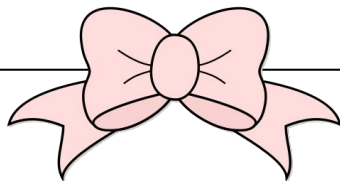




GirlScript Foundation
Education Outreach Program

Congratulations to all the Education Outreach Scholarship Winners



Welcome to the Introduction to Git & GitHub Course

Mentors for the course



Sukriti Shah

- Mentor and Web Developer at GirlScript Foundation
- Mentee at GSSoC'2020
- Microsoft Technology Associate (JavaScript & Python)
- Google Udacity Scholar - Web Development
- LinkedIn Profile - <https://www.linkedin.com/in/sukriti-shah>



Mentors for the course



Sakshi Grover

- Mentor at GirlScript Foundation
- Student Volunteer at Progate
- Campus Ninja
- Evangelist at Women Who Code, Mumbai Chapter
- Volunteer at IEEE SWAG
- LinkedIn Profile -

<https://www.linkedin.com/in/sakshi-grover-7300a9188/>



Mentors for the course

Kanjal Dalal



- **Mentor at GirlScript Foundation**
- **Software Engineer Intern, Lead Teaching Assistant and Teaching Assistant at Crio.Do**
- **Mentor at JGEC Winter of Code**
- **LinkedIn Profile -**

<https://www.linkedin.com/in/lonewolfkj/>



Overview of the Course

Day 1 - Version Control Systems (Sukriti Shah)

Day 2 - Intro. To Git and GitHub (Kanjali Dalal)

Day 3 - Installation and Setup (Sakshi Grover)

Day 4 - Creating Repository (Sukriti Shah)

Day 5 - Making Changes in Repository (Sukriti Shah)

Day 6 - git log, tagging and comparing commits (Sakshi Grover)

Day 7 to 9 - Parallel Development in a repository (Sukriti Shah)

Day 10 & 11 - Forking a repository & making a PR (Sakshi Grover)

Day 12 - Aliases, Common mistakes and Good practices (Sukriti Shah)



GirlScript Foundation
Education Outreach Program

Introduction to Git & GitHub

DAY 1

PRESENTED BY - SUKRITI SHAH

Agenda for the day –

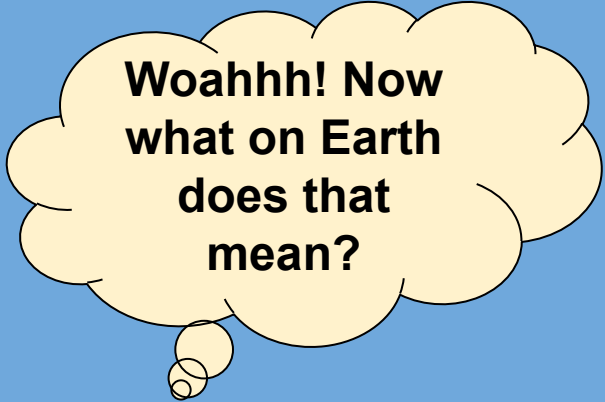
1. What is Version Control System(VCS)?
2. Why do we need VCS?
3. Different VC systems
4. Distributed and Centralized VCS




What is Version Control System (VCS)?

Version Control is the management of changes to documents, computer programs, large websites and other collections of information. These changes are usually termed as versions.

