Introduction

Git is a version control system. A version control system is a software that helps you manage the different versions of your work.

There is a common misconception that git is widely used in software development, but data scientists and machine learning engineers also use git extensively.

This module comprised of three sessions.

In the first session, you learnt:

- Need for version control systems
- Different types of version control systems
- Difference between git and github
- Creation of a project for fraud detection with version control
- Reverting back to stable versions
- Branching

In the second session, you learnt the concept of open source contributions and how to work with them.

In the third session, you learnt how to create your portfolio website using github pages and jekyll.

Why Version Control?

You must have definitely saved a file as final1, final2, final_final or really_final. This seems like a quick fix while you are working, but when you revisit the same directory after a few months, it becomes really difficult to locate the file and the folder.

If you are working in collaboration, it becomes difficult to track who made the change, what change was made and when it was made.

Other than that, it is difficult to retrieve the files in your system in case of any crash. However, if you use a version control system, such as git, you can fix these problems.

Version control system, such as git and github, helps you by taking care of the following:

1. Tracking changes
2. Ability to revert back to any version
3. Finding what changes were made, when were they made and by whom
4. Easy recovery in case of any disaster
5. Improving the efficiency and agility of team projects

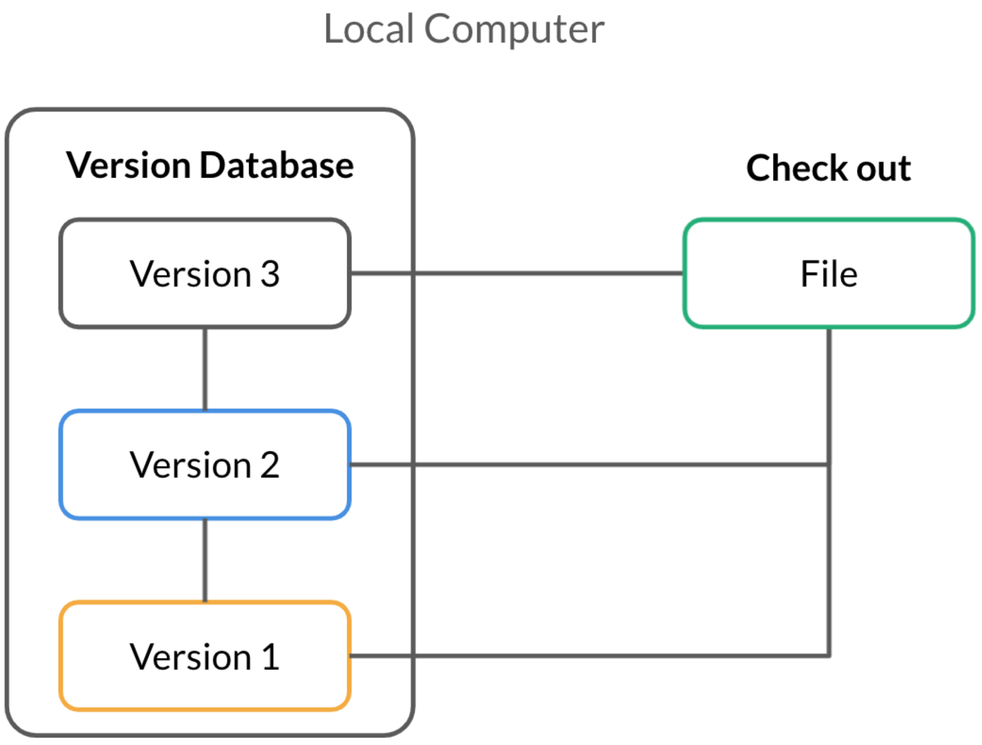
Git is a version control system for code files. However, you encounter version control systems in your daily life as well. If you have used Google docs or sheets, you know that it has the function of version history. However, git is a sophisticated tool that tracks a great deal of details and is used mainly for version control of code files.

