**Version Control(Git)**

**Version Control**is a system that records or keeps track of the changes to a file or set of files over time so that you can recall or go back to specific versions later.

The next video will introduce you to Git. You will get to know how and why is Git so powerful and why is it preferred over other version control tools like Mercurial.

Q1. Subversion is a version control system based on the

Ans: Centralised version control system

**✓ Correct**

**Feedback:**

Subversion is a version control system based on the centralised version Control System. In Subversion, there is only one repository filesystem, and it is completely dependent on the availability of a single server. Hence, it falls into the category of centralised filesystem.

Q2. What are the disadvantages of Subversion?

Ans: All

**✓ Correct**

**Feedback:**

In Subversion, there is only one repository filesystem, and it is completely dependent on the functionality of a single server. In Subversion, you cannot do any work on your project without a network connection; if anything happens to the central filesystem , you may lose your data and the history of the changes. These are all the disadvantages of Subversion.

**Version Control**is a system that records or keeps track of the changes to a file or set of files over time so that you can recall or go back to specific versions later.

The next video will introduce you to Git. You will get to know how and why is Git so powerful and why is it preferred over other version control tools like Mercurial.

Q3. What are the advantages of Git over Mercurial?

Ans: All

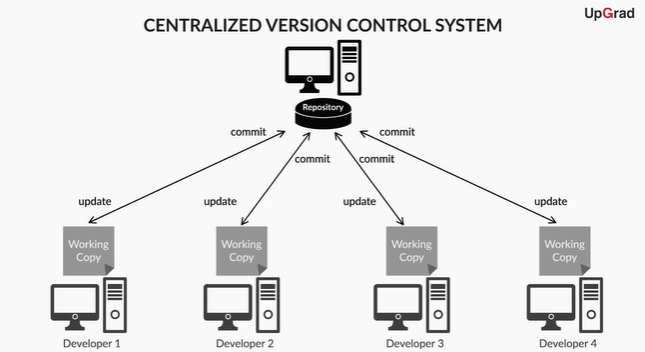
**✓ Correct**

**Feedback:**

Git is faster than Mercurial for network operations such as downloading and uploading project files to the file server. Git is used very widely by developer’s all across the globe and Git is more powerful than Mercurial for all small/big projects.

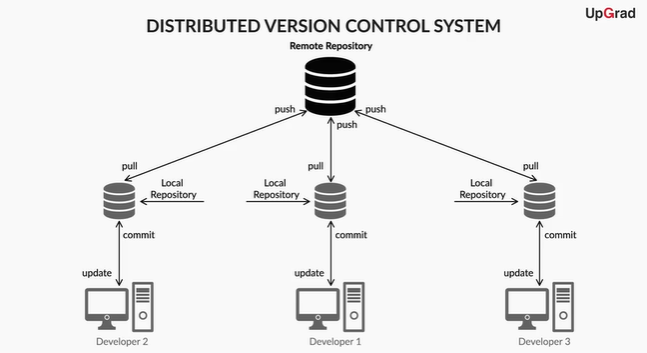
## Summary:

* Version control system is the management of changes to a collection of information such as documents, project code, etc.
* There are two major types of version control models, namely:
  1. Centralised
  2. Distributed
* The centralised version control system is working as a client-server model. Here, we have one centralised server and a localised repository filesystem that is accessible by a number of clients.



CVCS

* In the distributed version control system model, all developers have their own local file system, and changes between file system are implemented locally on their machines.



DVCS

* You also learnt why we use Git over all the other distributed version control systems.

#### Q4: Version Control

How does version control help us?

Ans: All

**✓ Correct**

**Feedback:**

Version control system helps developers to easily share the project code with their co-workers. Version control system can track collaborative changes to a project. It also allows developers to view all the past versions of the code and the difference between them.

#### Q5. Distributed Version Control

Which of the following is an example of a distributed version control system?

Ans: Git and Mercurial

**✓ Correct**

**Feedback:**

Git and Mercurial are both examples of distributed version control system because in Git and Mercurial, all developers have their own local set of files, and changes between files are also saved locally on their machines.

#### Q6: Version Control

Which version control system takes a peer-to-peer approach to version control?

Ans: Distributed version control system

**✓ Correct**

**Feedback:**

Distributed version control system takes a peer-to-peer approach to version control, which means that it synchronizes repositories by transferring changes from peer-to-peer. Here, all developers (i.e. peers) have their own local set of files, and changes between files are also saved locally on their machines.

But centralized systems use a client-server approach, which means that there is one repository filesystem, usually stored on a network server. Whenever you want to work on your project, you can connect the central collection of files through a network connection.

#### Q7: Git

The ***git --version*** command will show you the version of Git installed on your system.

Ans: True

**✓ Correct**

**Feedback:**

The ***git--version*** command will show the version of Git installed on your system.

