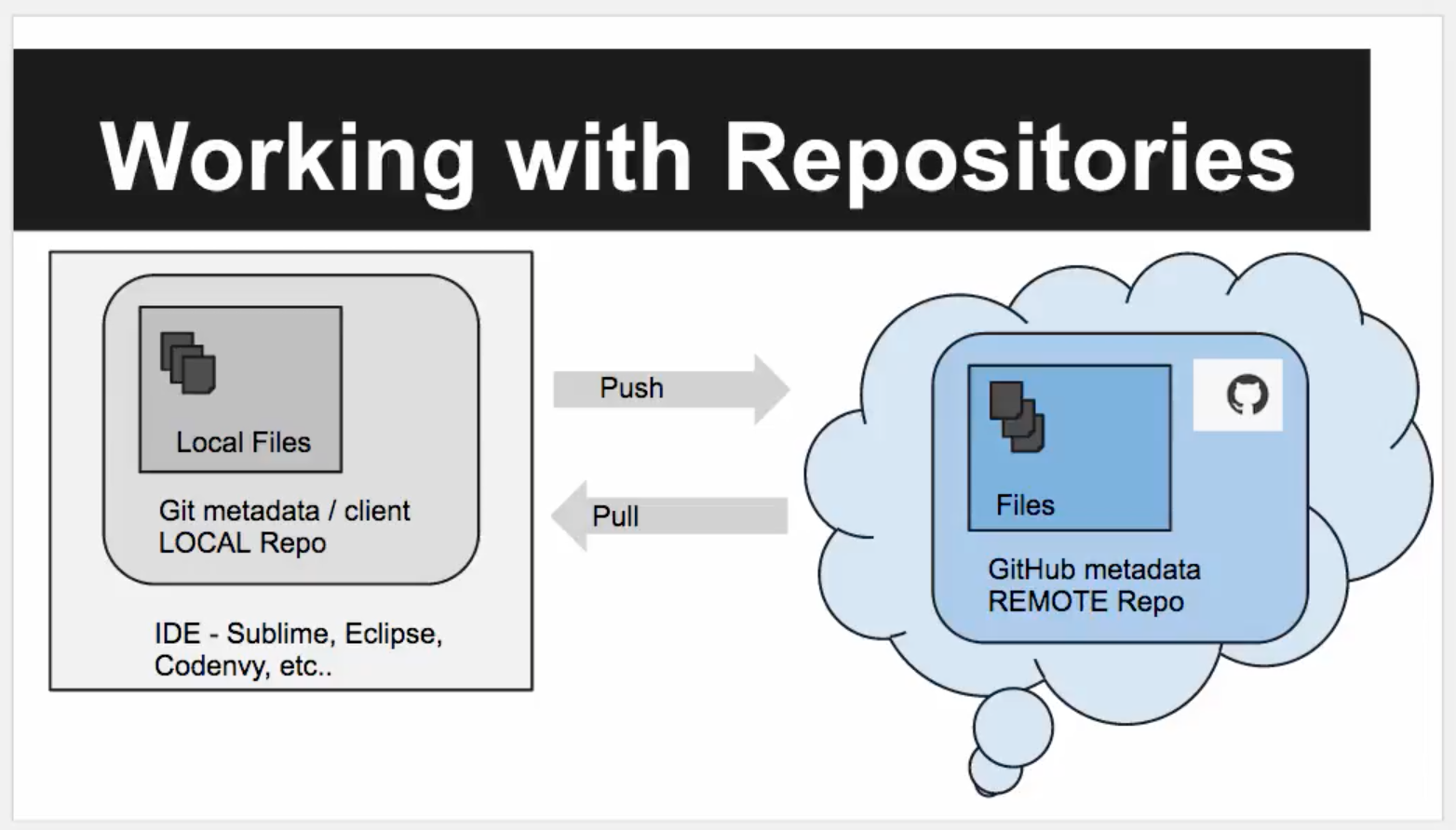
# **Using GitHub Source Control with TKP Students**

Code source control systems provide an easy way to store **all** versions of all of your code files for **all time**. This capability is vital for modern software development as software is emergent. The most commonly used source control system is GitHub. This is a public website that offers free storage for public code files. It provides an interface around the git source control system.



## **Concepts**

1. **** **Source Code Files** - your code files, for TKPJava, for example *'SimpleSquare.java'*
2. ** **IDE** - your code editor, we use Eclipse or Codenvy
3.  **Open Source Code Files** - your code files copied to a public open source system (website such as GitHub) is called OSS development
4.  **Source Control Systems** - allows storage of all versions of your code files. These systems commonly have both a *local* source control client application and also connect to a *remote* server, so there are **two** copies of all code files.
5.  **Repository or Repo** - your container that holds all of the code files and any other files need for your project, other files can be images, text files, etc.... Repos can also contain child folders.
6.  **GitHub** - a popular open source control system and a public website. It is free for hosting **public** code repositories. NOTE: GitHub also has a commercial offering, which allows clients to have **private** code repositories.
7.  **GitHub Desktop**- a free, standalone desktop application GitHub Desktop (rather than integration to git via an IDE) to connect our local Repos to their remote GitHub counterparts.

