**Git & Git hub**

* **Distributed 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
* 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
* 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

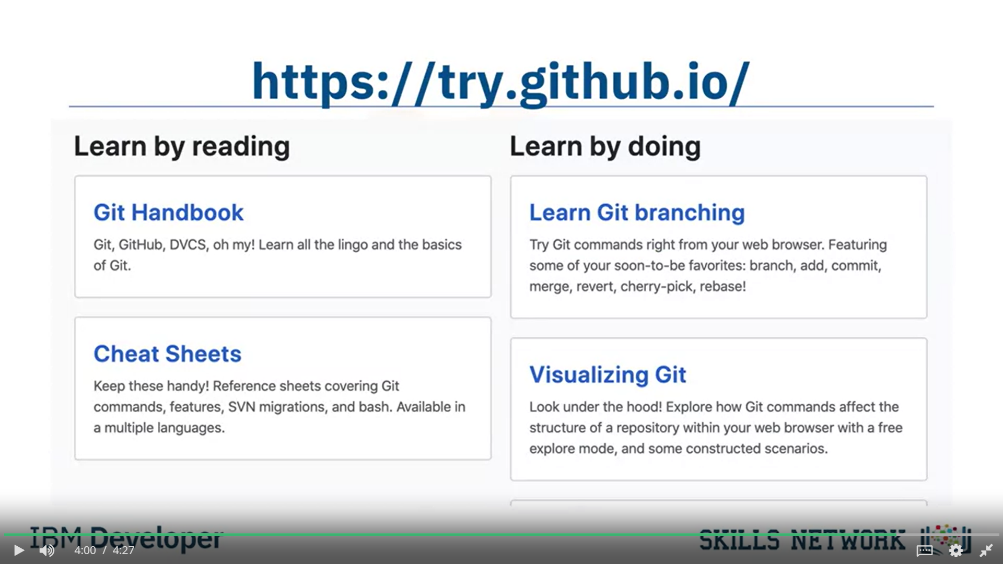
**Git:**

* 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.
* 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.
* GitHub is one of the most popular web-hosted services for Git repositories, others include GitLab, BitBucket, and Beanstalk.

**Basic terms:**

* 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.

**Basic Git commands:**

* 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.

**Github:**

* 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.
* 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.
* GitHub is an online hosting service for Git repositories.
* GitHub hosted by a subsidiary of Microsoft.
* GitHub offers free, professional and enterprise accounts.
* A Repository is:
  + A data structure for storing documents including application source code.
  + A repository can track and maintain version-control.

**GitLab**

* It 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).