Background of Version Control System

Historically, the version control system was not as great as it is today
There are three main types of version control systems:

1. Local Version Control System
2. Centralised Version Control System
3. Distributed Version Control System

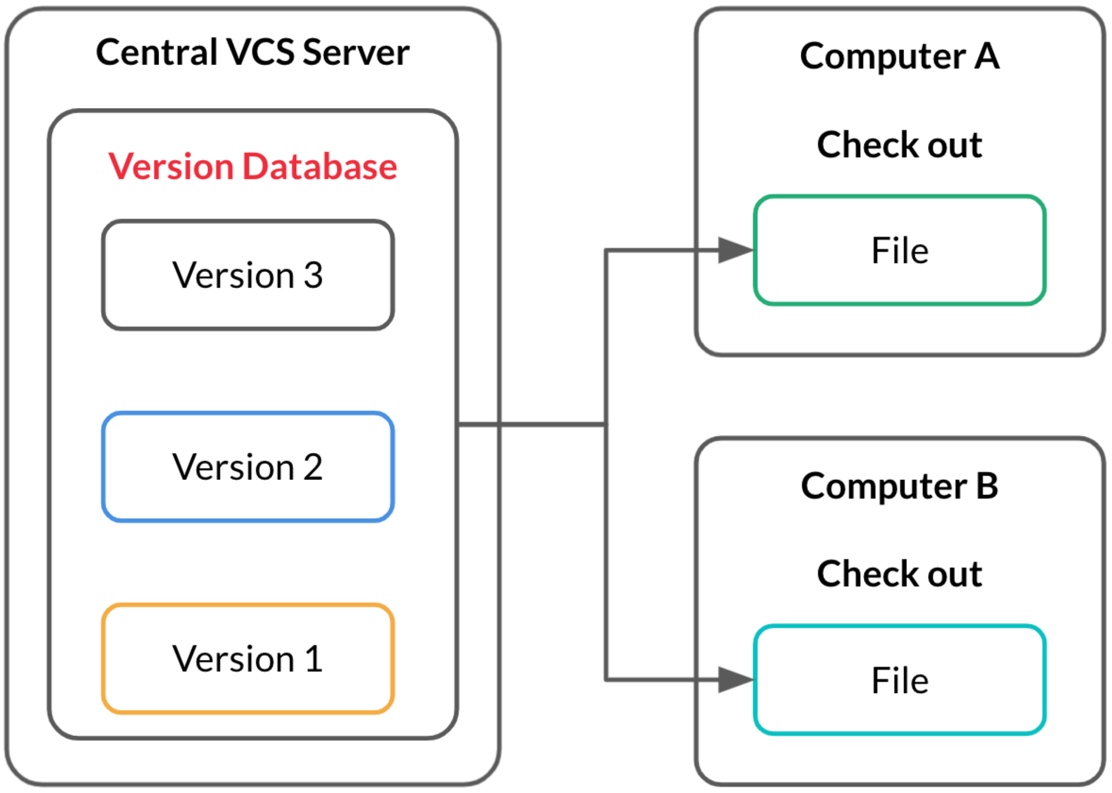
The Local Version control system maintains the database of different versions in the local computer. The problem with using the local version control system is that it is difficult to collaborate and disaster recovery after your local machine crashes is impossible.



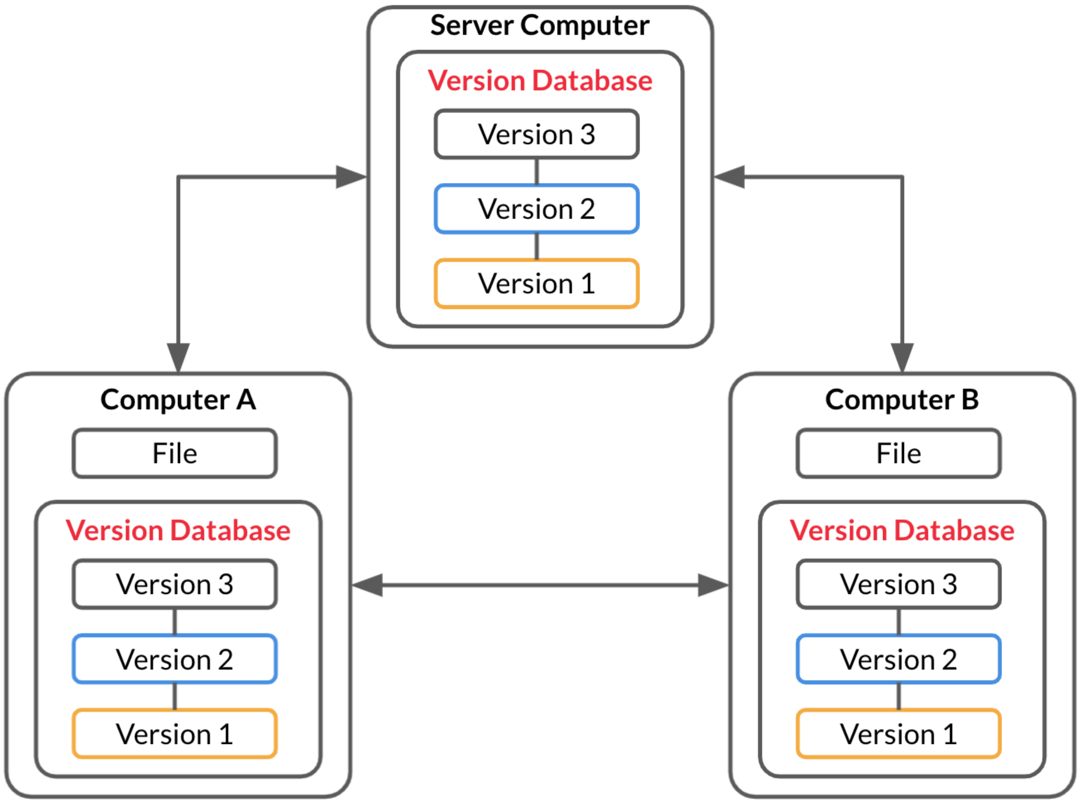
To fix these problems, thus, the Central Version control system (CVCS) came into existence. In CVCS, all the versions of a file were saved in a central server that enables collaboration.