## **Steps in Getting Started**

Because learning version control systems is a somewhat daunting task, even for professional developers, we at TKP have decided to take the 'minimum viable source control' approach to guide teachers in *getting starting* using GitHub with their TKPJava students.

# **GitHub QuickStart For TKP Teachers**

1. **Create a folder** on your local machine for your own example repository (repo).
   1. Navigate to the root C: drive, or an alternate location, i.e. ‘Downloads’ folder if you can't access C:
   2. Make a new folder named ‘hello-world’ in the that location
2. **Register for a free account** on [GitHub.com](https://github.com/)
   1. Note your credentials, you’ll need them for the next step
   2. Leave GitHub.com open for use in subsequent steps of this guide
3. **Download & Install the free** [GitHub Desktop](https://desktop.github.com/) tool
   1. In the ‘Welcome’ dialog box
      1. **Log in** with your Github.com Username and Password
      2. **Configure git** by entering your email and click ‘Continue’
      3. **Repositories** - you will see ‘No local repositories found’
      4. **Click on ‘Dashboard’** to go on
   2. In the GitHub Desktop Dashboard
      1. **Delete** the default ‘Tutorial’ Repo by right-clicking->remove
      2. **Add** our ‘hello-world’ folder as a new local GitHub Repo
         1. **Click** on the big **‘+’ (plus)** sign in the upper-left corner of the Dashboard
         2. **Click** on **‘Add’**.
         3. **Click** on ‘Browse’ (or 'Choose' if using a Mac) and then navigate to the ‘hello-world’ folder. Mine is on my C drive but you may need to click ‘This PC’ in order to be able to see the C: drive. My final path is: C:\hello-world
         4. **Click** on the blue words, **‘C**\*\*reate & Add Repository’\*\* and then click on ‘\*\*Create Repository\*\*’
      3. **See** your ‘hello-world’ folder appear on the left column as a git repository, but, right now it only exists locally (on your computer). This is indicated by the little icon to the left of the words ‘hello-world’ appearing as a computer monitor. We want to sync our local Repo with a remote version (copy) of itself in the GitHub.com cloud. To create a remote Repo on GitHub.com we need to ‘Publish’ our local ‘hello-world’ Repo.
      4. **Verify** that ‘hello-world’ is not yet created remotely in your GitHub.com account, go to your GitHub.com account and look to see if you notice a Repo there named ‘hello-world’. You will not see it there yet.
      5. **Click** on the dark ‘Publish’ button on the upper-right side of the Dashboard. NOTE: if you are prompted to Log in to GitHub, enter your [GitHub.com](http://github.com/) credentials
      6. **Add** a ‘Description’. Type something like, ‘trying my first GitHub repo’. NOTE: Do use human-readable check in comments.
      7. **Click** ‘Publish hello-world’. GitHub Desktop syncs your local Repo to a newly created ‘hello-world’ Repo and changes the Dashboard icon from a monitor to a little book with a bookmark icon when the sync is done
      8. **Refresh** your online GitHub.com account page in your browser
      9. **Verify** that you can now see your remote ‘hello-world’ Repo in your online [GitHub.com](http://github.com/) account. If you don’t, go back to GitHub Desktop Dashboard and notice that the ‘Publish’ button has changed to say ‘Sync’. Go ahead and click ‘Sync’. Now you should see the Repo on GitHub.com.
      10. **Congratulations!** You’ve just created your first GitHub repository with a both a local and remote copy.

### **TIPS & LINKS**

* GitHub **requires** that students are [13 years old](https://help.github.com/articles/github-terms-of-service/) or older to use their service
* Students need 'install' permissions (usually administrator) on their computer to install GitHub Desktop
* See 5 minute intro [here](https://github.github.com/on-demand/intro-to-github/)
* Get desktop client [here](https://desktop.github.com/)
* Get the 'student developer pack' [here](https://education.github.com/pack)
* See 15 minute on-line tutorial [here](https://try.github.io/levels/1/challenges/1)
* GitHub getting started guide [here](https://guides.github.com/activities/hello-world/)
* 4 short 'Commit yourself GitHub' screencasts [here](https://www.youtube.com/watch?v=HmYaZWU97GM)

## **Next Steps**

GitHub includes many other concepts, including making distributed copies via forking and branching. These concepts are often used by professional software developer teams. For a concise explanation - [see this link](https://guides.github.com/introduction/flow/).

Happy OSS Programming!

The TKP Team