2. What are the difference between Git and GitHub

Git	GitHub
• Git is a tool a developer installs locally on their computer. • **Git** is a revision control system, a tool to manage your source code history. • Git is a free and open-source distributed **version control system** designed to handle everything from small to very large projects with speed and efficiency. • Git is a distributed peer-peer version control system. Each node in the network is a peer, storing entire repositories which can also act as a multi-node distributed back-ups.	• GitHub is an online service that stores code pushed to it from computers running the Git tool. • **GitHub** is a hosting service for Git repositories. • GitHub is a **web-based** Git repository **hosting service**, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. • Github provides access control and several collaboration features such as wikis, task management, and bug tracking and feature requests for every project.

• Git competes with centralized and distributed version control tools such as Subversion, Mercurial, ClearCase and IBM's Rational Team Concert.	• GitHub competes with cloud-based SaaS and PaaS offerings, such as GitLab and Atlassian's Bitbucket.

3. What is Git Workflow

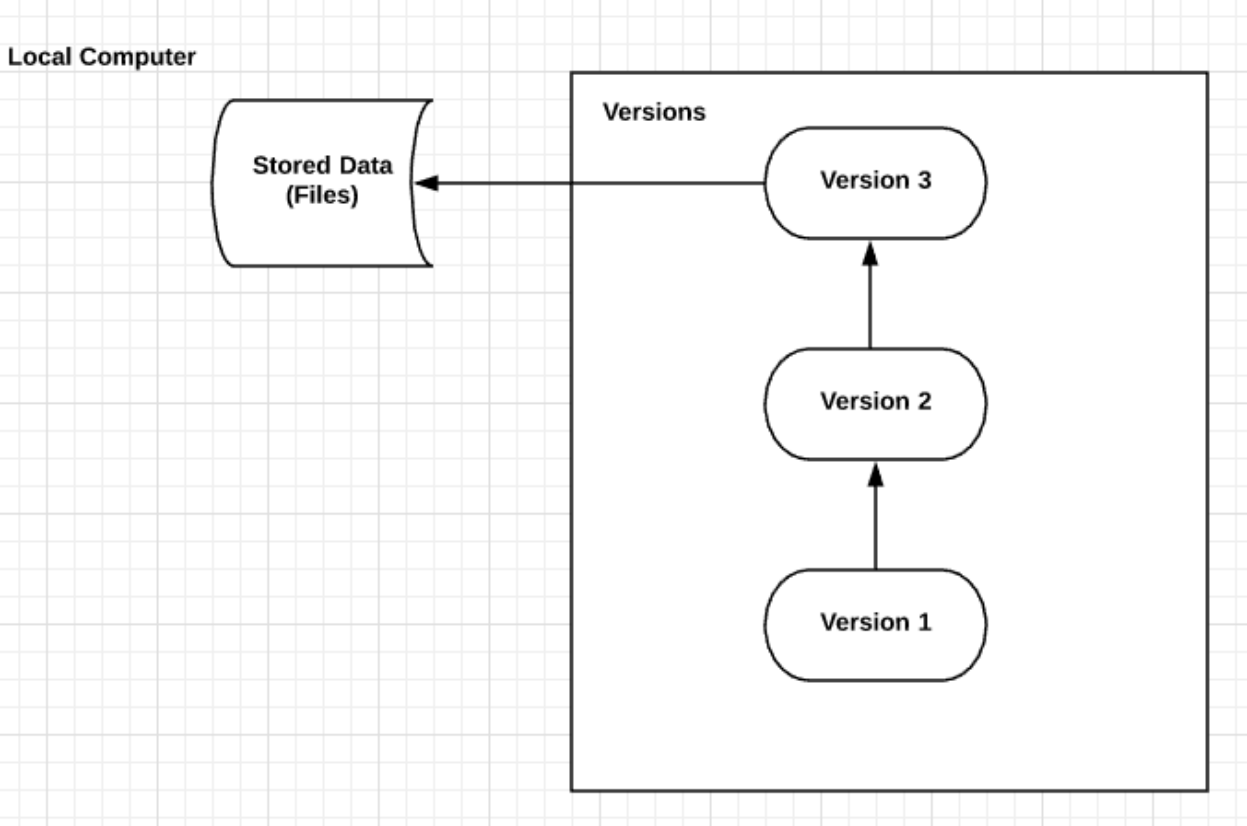
A Git Workflow is a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner. Git workflows encourage users to leverage Git effectively and consistently. Git offers a lot of flexibility in how users manage changes. Given Git's focus on flexibility, there is no standardized process on how to interact with Git. When working with a team on a Git managed project, it’s important to make sure the team is all in agreement on how the flow of changes will be applied. To ensure the team is on the same page, an agreed upon Git workflow should be developed or selected. There are several publicized Git workflows that may be a good fit for your team. Here, we’ll be discussing some of these workflow options.