It also eliminates the problems associated with local system crashes, as the probability of central server crashing is less than the probability of local system crashing.

However, as you can see in the following diagram, the developer needs to interact with the central server every time they need to save a version of the code. Further, there is a possibility of central server failure too.



Then, the Distributed Version control system (DVCS) came to the rescue by fixing all such problems. As shown in the following diagram, the entire database of different versions is present on the local machines of developers as well as the central server. Even if the central server crashes, this database is available on all the local machines as well. You might be thinking that this is a huge waste of storage. However, as the code files are text files, they take up minimal space.



As you learnt, git is based on the principles of DVCS.

Git came into existence because of a controversy between Linux developers and BitKeeper. It was developed by Linus Torvald who was the main developer of the Linux kernel, used by Linux distributions. The following was the first commit that Linus made in git and this is what he describes as Git.

```
GIT - the stupid content tracker

"git" can mean anything, depending on your mood.

 - random three-letter combination that is pronounceable, and not
   actually used by any common UNIX command.  The fact that it is a
   mispronunciation of "get" may or may not be relevant.
 - stupid. contemptible and despicable. simple. Take your pick from the
   dictionary of slang.
 - "global information tracker": you're in a good mood, and it actually
   works for you. Angels sing, and a light suddenly fills the room.
 - "goddamn idiotic truckload of sh*t": when it breaks

This is a stupid (but extremely fast) directory content manager.  It
doesn't do a whole lot, but what it _does_ do is track directory
contents efficiently.
```

Git and Github

There is a stellar difference between Git and GitHub.

- You learnt the differences between Git and GitHub. **Git** is a **distributed version control system**. It is a tool to manage your project source code history. **GitHub** is a **web-based Git file hosting service** that enables you to showcase or share your projects and files with others.
- You learnt a new term called ‘Repository’. A repository is a directory that contains your project work. All the files in the repository can be uploaded to GitHub and shared with other people either publicly or privately.
- Git was developed in 2005 and it became very popular across the world as a main software development tool across the world. As shown in the following diagram, it has been adopted by many fortune 500 companies, such as Google, Facebook, Microsoft, Twitter, LinkedIn, etc.

- As the popularity of git increased, many other repository hosting platforms came into existence, such as GitHub, Gitlab, BitBucket, etc. GitHub was the most popular repository hosting platform. GitHub provides the power to share your repository to the public and it also provides a graphical user interface.

- GitHub, which was launched in 2008, was widely adapted across several companies. It is presently being used by 65+ million developers, 3+ million organisations, 200+ million repositories and 72% of Fortune 50 companies.

Github Installation

Fraud detection is widely used across the financial services industries, which helps in saving huge amounts of losses. We will be creating a fraud detection project where we will use VCS to keep track of the project. Do not be alarmed if you do not know anything about machine learning yet. We will create a dummy fraud detection project as the main aim of this module is to understand how to use git and github in your project. The emphasis will be on the git commands and not on the project.

