Course Introduction

W​elcome to this course on Git and GitHub!

**D​istributed Version Control Systems (DVCS) have become critical tools in software development,** and key enablers for **social and collaborative coding**. They are not only being used by Software Engineers and DevOps professionals but also by many other technology practitioners such as Data Scientists and Data Engineers.

However their usage is not limited to coding professions only. They are useful anywhere tracking changes/versions and/or collaboration between multiple users is required. At IBM Skills Network, the course instructors and authors use Git repositories extensively even for developing course content such as lab instructions. You will also find usecases in technical documentation, legal document management, and even collaborative development of recipes, books, etc.

While there are many distributed versioning systems, Git is amongst the most popular ones. And **GitHub is a highly popular Git-based hosted version control platform**, and is seeing incredible growth. When some of the videos for this course were developed couple of years earlier, there were over 100 million GitHub repositories, whereas at the time of writing, January 2022, they have grown to over 200 million repositories. These include both public and private repositories for both open source and closed source projects.

The popularity of **Git and GitHub** make their use **an essential skill for coding-related professionals** like Software Engineers, Application Developers, Mobile Developers, DevOps & Site Reliability Engineers, Data Scientists, and Data Engineers. W​hen you try to get a software-related job or switch to a different one, employers expect you to provide links to your GitHub profile on your resume.

I​n this course you will **develop the essential conceptual and hands-on skills to work with Git and GitHub**. We will start with an overview of Git and GitHub, followed by **creation of a GitHub account and a project repository**, adding files to it, and committing your changes using the web interface.

Next, you will become familiar with **Git workflows involving branches and pull requests (PRs) and merges**. You will learn to **fork and clone** public repositories, use **pull and push** to synchronize your codebase between local and remote repositories, and **practice working with Git commands** for use in collaborative development workflows. Y​ou will also **complete a project** at the end to apply and demonstrate your newly acquired skills.

I​f you require any clarifications or help, feel free to post on the course discussion forums to interact with your peers and get assistance from the course team.

H​ave fun and best wishes!

Y​our course instructors,

Rav Ahuja and Upkar Lidder

**Overview of Version Control, Git, and GitHub**

In this video, you’ll get an overview of Git and GitHub, which are popular environments

among developers and data scientists for performing version control of source code files and projects and collaborating with others. You can’t talk about Git and GitHub without a basic understanding of what version control is.

A version control system allows you to keep track of changes to your documents.

This makes it easy for you to recover older versions of your document if you make a mistake,

and it makes collaboration with others much easier. Here is an example to illustrate how version control works. Let’s say you’ve got a shopping list and you want your roommates to confirm the things you need and add additional items. Without version control, you’ve got a big mess to clean up before you can go shopping. With version control, you know exactly what you need after everyone has contributed their ideas

Git is free and open source software distributed under the GNU General Public License.

Git is a distributed version control system, which means that users anywhere in the world

can have a copy of your project on their own computer. When they’ve made changes, they

can sync their version to a remote server to share it with you. Git isn’t the only version control system out there, but the distributed aspect is one of the main reasons it’s become one of the most common version control systems available. Version control systems are widely used for things involving code, but you can also version control images, documents, and any number of file types. You can use Git without a web interface by using your command line interface, but GitHub is one of the most popular web-hosted services for Git repositories. Others include GitLab, BitBucket, and Beanstalk. There are a few basic terms that you will need to know before you can get started. The SSH protocol is a method for secure remote login from one computer to another. A repository contains your project folders that are set up for version control. A fork is a copy of a repository. A pull request is the way you request that someone reviews and approves your changes before they become final. A working directory contains the files and subdirectories on your computer that are associated with a Git repository.

 There are a few basic Git commands that you will always use. When starting out with a new repository, you only need create it once: either locally, and then push to GitHub, or by cloning an existing repository by using the command "git init".

"git add" moves changes from the working directory to the staging area. "git status" allows you to see the state of your working directory and the staged snapshot of your changes. "git commit" takes your staged snapshot of changes and commits them to the project. "git reset" undoes changes that you’ve made to the files in your working directory. "git log" enables you to browse previous changes to a project. "git branch" lets you create an isolated environment within your repository to make changes. "git checkout" lets you see and change existing branches. "git merge" lets you put everything back together again.