#### Q8: Git and GitHub

How are Git and GitHub different?

Ans: Git is a distributed version control system and a tool to manage your project source code history. Git will help you keep track of the different versions of your project, but it tracks those changes locally on your computer; only you can see your project code and all the changes and revisions made to your project code.

GitHub, on the other hand, is a web-based, Git file hosting service that enables you to upload your project code, along with its changes and revisions, so you can showcase/share your projects and files with others.

#### Q9: Repository in Git

A repository is a directory where you can store and track all your project work, code, docs, files, etc.

Ans: True

**✓ Correct**

**Feedback:**

A repository is a storage space where all your files, their revision history, and anything related to the project can be stored. This can be either local to your system or in some storage space on an online host.

#### Q10: Files in Git

Which are the three states that files in Git can be in?

Ans: Files in GitHub can be in any of the following three states:

* Modified: In this state, modifications are made to a file or files, and the changes are still on our local system.
* Staged: In this stage, the changed files are added to the the staging area, which means the files will now become a part of your development history.
* Committed: In this stage, we make a record or take a snapshot of the files we have added to the staging area or our development history.

#### Q11: Commit in Git

Which of the following correctly describes the commit?

Ans: After staging your files, you can take a snapshot of your changes, and git will remember the changes. This snapshot is known as a commit.

**✓ Correct**

**Feedback:**

After staging your files, you can take a snapshot of your changes, and git will remember the changes. This snapshot is known as a commit. Commit is essentially a record of your progress in the project in the form of snapshots. Once you are done with the project, you can go back and have a look at these snapshots to revisit your progress during the project.

Let’s summarise the learnings from this video:

* You learnt that using GitHub, you can share your file system like files, documents, etc. with others, access another user’s file system, and store remote files and projects of other developers on your local system.
* You learnt about the differences between Git and GitHub. **Git** is a **distributed version control system**. It is a tool to manage your project source code history. **GitHub** is a **web-based Git file hosting service** which enables us to showcase or share our projects and files to others.
* You learnt a new term called Repository. A repository is a directory that contains your project work. All the files in the repository can then be uploaded to GitHub and shared with other people either publicly or privately.
* You also learnt about:
  + The three steps your files may be going through internally. These steps are:

|  |
| --- |
| 1. Modified |
| 2. Staged |
| 3. Committed |

**A glimpse of the next segment:**

In the next segment, you will learn how you can make your local work shareable with other people. In technical terms, you will learn how to link your local repository with a remote repository on GitHub. Confused? Don’t worry, the next segment will answer all your questions.

 Before moving on to the next segment, answer the questions given below to solidify your understanding of the concepts explained in the video and content of this segment.

#### Q11: Stages of a file in Git

Which of the following orders represent the stages that files go through before its changes can be officially tracked by Git?

1. Files are committed
2. Files are modified
3. Files are staged

Ans: 2, 3, 1

**✓ Correct**

**Feedback:**

In the first stage, the files are modified. After that, the files are added to the staging area where they can be reviewed. Finally, once you have reviewed the changes in the staged files, you can create a record of your changes and Git will remember them. This is known as making a commit. This commit will now become a permanent part of your development history and will be tracked by Git.

Q12: How can different commits in Git be uniquely identified?

Ans: Each commit is identified in Git by a unique ID, also known as SHA. Every time a commit is made in Git, that commit is assigned a unique ID or SHA. Using this unique ID, one commit can be distinguished from other commits.

#### Q12 Commit Messages

What is the significance of **-m**in ***git commit -m*** command?

Ans: The -m lets you add a message to the commit which will be made/saved. The commit message can also be used to remind you of the changes you had made with a specific commit. Also, it is important to write good commit messages when you are working in a team, as commit messages would help other members of your team in understanding what files you have created or what changes you have made to them.

#### Q13: git commit command

After running the command ‘git commit’, will your file get pushed to the remote repository on GitHub?

Ans: No

**✓ Correct**

**Feedback:**

The git commit only takes a snapshot of our changes, and the changes remain on our local machines. Further, the changes are not pushed onto our remote repositories on GitHub at the time.

#### Q14: Pushing changes

Now you know that a git commit command will not upload your changes to your remote repository. Which of the following commands will push your changes from your local system to your remote repository on GitHub?

Ans: git push origin master

**✓ Correct**

**Feedback:**

The ‘git remote add origin url ’ will only link our local repository to the remote repository on GitHub for the very first time. However, ‘git push’ will push all our changes from our local repository to the remote repository on GitHub.