The following steps will be followed in the upcoming segments:

1. Installation and configuration of git and github
2. Implementation of basic git commands
3. Reverting back to previous stable versions of code if there is any mistake
4. Creating branches to understand the concept of branching

The first step is to create a github profile. As you have already learnt, git and github are two separate entities. You also need to install the git software in your local system.
The following document will help you understand the same in more detail.

https://cdn.upgrad.com/uploads/production/06a80281-8326-49af-b9f6-dba34330b7df/Git+-+Installation+Document.pdf

Basic Git Commands

1. The first step is creating a repository named 'fraud detection' in github.In creation of repository you came across the following three files:README.md, .gitignore and a License file
    a. README file is used to state an overview of what the repository contains,
    b. .gitignore files are the files that you do not want to be committed in the repository and
    c. the license file is used when you have created a software and you want to share it for others to use.
2. Now that you have created a repository in our github profile, you need to configure it with the local machine.
   Commands to be executed are:
```
echo "# fraud_detection" >> README.md #This command create a readme file and adds # fraud
detection as contents in the file.
git init
git add README.md
git commit -m "first commit"
git branch
git remote add origin https://github.com/sajankedia/fraud_detection.git
git push -u origin master
```

3. You have configured a git account with your local system. For the first-time Git configuration, you use the following commands. These commands are required only for the first-time configuration. For later instances, you do not need to run these commands.
```
git config --global user.name "yourusername”
Using this, you will enter your GitHub username
git config --global user.email “youremail@example.com”
Using this, you will enter your GitHub username
```

4. You need to install VS code. **You can find the installation instructions of VS code here.**

5. We created a file called data_processing.py in our local repository ‘fraud_detection’. Then, we wrote a dummy code in the data processing file and performed the following functions:

```
git add
git status
git commit -m “message”
git push -u origin master
```

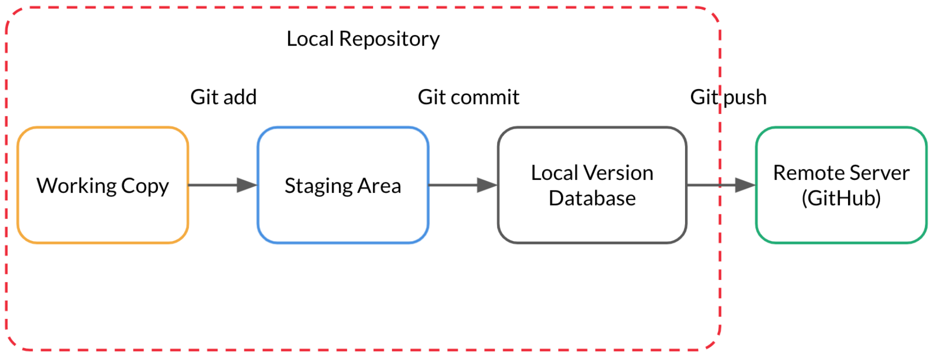
Let’s understand these commands in detail:

Git Command	Description
● 'git add <filename>' or 'git add .'	It is used to add modified files to the staging area. You can add a specific file using the command 'git add <filename>'. If you wish to add all modified and unstaged files present in the workspace to the staging area, you can use the 'git add .' command.
● 'git status'	This command will display the state of the working directory and the staging area. In other words, it lets you see the changes that have been staged and the changes that have not been added to the staging area.
● git commit -m “New commit message”	It gives a new commit message and commits all the files sitting in the staging area.
● git push -u origin master or git push	It is used to upload all the files and changes that were included in the most recent commit to your remote repository on GitHub.

In the creation of any project with version control, they are three main stages:
- Modified: When you change any code in the file.
- Staged: When we add the file using ‘git add’, it goes into staging area.
- Committed: When you commit all the changes in the files that were in the staging area.

Basic Git commands: click here.

Let’s summarise what you have learnt using a simple flow diagram.



As shown in this diagram,

- ‘git add’ puts the file that you are working on in the staging area where the git software keeps a track of the files,
- ‘git commit’ transfers it to the local version database and
- ‘git push’ transfers it to the remote repository in github.

