

Trainer Guide

Version Control with Git and GitHub

For Aptech Centre Use Only

Version Control with Git and GitHub

Trainer's Guide

© 2022 Aptech Limited

All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.

APTECH LIMITED

Contact E-mail: ov-support@onlinevarsity.com

First Edition - 2022



Preface

The book 'Trainer Guide Version Control with Git and GitHub' introduces the Git version control system. The book begins with describing version control systems and their purpose and then elaborates on Git in detail. The book covers various commands and operations in Git and also includes vital information about GitHub.

The faculty/trainer should teach the concepts in the theory class using the slides. This Trainer's Guide will provide guidance on the flow of the session and also provide tips and additional examples wherever necessary. The trainer can ask questions to make the session interactive and also to test the understanding of the students.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 Certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

Design Team

Sessions

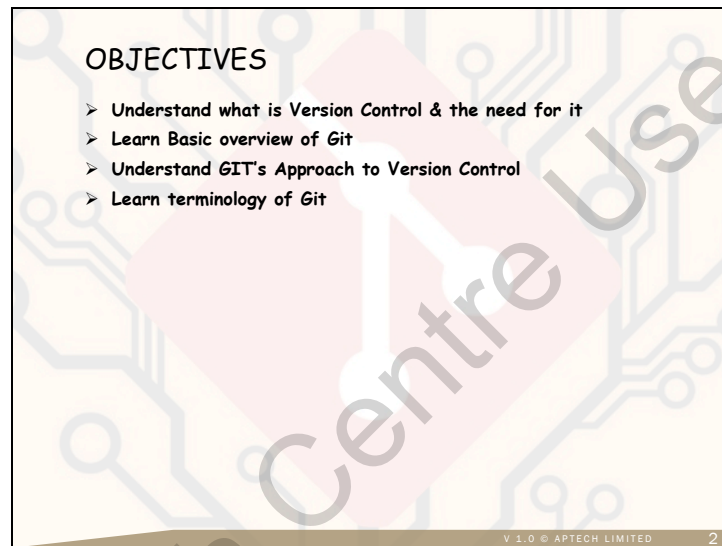
1. Getting Started with Git
2. Working with Git
3. Git Branching
4. Different Approaches to Git
5. GitHub as a Project Repository
6. More on GitHub and Overview of GitHub Pages

Session 1: Getting Started with Git

1.1 Pre-Class Activities

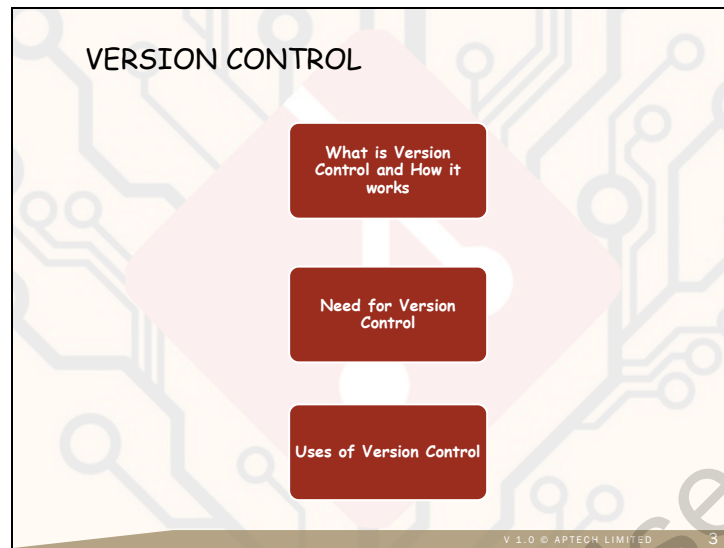
Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

Slide 2



Using the slide 2, give the students an overview about the objectives of the current session. Read out the objectives which are mentioned.

Slide 3



Using slide 3, tell the students that the necessity for version control will be understood in three different stages. During the first stage, discuss what version control is.

Explain to students about how a majority of programming projects end up generating a large number of files. To maintain these large number of files, a programmer would often have to spend a lot of time and efforts. There are also many risks which are associated with this procedure. Some of those risks include saving the wrong edit in a file and losing important changes in a file. Saving these files can also become chaotic if a group of people collaborate on the same project.