The array of possible workflows can make it hard to know where to begin when implementing Git in the workplace. This page provides a starting point by surveying the most common Git workflows for software teams.

As you read through, remember that these workflows are designed to be guidelines rather than concrete rules. We want to show you what’s possible, so you can mix and match aspects from different workflows to suit your individual needs.

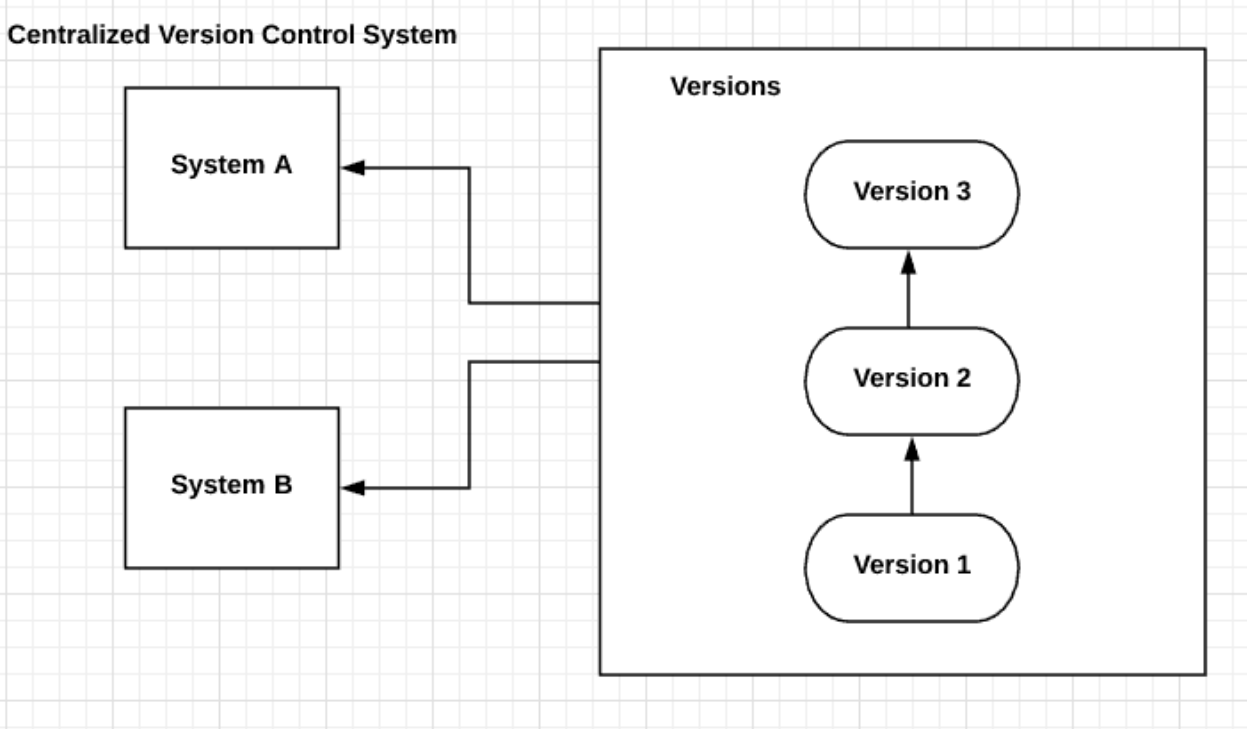
4. How many types of version control systems are there?

1. Local Version Control System:



Local version control system maintains track of files within the local system. This approach is very common and simple. This type is also error prone which means the chances of accidentally writing to the wrong file is higher.

2. Centralized Version Control Systems



In this approach, all the changes in the files are tracked under the centralized server. The centralized server includes all the information of versioned files, and list of clients that check out files from that central place.

3. Distributed Version Control System:

Distributed version control systems come into picture to overcome the drawback of centralized version control system. The clients completely clone the repository including its full history. If any server dies, any of the client repositories can be copied on to the server which help restore the server.

Every clone is considered as a full backup of all the data.