To learn how to use Git effectively and begin collaborating with data scientists around the world, you will need to learn the essential commands. Luckily for us, GitHub has amazing resources available to help you get started. Go to try.github.io to download the cheat sheets and run through the tutorials.

In the following modules, we'll give you a crash course on setting up your local environment

and getting started on a project.

**Introduction to GitHub**

Welcome to Introduction to GitHub After watching this video, you will be able to:

Describe the purpose of source repositories and explain how GitHub satisfies the needs of a source repository. Linux development in the early 2000’s was managed under a free-to-use system known as BitKeeper. In 2005, BitKeeper changed to a for-fee system which was problematic for Linux developers for many reasons. Linus Torvalds led a team to develop a replacement source-version control system. The project ran in a short a timeframe and the key characteristics were defined by a small group.

These include:

Strong support for non-linear development. (Linux patches were then arriving at a rate of 6.7 patches per second) Distributed development. Each developer can have a local copy of the full development history. Compatibility with existing systems and protocols.

This was necessary to acknowledge the diversity of the Linux community. Efficient handling of large projects. Cryptographic authentication of history. This makes certain that distributed systems all have identical code updates. Pluggable merge strategies. Many pathways of development can lead to complex integration decisions that might require explicit

integration strategies. What is special about the Git Repository model? Git is designed as a distributed version-control system. Primarily focused on tracking source code during development. Contains elements to coordinate among programmers, track changes, and support non-linear workflows. Created in 2005 by Linus Torvalds for distribution of Linux kernels.

Git is a distributed version-control system that is used to track changes to content.

It serves as a central point for collaboration with a particular focus on agile development

methodologies. In a central version control system, every developer needs to check out code from the central system and commit back into it. As Git is a distributed version control, each developer has a local copy of the full development history, and changes are copied from one such repository to another. Each developer can act as a hub. When Git is used correctly, there is a main branch that corresponds to the deployable code.

Teams can continuously integrate changes that are ready to be released and can simultaneously work on separate branches in between releases. Git also allows centralized administration of tasks with access-level controls for each team. Git can co-exist locally such as through the GitHub Desktop client or it can be used directly through a browser connected to the GitHub web interface. IBM Cloud is based on sound and established open-source tools including Git repositories, often called repos. GitHub is an online hosting service for Git repositories. GitHub hosted by a subsidiary of Microsoft. GitHub offers free, professional and enterprise accounts. As of August 2019, GitHub had over 100M repositories.

A Repository is: A data structure for storing documents including application source code.

A repository can track and maintain version-control. GitLab is a complete DevOps platform, delivered as a single application. GitLab provides access to Git repositories, controlled by source code management.

With GitLab, developers can: Collaborate, reviewing code, making comments and helping to improve each other’s code. Work from their own local copy of the code.

Branch and merge code when required. Streamline testing and delivery with Built-in Continuous Integration (CI) and Continuous Delivery (CD).

In this video, you learned: GitHub is the online hosting service for Git repositories.

Repositories store documents including application source code and enable contributors to track and maintain version-control. What is special about the Git Repository model?

Git is designed as a distributed version-control system. Primarily focused on tracking source code during development. Contains elements to coordinate among programmers, track changes, and support non-linear workflows.

**GitHub Repositories**

(Music)

Welcome to GitHub Repositories! After watching this video, you will be able to: Explain how to sign up for a GitHub account and describe how to create a repository. Signing up for a free, personal account on GitHub is quick and easy. Start at the GitHub site, https://github.com

You’ll need to choose a username, enter your email address and select a password, then click Sign up for GitHub. Next, you’ll have a short test to prove that you’re a person.

Click Verify and solve the puzzle presented. When you’re done, click join a free plan and then you’ll be taken to a screen where you can select the type of account – most likely a free, personal account is all that you’ll want. Choose to set up a personal, free account, which is the default. GitHub asks some questions about your work, programming experience and interests. You can skip these if you want.