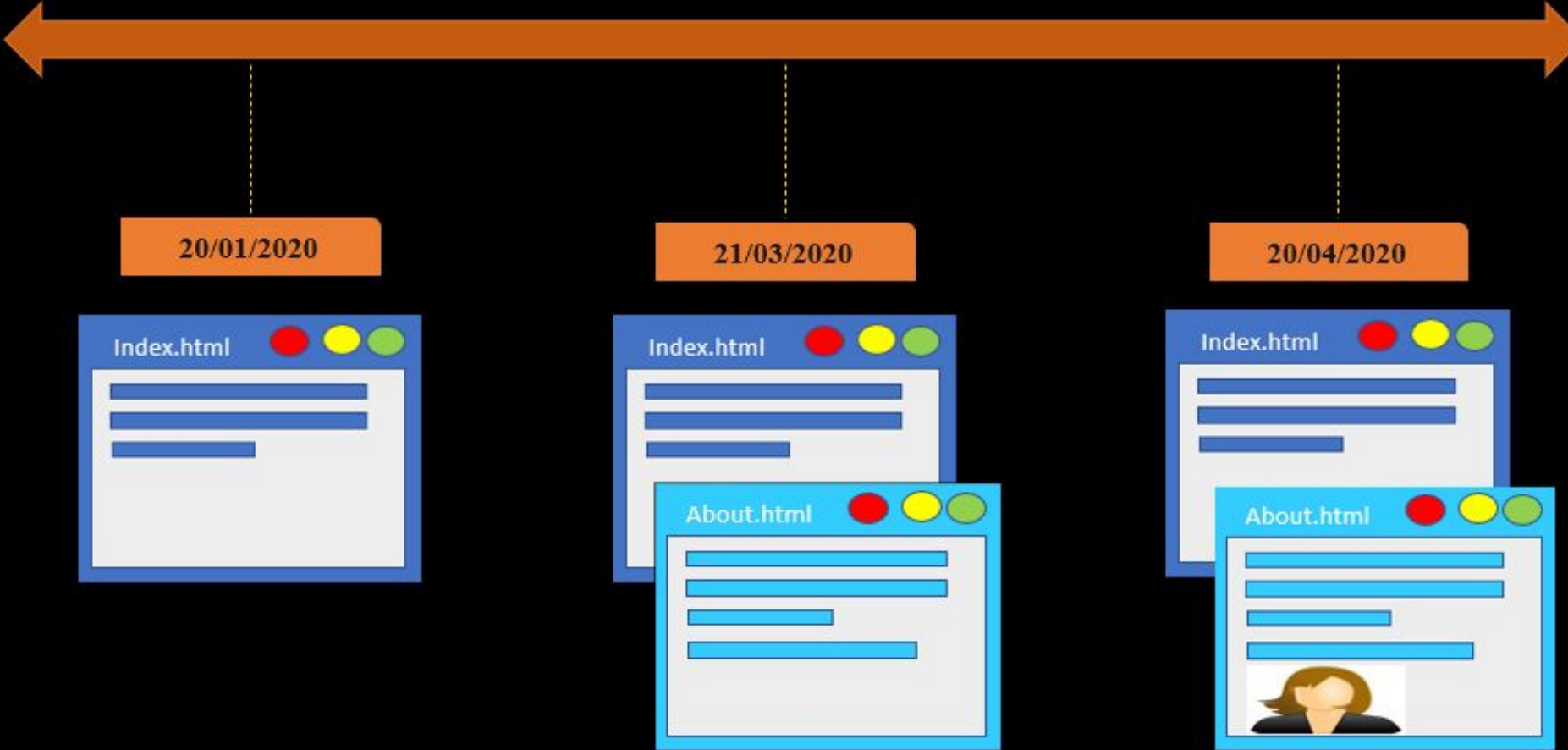
Changes can be of any kind like adding some new file to the project or modifying some existing file in the project.



**Woahhh! Now
what on Earth
does that
mean?**



**Hang in
there... Let's
move to the
next slide...**



A horizontal timeline with a large orange double-headed arrow at the top. Three vertical dashed lines mark specific dates: 20/01/2020, 21/03/2020, and 20/04/2020. Below each date is a screenshot of a web browser window. The first window (20/01/2020) shows 'Index.html' with three blue bars. The second window (21/03/2020) shows 'Index.html' with three blue bars and a new 'About.html' window with four blue bars. The third window (20/04/2020) shows 'Index.html' with three blue bars and an 'About.html' window with four blue bars and a profile picture of a woman.

20/01/2020

21/03/2020

20/04/2020

Index.html

Index.html

Index.html

About.html

About.html



1. Collaboration



In absence of Collaboration-

- Repetitive work?
- Different approach towards the same goal, hence inability to merge?
- Miscommunication?
- Conflicts?
- Chaos?



Collaboration
is important.
Agree?

2. Storing Versions & Backup

What if?

- I lose all my project files?
- I do not like this change in my project later?

I want to keep this version of my project, but I want to try few more designs. Should I maintain a separate copy each time? Should I just save the changed file?



3. Analysis



When did I make this
change?
How does this change
affect my project?
Who made this change?

As per documentation on Git's official website

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

What is “version control”, and why should you care? Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.

If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use. It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also generally means that if you screw things up or lose files, you can easily recover. In addition, you get all this for very little overhead.