After this, tell the students how version control works.

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 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, and compare changes over time. You can see who last modified something that might be causing a problem, who introduced an issue and when, and more.

Tell students that to avoid all those issues which are mentioned, version control systems are used. Version control can simply be looked on as a type of systems which simply records all kinds of changes to a file. These version control systems maintain records of all the changes that are made to a file over its lifetime. With the help of version control, any version of the file can be viewed and used by the programmer. This can also help the programmer in recalling the versions of the file at a later date. In the next stage, discuss the uses of version control.

Tell the students primary benefits you should expect from version control are as follows:

A complete long-term change history of every file. This means every change made by many individuals over the years. Changes include creation and deletion of files as well as edits to their contents.

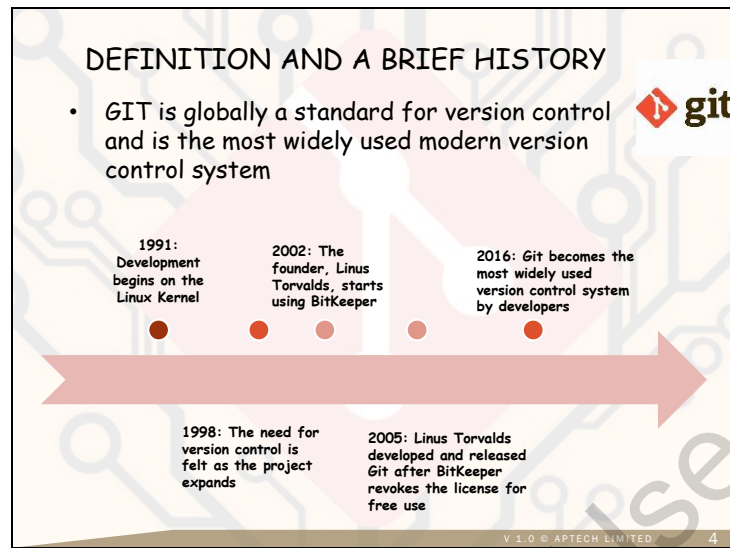
Branching and merging. Having team members work concurrently is a no-brainer, but even individuals working on their own can benefit from the ability to work on independent streams of changes. Creating a 'branch' in VCS tools keeps multiple streams of work independent from each other while also providing the facility to merge that work back together, enabling developers to verify that the changes on each branch do not conflict.

Traceability. Being able to trace each change made to the software and connect it to project management and bug tracking software and being able to annotate each change with a message describing the purpose and intent of the change can help not only with root cause analysis and other forensics.

Also, before moving to the next slide, ask the students if they have any doubt regarding the topics that have just been explained.

Use this link to read more - <https://betterexplained.com/articles/a-visual-guide-to-version-control/>

Slide 4



Using slide 4, tell the students that Git is currently the most popular version control system. Git is the standard for version control systems and is also used in a number of different domains for a wide range of applications. Git is an open-sourced software which is also distributed freely under the terms of the GNU General Public License version 2.

Tell the students that Linus Torvalds who is the creator of Git is also the creator of the Linux Kernel. Tell the students that the history of Git begins in 1991 when Linus Torvalds began the development of the Linux Kernel. The project of Linux Kernel eventually expanded and the necessity for version control was felt in 1998. In 2002, Linus Torvalds started using BitKeeper but eventually, BitKeeper revoked its license for free use. Eventually, Linus Torvalds started to develop Git. He released Git in the same year. Tell the students that eventually in 2016 Git became the most widely used version control system. Also, tell the students that since 2005 the current maintainer of Git is Junio Hamano. Then, tell the students that the last stable release of Git was on 15th December 2018.