Resetting and Reverting

1. We had written a function and committed it to your local repository. However, after committing the change, we realised that the function was not correct. In order to revert back to the previous commit, we used the **git revert** command

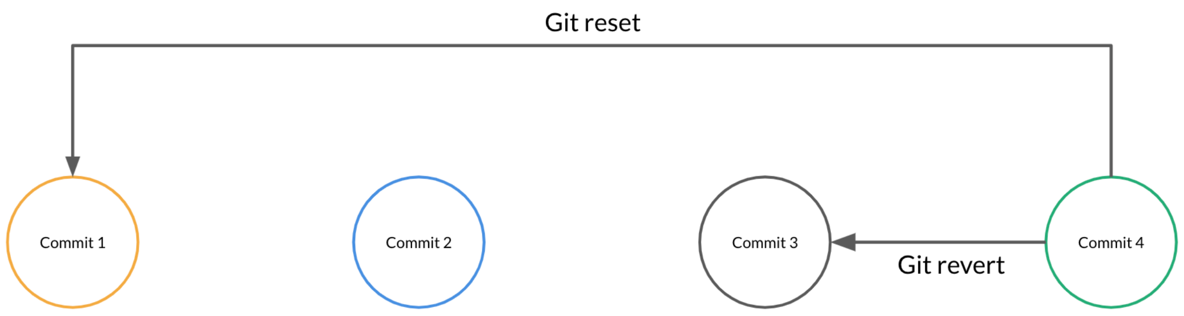
   ```
   git revert HEAD   → Reverts the project to the previously committed version
   ```

2. In a practical scenario where multiple data scientists or developers work on a project, there would be multiple commits happening round the clock. Suppose you found that there has been a mistake 10 days ago and you would want to revert back to that version. The first thing that might come to your mind would be to use ‘git revert HEAD’ multiple times; however, this isn’t a ideal approach. The ‘git reset’ command helps you in this scenario. To go back to a commit that was 3-4 commits back, the command

   ```
   git reset --hard <commit ID> was used.
   ```

Other Important Commands
git log: This command shows you the commit details. It lists out the commits made in the repository in reverse-chronological order, that is, the most recent commits show up first. It shows commits with the following details:
The commit ID or SHA
Author's name (who made the commit)
Date and time
For a shorter version of git log, you have git log --oneline

To summarise in this diagram, 'git revert' is used to go back to the previous version and 'git reset' is used to go back to any version.