Different Version Control Systems



**Apache
Subversion**

Git

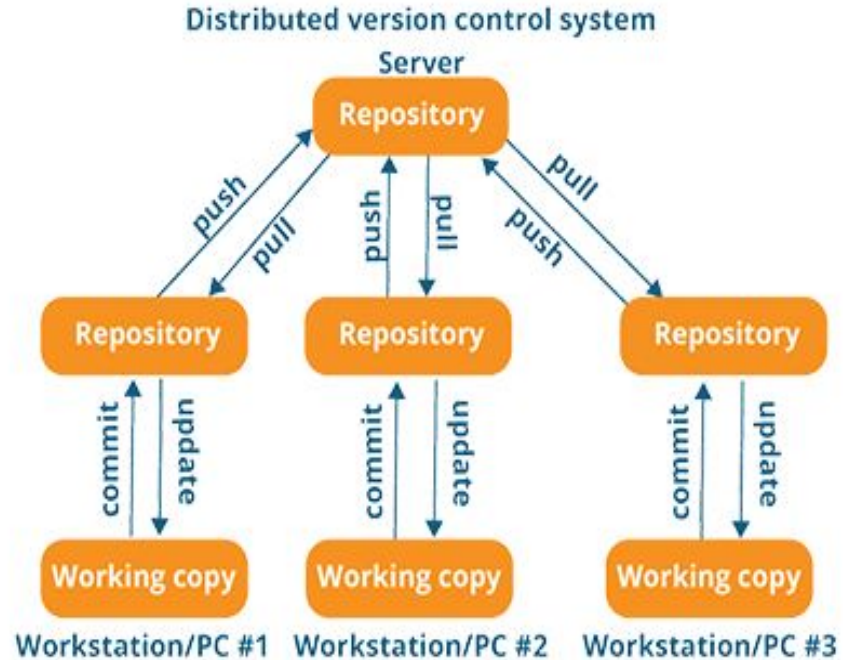
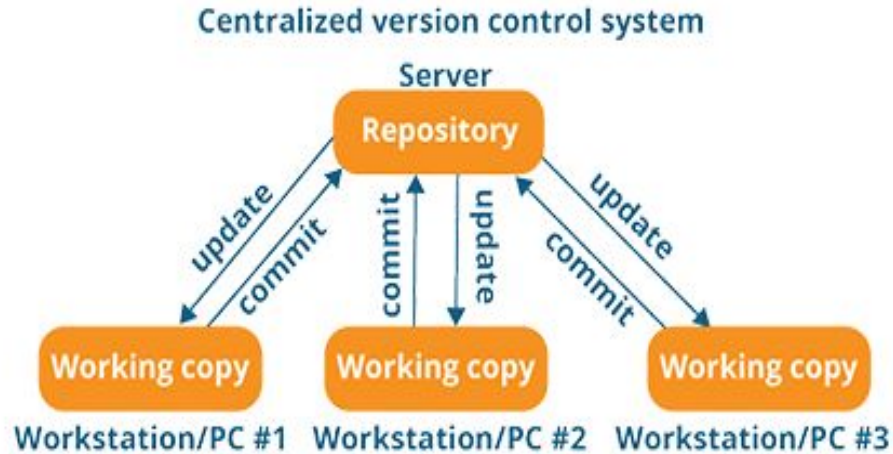


**Concurrent
Version
System**

Mercurial



Centralized VCS v/s Distributed VCS



Centralized Version Control Systems (<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>)

People need to collaborate with developers on other systems. To deal with this problem, Centralized Version Control Systems (CVCSs) were developed. These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place.

Advantages -

- Everyone knows to a certain degree what everyone else on the project is doing.
- Administrators have fine-grained control over who can do what, and it's far easier to administer a CVCS than it is to deal with local databases on every client.

Disadvantages -

- The most obvious is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on.
- If the hard disk the central database is on becomes corrupted, and proper backups haven't been kept, you lose absolutely everything — the entire history of the project except whatever single snapshots people happen to have on their local machines.

Distributed Version Control Systems

(<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>)

In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

Furthermore, many of these systems deal pretty well with having several remote repositories they can work with, so you can collaborate with different groups of people in different ways simultaneously within the same project. This allows you to set up several types of workflows that aren't possible in centralized systems, such as hierarchical models.