Here are your takeaways:

* **First-time Git setup:**
  + For the first time Git configuration, you use the following commands:
    - git config --global user.name "random”
      * Using this, you will enter your GitHub username
    - git config --global user.email “random[*@example.com*](mailto:johndoe@example.com)”
      * Using this, you will enter your GitHub email
* **Making a commit and pushing your changes to GitHub:**
  + Commands to be executed are:
    - **git init**
    - **git add filename**
    - **git commit -m “commit message”**
    - **git remote add origin <url of the remote repository>**
    - **git push -u origin master**
* **Some important Git commands that are used very frequently:**
  + **git status**: This command will display the state of the working directory and the staging area. In other words, it lets you see the changes that have been staged and the changes that have not been added to the staging area.
  + **git log**: This command shows you the commit details. It lists out the commits made in the repository in reverse-chronological order, that is, the most recent commits show up first. It shows commits with the following details:
    - The commit ID or SHA
    - Author’s name (who made the commit)
    - Date and time
    - Commit message

**More on the ‘git remote add origin url’ command**

* Using this command, you can add a new remote repository to your local repository. To do so, you should use the **git remote add**command on the terminal, in the directory your repository is stored at. **The git remote add** command takes two arguments:
  + A remote URL, for example, <https://github.com/user/repo.git> (the address of the repository on your GitHub account that you want to link your local repository to.)
  + A remote name, for example, origin (it can be any name)

**A glimpse of the next video:**

In the next video, you will execute all the commands from the command line. In the upcoming videos, the professor will execute step 6, which is—making modifications to your files and uploading those changes to GitHub.

## Making your project remote part -2

So, you have the tools, which in this case are the commands in your hand. In the following video, the professor will show you how to execute the Git commands from the command line. You will also learn to make modifications to your files and add them again to GitHub.

All set? Brace yourself because now you are going to get the real feel of how developers across the globe work.

Summary: Some important points to keep in mind:

When you are pushing your changes after the first commit, i.e. after:

1. Initialising your Git repository and
2. Linking it with the remote repository on Git

You only need to run the following Git commands to commit any changes into your local Git        repository and then push those changes to the remote repository on GitHub:

1. **'git status'**: Used to check which files are changed and not yet moved to the staging area.
2. **'git add <filename>' or 'git add .'**: Used to add modified files to the staging area. You can add a specific file using the command 'git add <filename>'. If you wish to add all modified and unstaged files present in the workspace to the staging area, you can use 'git add .' command.
3. **git commit -m** “**New commit message**”: Gives a new commit message and commits all the files sitting in the staging area.
4. **git push -u origin master or git push**: Used to upload all the files and changes that were included in the most recent commit to your remote repository on GitHub.

#### Q15: The git log command

The command ***git log*** will show you your files in reverse-chronological order. In other words, the most recent commits will be displayed first.

Ans: True

**✓ Correct**

**Feedback:**

The command ***git log***will display a list of the commits you made, as well as its commit messages and commit IDs. In addition, ***git log*** will show the most recent commits first, i.e. commits will be shown in reverse-chronological order.

#### Q16: Commit Messages

Why do you think we need to write a commit message?

Note: Multiple options can be correct for this question

Ans: i)   
A commit message would help you and the other members of your team understand which files you have created

**✓ Correct**You missed this!

**Feedback:**

A commit message would help you and other members of your team understand which files you have created or what changes you have made to the files. But can you think of any additional uses of a commit message?

ii) The commit message can be used to remind you and the other members of your team of the changes you had made with a specific commit

**✓ Correct**You missed this!

**Feedback:**

The commit message can be used to remind you and the other team members of the changes you had made with a specific commit. But can you think of any additional uses of a commit message?

#### Q17: The git status command

What is the functionality of the command ***git status***?

Ans: All

**✓ Correct**

**Feedback:**

The ‘git status’ command will display the state of the working directory and the staging area. It lets you see the changes that have been staged or added to the development history. It also lets you see which changes have not been added to the staging area or the development history.

#### Q18: Comprehension 1

1. Create a folder on your desktop named ‘exercise’. This folder should not contain any files.

          (Note: If you are creating it in some other place and not on the desktop, just remember the directory where you have stored it to proceed with the following steps.)

1. Navigate to the folder named ‘exercise’ using the command line or terminal. Execute the command cd, such as ***cd Desktop/exercise/***to navigate.

(Note: If you have not created your ‘exercise’ folder on the desktop, you can type in your file’s address for the ‘exercise’ folder, depending on where you have created it.)

1. Now run the command ***git init***

Now, what do you see on the command line screen?

Ans: You should see a message being displayed, which says something similar to: “Initialized empty Git repository in /Users/<username>/Desktop/exercise/.git/”

Here, <username> will display the name of the user on your respective systems.

Note: The command***git init*** will initialize an empty Git repository for you. A repository is a directory for you to store project files that Git will track. Also, this command will create something called the **.git directory** in the same directory, in which your other files are present on your respective local systems; so, the directory created will have some default files.