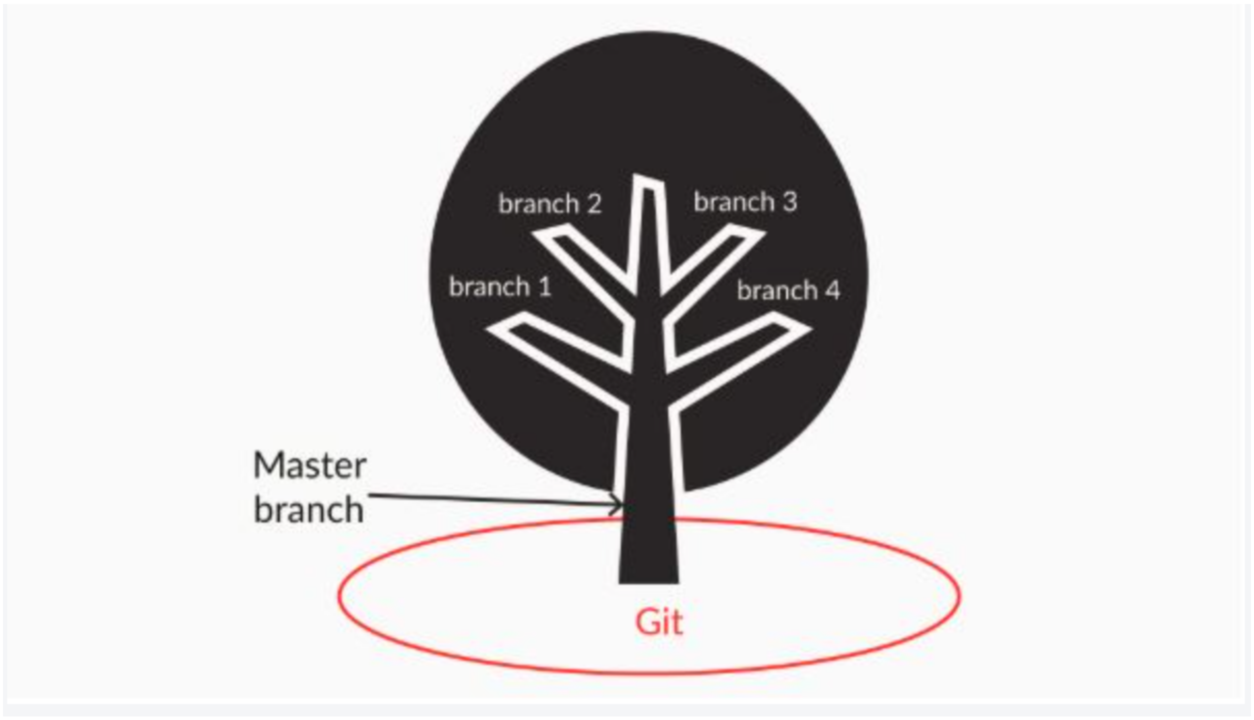


You need to be very careful about using 'git reset' as you will lose all the commits that you have done after the desired commit. Although 'git reset' and 'git revert' are good tools to get back to previous commits. It is not the best practice in the industry to experiment on a stable code. So, you use branching, wherein each individual developer can experiment on the code separately, and after successful experimentation, they can merge their branches to the master branch.

Branching

When you come across the term branching, you might correlate it to branches of a tree.

Well, yes! Branching means exactly the same. Imagine the branches growing out of the trunk of a tree. This trunk represents Git, as shown in the following illustration.



The trunk in this image plays the role of a master branch and the branches coming out of the trunk represent the branches in Git. For branching, following steps were followed:

1. In machine learning projects, after preprocessing the data, you proceed to the modelling step. In this process, you may not want to disturb the master branch with unnecessary experimentation, so you will create another branch named ‘model’.
2. Using this function, we created a file named ‘model.py’ and, later, pushed the code into our github repository.
3. We made some changes to the model.py file, and once the experimentation work was done and we had the final model ready, we then merged it to the master branch. In our case, ‘model2’ was the best performing model.
4. To merge it to the main branch, we used the github UI to create the pull request.
5. Pull request means to request the merger of branches. In our case, there were no conflicts and the status was shown as ‘able to merge’. Conflicts arise when you have different codes in the same file in different branches. Once the pull request was created, we went to the pull request tabs and merged the two branches.

To summarise, you learnt that there are situations where you may want to parallelly develop an existing project code, without making any changes to your initial/original branch. You can accomplish this goal by creating different branches based on your need (that is, creating a branch per team member or a branch for every new feature) and each branch will have the same copy of the initial/original branch of the project source code.

Introduction to Open source

You learnt how to use a version control system for the creation of a project. However, Git and Github are primarily used for collaboration among peers. You learnt that open source means anything that is available for use and modification under public license. Open Software has radically changed the digital ecosystem since its inception. The beginning of Linux as an Open Source, free-to-use OS served as a launchpad to this disruption.

You also learnt that many companies, such as Microsoft and Google, have open sourced their projects. VS code, which you have used a while back in this module, is itself an open source software. Big tech giants, such as Microsoft and Google, believe that open source development has great potential, so much so that Microsoft acquired GitHub in the year 2018.
Listed below are the links of the libraries:
https://github.com/Microsoft
https://opensource.google/projects/explore/featured
https://github.com/facebook
https://github.com/python
https://github.com/numpy

You can check out this article to take a look at famous open source repositories.

Contributing to open source has many benefits, such as:
1. You can contribute to real-world applications.
2. You can understand the structure of production-level codes.
3. You can build a network of contributors all around the world
4. You can leverage it for your career growth. In the tech industry, open source contribution is seen as a great achievement.

