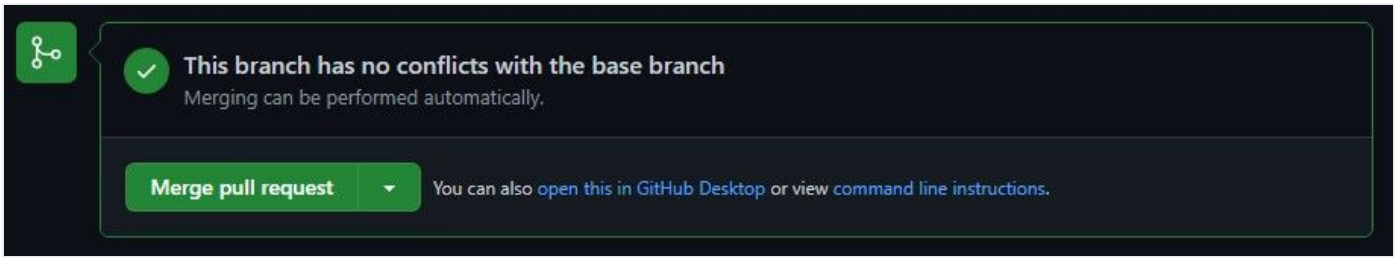
Pull Request

This functionality is common for organizations. For example, a software developer has worked on a new feature and wants to merge changes to the main remote branch. We will now create pull requests using GitHub GUI by clicking on the pull request button. After that, select the readme branch which we want to merge with the base (main). You can type a detailed explanation of what features were added and click on the pull request button.



Pull request from readme to main branch

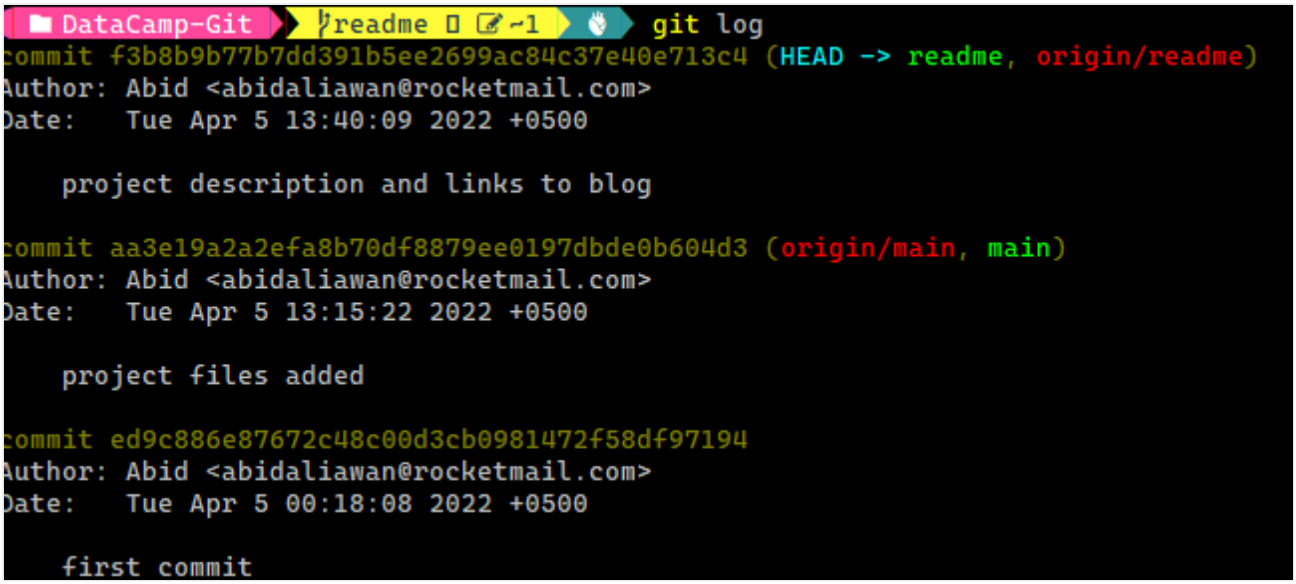
The maintainer of the repository will compare your changes and merge them when they have passed all the tests. In our case, you are the maintainer, so click on the merge request to blend changes with the main branch.



Merge pull request on GitHub

Congratulations, we have successfully created a pull request and merged it with the main branch. You can view the changes on the main branch **here**.

If you want to view all the changes within your git repository, just type `git log`, and it will show historical changes to your project. Logging changes in data science projects are important, and Git helps us track all the changes, even large datasets.



History of Git Logs

Conclusion

GitOps are crucial for data application development. They have become an essential skill for all types of IT jobs; even academic researchers are using them to share experimental code with a wider audience. On the other hand, GitHub plays a larger role in promoting open-source projects by providing a free software development ecosystem for all.

In this tutorial, we have learned about Git and GitHub and why they are important for data science projects. The tutorial also introduces you to basic Git commands and provides hands-on experience on how to track changes in data, model, and code. If you are interested in learning more about Git, then take an **Introduction to Git** course on DataCamp.

Git FAQs

What is Git?

An open-source distributed version control system. It allows developers to store, version, and visualize changes in a development project. It promotes flexible teamwork and