Refer to following links for information:

<https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>

<https://medium.com/@willhayjr/the-architecture-and-history-of-git-a-distributed-version-control-system-62b17dd37742>

Slides 5 to 9

BASICS OF GIT 1-5

Branching in Git

Approaches to Versions

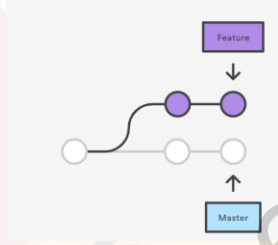
Focus on Locality and Integrity

States and Sections in Git

Git Repositories

Overview

- Most unique feature of GIT is the provision of Branching
- Git branches are cheap and easy to merge. This facilitates the feature branch workflow popular with many Git users
- **Advantages**
 - Context Switching is Easy
 - Role-based code lines are possible
 - Feature based workflows are possible



V 1.0 © APTECH LIMITED 5

BASICS OF GIT 2-5

Branching in Git

Approaches to Versions

Focus on Locality and Integrity

States and Sections in Git

Git Repositories

Overview

- Git's approach to versions is fundamentally different. It stores data as a sequence of snapshots of a set of files.
- If there is a file that has no changes, the file is not stored again.

V 1.0 © APTECH LIMITED 6

BASICS OF GIT 3-5

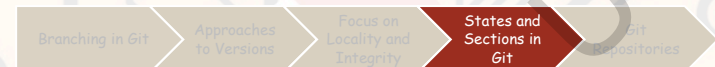


Overview

- Git is much faster than other version control systems as most operations require only the user's local files and resources to work. The dependence on a network is minimized.
- Git maintains integrity by using a checksum. Each entity stored is referred to by its checksum. Thus, there remains no way for a file or directory to be changed, without Git being notified of it.

V 1.0 © APTECH LIMITED

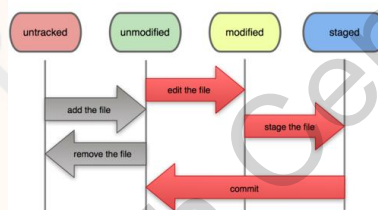
BASICS OF GIT 4-5



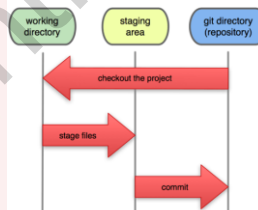
States - A user's work in Git iterates over three states

Sections - A Git project works in three main sections

File Status Lifecycle



Local Operations



V 1.0 © APTECH LIMITED

8

BASICS OF GIT 5-5



- Git repositories are not binary files, unlike most databases. Instead, a Git repository is a directory that contains a set of files and subdirectories within it

Standard GIT Repository

A standard Git repository is linked to a copy of the working tree.

Bare Repository

A bare repository is just a copy of the repository itself and is not linked to a copy of the working tree.

V 1.0 © APTECH LIMITED

9

Using slide 5, tell the students that there are five subtopics that will be covered under the topic of basics of Git. Each of the subtopics will be discussed in detail.

Branching in Git

Tell the students that Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something— no matter how big or small—they create a new branch. This ensures that the master branch always contains production-quality code.

Using feature branches is not only more reliable than directly editing production code, but it also provides organizational benefits. They let you represent development work at the same granularity as the agile backlog. For example, you might implement a policy where each Jira ticket is addressed in its own feature branch.

Explain to the students various advantages of Branching.

Context Switching is Easy - One can create a branch to explore an idea, commit changes to that branch, and go back to the original branch. This can be done without affecting integrity of data.

Role-based code lines are possible - One branch can be reserved for testing, while another can be used to try out uncertain approaches.

Feature based work flow are possible - Out of all the components of a project, different features can be worked on independently.

Refer to following link for more information:

<https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging>

Using slide 6, explain approaches to versioning. Tell students that the approach to versions of Git is fundamentally different from the approach that is taken by other version control systems. Tell the students that most of the other version control systems are delta-based in their approach. This means that those version control systems store data in the form of a list of changes that are made to a file. While Git stores the data in a sequence of snapshots of a set of files. These snapshots of the files also have a unique reference number. This makes Git a miniature file system in itself.

Refer to following link for more information:

<https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

Using slide 7, explain Focus on Locality and Integrity. Also, explain to students that Git focuses on both locality and integrity. The focus on locality is maintained by ensuring that most of the operations only require the local files and resources to work. This also makes the entire network faster. The focus on integrity is maintained by using a checksum. Every entity that is stored

contains a checksum. This way any kind of file loss or corruption is prevented by Git.