Contributing to Open Source

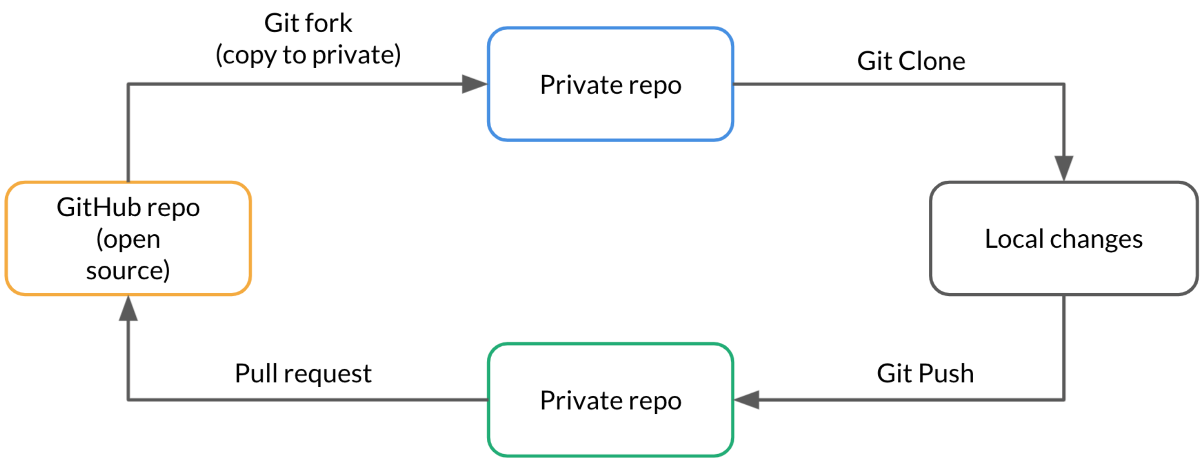
1. We forked the numpy library into our user. Forking a repository basically means copying the entire repository into your repository.
2. After forking the repository, you saw the different branches in the repository. Developers across the globe use different branches to develop different features in the library.
3. To download the code in your local repository, you use the ‘git clone <https link>’ command.

4. Now that you have the repository in your local machine, we created a feature in it. In the upcoming video, you will learn how to create a dummy feature and understand the process, as this module is about getting to know the process of contribution. In the course of the program, when you will be proficient in machine learning, we encourage you to take up open source contributions.
5. We created a new branch ‘add_dummy_feature’.
6. We created a file named ‘dummy_feature.py’ and, after writing our feature, we pushed the code into our remote repository.
7. After we pushed the code into our remote private repository, we created a pull request for our repository to merge it with the main branch of the numpy repository.
8. After creating the pull request, it is upto the admins of the repository, if they want to merge it or not. If they accept your request, your feature can be used by everyone all across the world. That is the power of open source.

Error: Expert has named the file as ‘dummpy_feature.py’, which should have been ‘dummy_feature.py’. The naming won't affect the process.

Note: As we created a dummy feature, our pull request won’t be merged with the main branch.

The process of contributing to the open source can be summarised as per the flow diagram shown below:



You can keep this diagram handy for contribution to open source.The same procedure can be followed even when you create a collaborative project. There will be only one more extra step of merging after creating a pull request.

Introduction to Portfolio Website Creation

As the popularity of github grew, many employers across the world started seeing github profiles as their virtual resume.
You can use github pages at https://pages.github.com/ to create your website. We learnt that github pages are a free feature of github through which you can host your website. Another main advantage of github pages is that it is powered by jekyll.
Jekyll is a software that allows you to convert the simple readable markdown texts (that we used to create a README file) into html, CSS and javascript, which means you do not need to be a great web developer to create your own website.

Github provides the themes of various websites of jekyll, which can be used directly by forking the desired repository to your private repository and making the changes there.
You can find several themes of Jekyll here.
If you want an advanced website, then you can find the jekyll themes here as well - https://jekyllthemes.io/. Although some of these themes are freely available, there are some paid versions as well.

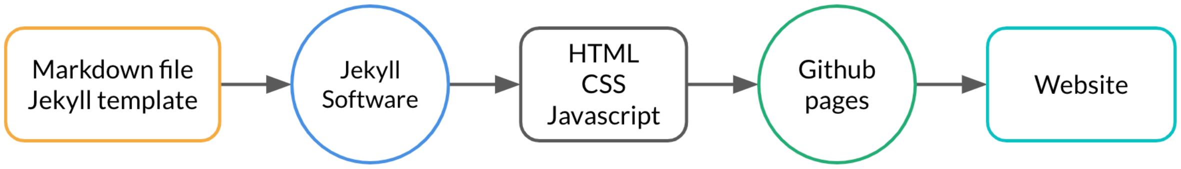
For our demonstration, we have taken the jekyll theme: **minimal** and made changes in the files of the repository.
The following major steps were followed here:

1. Change index.md.
2. Change config.yml.
3. Change the pictures.

For more markdown commands, you can check out the sheet here.
Now that you have changed the appropriate files, you can now proceed to publishing.
For publishing, the github pages will be used. The following steps are followed for publishing the site:

1. Go to the Settings tab.
2. Go to the Pages tab in the left hand corner.
3. You might see that your website is already published, but if not, you can click on ‘publish’ and make sure your repository is public. It might take a few minutes for the website to get published.
4. After publishing, you can share this URL with anyone and it will be your virtual resume.