Finally, you’ll have to respond to an email that you receive which proves that you linked

to GitHub from an account that you access GitHub provides you with some starting points.

You can choose to create a repository or an organization, or you can take the Introduction

to GitHub course. Remember, a repository is a data structure for storing documents including application source code which tracks and maintains version-control. An organization is a collection of user accounts that owns repositories. Organizations have one or more owners, who have administrative privileges for the organization. Or you can skip this for now and get straight to work. GitHub provides many resources to help you work effectively. When you have time, read the GitHub guide. The heart of a Git-based project is the repository.

This contains all your code and the related artifacts, including things like: A README file to describe the purpose of the project. A license to express the ways in which people can use your code, Etc. You can also make your repository private (only available to people with accounts that have permission to see it) or public (searchable and seen by everyone).

When you create your repository, you’ll notice that it has a number of tabs, and is 37 00:02:34,310 --&gt; 00:02:37,069 opened to the Code tab.

Code – this is where all the source files reside. Git was initially created as a source code repository and now all sorts of files end up in here. If you created a README and/or license, that’s all that’s here right now. Issues – as you can imagine, you can track and plan with tools such as “Issues” that lists all open items against your project base. Pull Requests – this is part of the mechanism for collaborating with other users. Pull requests define changes that are committed and ready for review before being merged into the main branch. Projects – all the tools for managing, sorting, planning, etc. your various projects. This is the core of the collaborative power of GitHub. Wiki, Security, and Insights – often left for more advanced users, these tools provide a communication base to the external user community.

Settings – GitHub allows for a lot of personalization, including changing the name of your repository and controlling access.

In this video, you learned:

How to create and verify a GitHub account. Repositories are storage structures that can hold Code, track Issues, and enable you to collaborate with others.

GitHub - Getting Started

In the previous video, you learned about Git  and GitHub. Before you continue with this video,   register for a GitHub account and log in.  Let’s start by creating a new repository.

Click + then click New Repository. To  create a new repository, you need to provide

these details: give your new repository a name;  optionally, add a description of your repository;   choose the repository visibility - whether you  want it to be public or private; and choose the option   to Initialize this repository with readme file. Then click Create Repository.

You will now be redirected to the  repository you have created.   The root folder of your repository is listed  by default and it has just one file ReadMe.md.  Now, it’s time to edit the readme. You can do this  in your browser. Just click the pencil to open the   online editor and you can change the text of the  readme. To save your changes to the repository,   you must commit them. After you have made your  changes, scroll down to the Commit changes   section. Add a commit message and optionally add  a description, then click Commit changes. The   "commit changes" is used to save your changes  to the repository. Go back to the home screen by   clicking the repository name link. Note that the  readme file is updated and verify your changes.

Let’s learn how to create a new file using the  built-in web editor provided by GitHub which

runs in the browser. Click Add File, then  click Create New File to create the new file.

To create a python file called firstpython.py.

First, provide the file name. Next, add a comment  that describes your code, then add the code

Once finished, commit the change to the  repository. You can see that your file is

now added to the repository and the repository  listing shows when the file was added or changed.   When you need to change the  file, you can edit it again.   Click the file name, and then click the pencil  icon, make your edits and commit the changes

You can also upload a file from your  local system into the repository.   From the home screen of the repository, click  Add File and choose the Upload files option.

Click Choose Your Files and select the files  you want to upload from your local system.

The file upload process may take a short time,   depending on what you are uploading.  Once the files finish uploading, click Commit Changes. The repository now reflects  the files that were uploaded. In this video, you learned how to create a repository, edit  files, and commit changes using the web interface.

**Hands-on**

**Module Summary**

In this module, you learned that:

A Distributed Version Control System (DVCS) keeps track of changes to code, regardless of where it is stored. This allows multiple users to work on the same codebase or repository, mirroring the codebase on their own computers if needed, while the distributed version control software helps manage synchronization amongst the various codebase mirrors.

Repositories are storage structures that:

* Store the code
* Track issues and changes
* Enable you to collaborate with others

G​it is one of the most popular distributed version control systems. GitHub, GitLab and Bitbucket are examples of hosted version control systems.

**Practice Quiz**

**Graded Quiz**