Refer to following link for more information:

<https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

Using slide 8, explain States and Sections in Git. Tell students that a user can work in three different states in Git. Those three main states are:

- Committed: This state means that the data has been stored in the database. Tell the students that this simply indicates that the entire data is safe
- Modified: This state means that some changes have been made to the file, but those changes have not been committed yet
- Staged: This state means that the file has been modified and it will go into the next commit's snapshot

Tell students that there are three main sections of project work in Git. Those three main sections are:

- Git directory: In this section, the object database and the metadata of the project are stored
- Working tree: This section represents a collection of files which originate from a certain version of the file system
- Staging area: This section represents the file in the Git directory which stores the information about what is going to go in the next commit

Now, ask the students if they have understood the difference between States and Sections.

Refer to following link for more information:

<https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

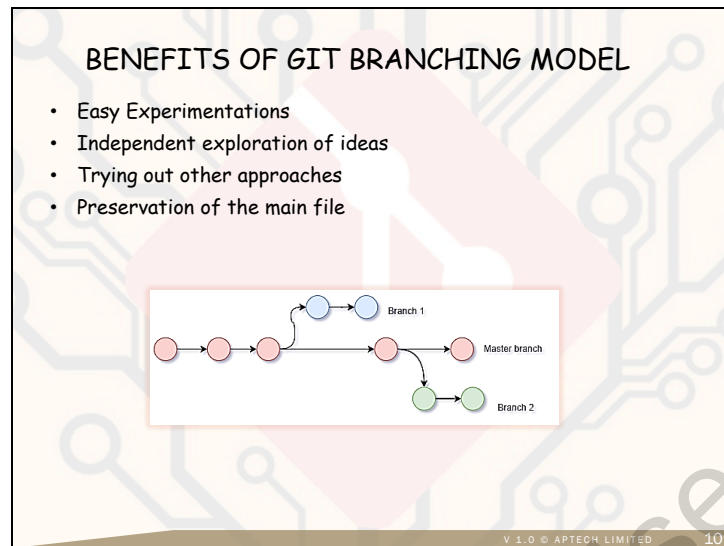
Using slide 9, discuss Git Repositories. Tell students that the Git repositories can roughly be treated as a type of folder. These repositories contain the project files in order of the directory hierarchy. This is why the Git repositories are also considered as the fundamental unit of many operations in Git. These repositories consist of a number of things including a commit object and a set of references to those commit objects. Tell the students, that there are two types of Git repositories. Those two types are:

- Bare repositories: These are intended to be used on a server so that programmers can share the changes that they make to a file
- Non-bare repositories: These repositories are common and are designed for users. It allows users to make the changes to the files and then, create the new versions of those files

Refer to following link for more information:

<https://www.sbf5.com/~cduan/technical/git/git-1.shtml>

Slide 10

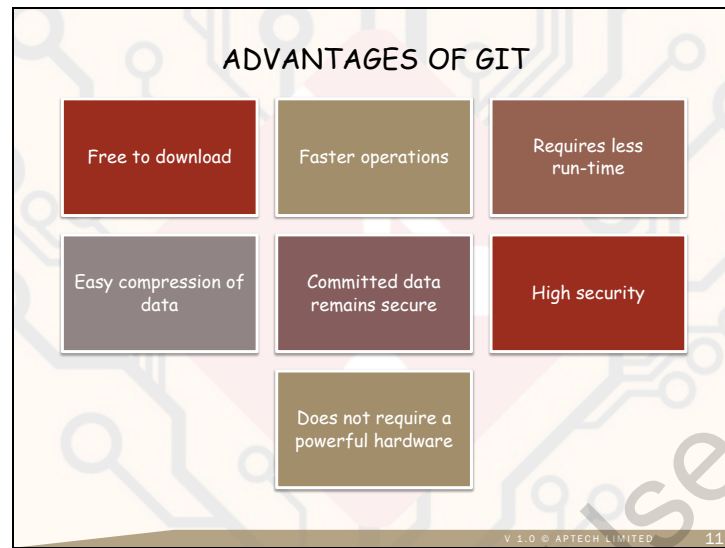


By using slide 10, tell the students that there are a number of benefits of using this branching model of Git, such as:

- A new branch can easily be created to try out an experiment or a new idea. If this new idea does not work then, it can easily be discarded without affecting the entire project
- All the features of the project can be worked on independently without affecting the other features
- Apart from the testing of the main approach, branches can also be used to try out the other approaches about which the developer might still be uncertain
- Various ideas can be tried out in the other branches without affecting the main version of the file or the document

Before proceeding to the next slide, recap everything that has been covered in the session until now. This will help the students learn better in the session.