#### Q19: Comprehension 1

Now, run the command ***git status*** and look at the output. What message do you see on your command line?

Ans: All of the above

**✓ Correct**

**Feedback:**

When you run the command ‘git status’ inside a folder that has no files inside it, you will see the following messages:

* On branch master
* No commits yet
* Nothing to commit

Note: The git status command will display the state of the working directory and the staging area. In other words, it lets you see the changes that have been staged and the changes that haven’t been added to the staging area.

#### Q20: Adding files to the staging area

Which command will add a file named ‘caladd.java’ to the staging area?

Ans: The command ***git add caladd.java*** will add a file named caladd.java to the staging area.

#### Q21: Finding commit’s ID

Which command will you use to find a commit’s ID in git?

Ans: Using the command ***git log***, you can get a commit’s ID in git.

* This command will show you the complete commit history for the project
* In addition, it will show us the commit ID for each commit listed in the commit history, the username and email ID of the person who made the changes, as well as the date and time when the commit was made.

#### Q21: git command

Which of the following commands can be used to initialise a new git repository?

Ans: git init

**✓ Correct**

**Feedback:**

The command git init is used to initialise a new Git repository

# Downloading Repository

Sometimes it is also required to download an online repository. The command used to accomplish this is **git clone**.

**Note**: This is just a basic introduction to this git command. You will get to learn about this and more such commands in more detail later in the program.

**git clone**

It is used to clone or copy another user's repository. When you join a new job you will most likely work on existing projects. In order to get the code of that project in your system, you will have to clone it from an existing GitHub repository.

Syntax:

**git clone <url of the remote repository>**

Example:

**git clone https://github.com/user/repo.git**

Q22: How can you get the URL of a GitHub repository?

1. Ans: Go to the main page of the repository on GitHub.
2. Click "Code" above the list of files.
3. Under "HTTPS" you will see the URL of the repository.
4. You can press the copy icon to copy the URL.

#### Q23: Comprehension

1. Go to the folder named ‘exercise’ which you created in the assessment of segment 5. You can do so using the cd command. eg. cd Desktop/exercise/
2. Create a new file inside it named ‘file1’ by typing the command **touch file1**in the command line or terminal and press Enter
3. Run the command **git status**and tell what you see on the command line? (You can copy-paste your answer.)

Ans: You should see the following on your command line or terminal:

* On branch master
* No commits yet
* Untracked files:

           (use "git add <file>..." to include in what will be committed)

file1

* Nothing added to commit but untracked files are present (use ‘git add’ to track)

Here, ‘Untracked files’ tell us that file1 is a new file or has certain modifications and these modifications have not been added to the staging area and are not being tracked by git currently.

Note: The git status command displays the state of the working directory and the staging area. In other words, it lets you see the changes that have been staged and the changes that haven’t been added to the staging area.

#### Q24: Comprehension

1. Now, run the command **git add file1** to add the file to the staging area
2. Run the command ***git status***
3. What do you see on the command line?

Ans: On following the above steps, you should see the following messages on your command line:

* On branch master
* No commits yet
* Changes to be committed:

           (use "git rm --cached <file>..." to unstage)

new file:   file1

Here, using the command ‘git add file1’, you have added the file to the staging area, where it will be tracked by git. Next, git is telling you that there have been ‘no commits yet’ and the changes should next be committed onto git’s memory. In other words, you should take a snapshot of its changes and commit the changes made to file1 to git’s memory and add those changes to the development history. Alternatively, if you have staged the changes by mistake and want to unstage the changes, that is, remove it from the development history, you can use the command ‘git rm --cached <file>’.

Note:

**git add**: This command will add your file with a name, e.g. file1, to the staging area, that is, your file will now become a part of your development history.

**git status**: This command will display the state of the working directory and the staging area. In other words, it lets you see the changes that have been staged and the changes that haven’t been added to the staging area.

#### Q26: Comprehension

1) Next, run the command: ***git commit -m “my first commit”***and

2) Run the ***git status***command again

3) What message do you see on your command line? (You can copy and paste your answer.)

Ans: On following the steps above, you should see the following messages on your command line:

* On branch master
* Nothing to commit
* Working tree clean

**git commit -m** “**my first commit**”: Git commit helps save files and its changes to git’s memory. Specifically, this means that all the  files in your staging area and its associated changes will be added to your development history. In other words, every time you make a commit to a git repository, git will take a snapshot of all the files added to the staging area (as well as the files’ changes) and save that snapshot to its memory.

* The command ‘-m’ is used for adding a message to your commit

**git status**: This command will display the state of the working directory and the staging area. In other words, it lets you see the changes that have been staged and the changes that haven’t been added to the staging area.

**Working tree clean**: This means that no modifications or changes have been made to the files that are being tracked by git