Although you have learnt how to use jekyll themes to create a website of your own, you can publish any website from github pages. You can read more about the github pages and their documentation here.



We changed the markdown file and the jekyll template in config.yml. These files were taken by the Jekyll software and converted into HTML, CSS and Javascript files at the backend. These files were fed into Github Pages, which then created the Website.

We have created a basic version of the website for demonstration; however, we recommend you to explore and make great websites for yourself.

Throughout the program, you will be asked to push all the code that you wrote in the assignments, case studies and capstones into your github repository. When you finish the course, you will have a robust profile of ML and AI projects. Don't be worried if you don't have many projects to showcase, but do keep on updating the github repository and github website.

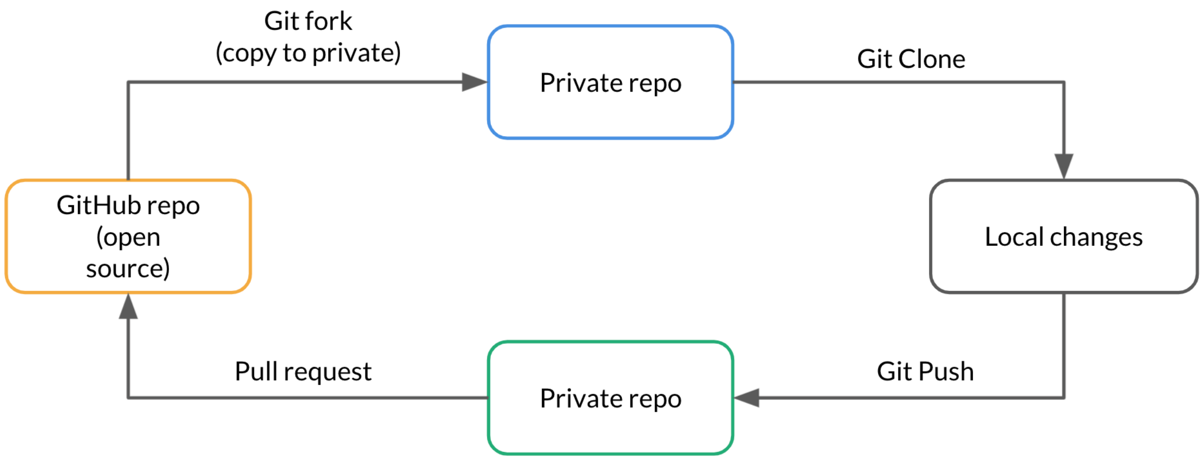
Interview Questions:

What is the main utility of Git and GitHub? What is the difference between them?

Git and GitHub are used for Version Control. Git is a distributed version control system. It is a tool to manage your project source code history. GitHub is a web-based Git file hosting service that enables you to showcase or share your projects and files with others.

What is the flow of commands used for contributing to an open source project?

1. Git fork
2. Clone
3. Make the changes in local system
4. Git push
5. Pull request



What is the difference between 'git fork' and 'git clone' commands?
Git fork is used to copy the open source project into your private GitHub repository and git clone is used to download any GitHub repository into your local system

What are the uses of branching?
Branching is used when you want to perform experimentation either in your individual project or a group project. We can edit our code in these branches without disturbing the stable code.

Can you tell the differences between git revert and git reset?
In order to revert back to the previous commit, the **git revert** is used.
In order to go back to a commit that was >2 commits back, the git reset is used.

In a practical scenario where multiple data scientists or developers work on a project, there would be multiple commits happening round the clock. Suppose you found that there has been a mistake 10 days ago and you would want to revert back to that version. The first thing that might come to your mind would be to use 'git revert HEAD' multiple times; however, this isn't a ideal approach. The 'git reset' command helps you in this scenario.

git revert creates a new commit which shows that you have reverted back to the previous commit whereas git reset goes back to the specified commit ID and erases all the commits after that

What is a commit message?
git commit requires a commit message so that developers can identify what changes was made in that particular commit.