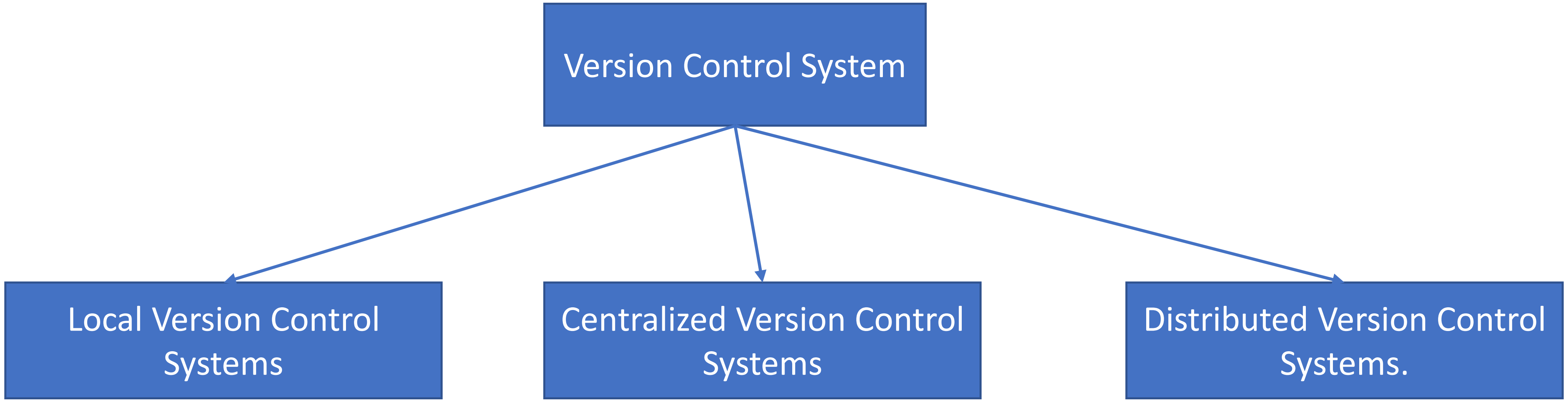
Md. Rasel

Git and Github for version controlling

Version Control ?

A Version Control System is a tool you use to track, make, and manage changes to your software code. It's also called source control.
A version control system helps developers store every change they make to a file at different stages so they and their teammates can retrieve those changes at a later time.

Types of Version Control


Version Control System
Local Version Control Systems
Centralized Version Control Systems
Distributed Version Control Systems.

Types of Version Control

- ❖ **Local Version Control System**

In This type of Version control system the files and codes are store in local system and very handy to use this tools. But it is risky to handle data and code on local system for error and attacks.

- ❖ **Centralized Version Control System (CVCS)**

a server act as a repository that stores each version of the code. The CVCS helps different developers collaborate together. Despite the helpful collaboration and communication between developers, if a server goes down for few seconds or gets corrupted, there's the chance you'll lose your work. Unfortunately, this is a very big problem with the CVCS.

- ❖ **Distributed Version Control System (DVCS)**

In this type of system developer have full backup of their codes in a clone directory, if the server get down developers can still work as they have backup repository to the server to restore them.

In this type of DVCS many developers can work together simultaneously.

Few popular version Control system

git

CVS

Microsoft Visual Studio Team Foundation Server

IBM Rational ClearCase

Git

Git is a tools to control your code version which lets you track changes you make to your files over time. With Git, you can revert to various states of your files. We can also make a copy of your file, make changes to that copy, and then merge these changes to the original copy.

We are not limited in Git just for source code file we can even keep track of our text and images also.

GitHub

GitHub is a working repository of a project where we can store our code from remote location. We can pull and push our code to that repository from remote client.

Git Command

repo -> repository

clone -> bring a repo down from the internet (remote repository like Github) to your local machine

add -> track your files and changes with Git

commit -> save your changes into Git

push -> push your changes to your remote repo on Github (or another website)

pull -> pull changes down from the remote repo to your local machine

status -> check to see which files are being tracked or need to be commited

init -> use this command inside of your project to turn it into a Git repository and start using Git with that codebase

Git State

Committed state
A file is in the **committed** state when all the changes made to the file have been saved in the local repo. Files in the committed stage are files ready to be pushed to the remote repo (on GitHub).

Modified state
A file in the **modified** state has some changes made to it but it's not yet saved. This means that the state of the file has been altered from its previous state in the committed state.

Staged state
A file in the **staged** state means it is ready to be committed. In this state, all necessary changes have been made so the next step is to move the file to the commit state.

Installing and Configuring Git

You can install git from below link - https://git-scm.com/download/win

Install as per guidance. After that lets Start working with Git.

```
$ git config --global user.name "YOUR_USERNAME“

$ git config --global user.email "YOUR_EMAIL“

$ cd desktop
$ mkdir demoforclass
$ git init
$ git add .
$ git commit -m "first commit“
```

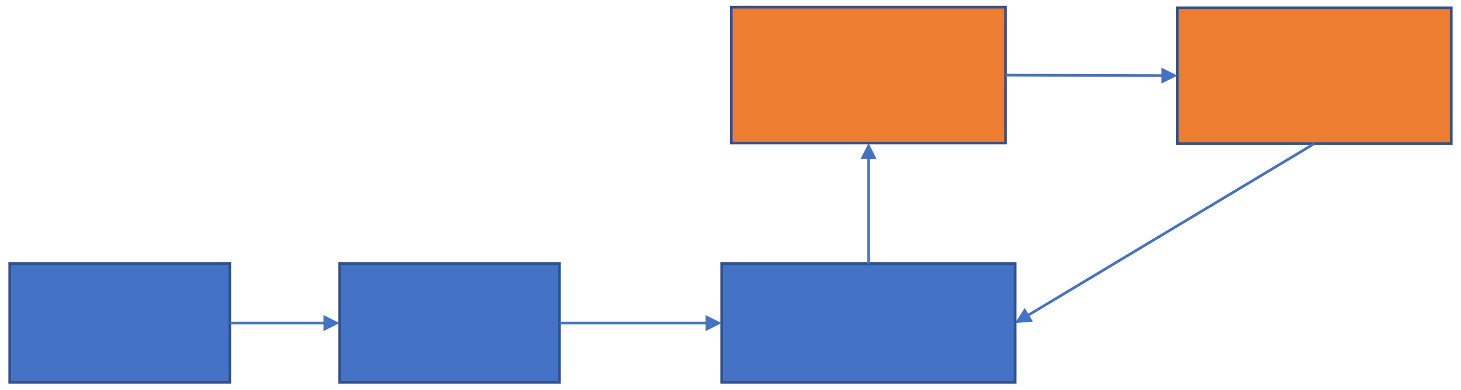
Pushing your file to a GitHub repo

In This stage we have make a test file and change it status to committed and now we are ready to push our code to a GitHub repository which we earlier created.

```
$ git branch -M main
$ git remote add origin https://github.com/MdRasel0/democlass1.git
$ git remote –v
$ git push -u origin main
```

Git Branches

create a copy of a file you would like to work on without messing up the original copy. You can either merge these changes to the original copy or just let the branch remain independent.



Command for Branch

Making a new branch

```
$ git checkout -b test

$ git add .
$ git commit –m “branch commit”
$ git checkout main
$ git branch
$ git merge test
$ git push -u origin main
```

remote main ;
git push -f
origin origin

جَزَاكَ اللّٰهُ خَيْرًا