Slide 11

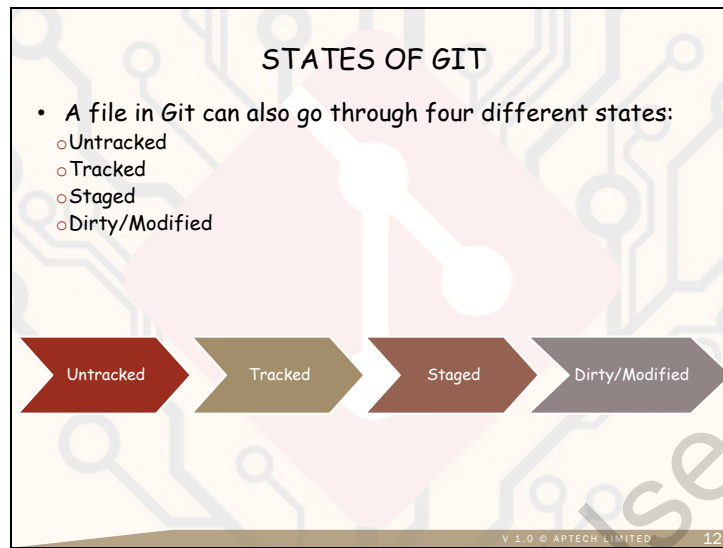


Using slide 11, proceed with explaining advantages of Git, such as:

- It is free to download over the Internet. Its source code is also publicly available and anybody can modify it too
- It is faster due to the locality of its operations
- It requires less run-time overheads since it is written in the C language
- It can easily compress data as most of the files in the local machine of the users are often quite small
- It is highly unlikely for a data to be lost once it is committed into Git
- Cryptographic hash and checksums are used which makes Git very secure
- Does not require an expensive or powerful hardware server

Refer to following link for more information: <https://www.toolsqa.com/git/what-is-git/>

Slide 12



Using slide 12, tell the students that in Git all files can also go through four different states. Those states are:

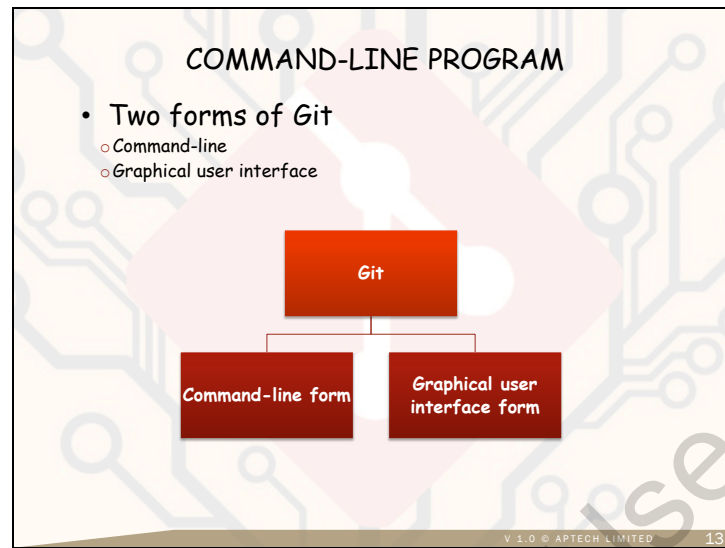
- **Untracked:** This state means that the file has never been committed or stages which means that the file is new to the entire Git project
- **Tracked:** This state means that the file has been committed, but it has not been staged
- **Staged:** This state means that the file has to be included in the commit that is coming. The file is also taken in the staging area. Tell the students that the staging area is also known as the index
- **Dirty/Modified:** This state means that the file has been modified but it has not been staged

Give an elaborate example to explain these states. Take the example of a hypothetical scenario involving a developer Paul who is using Git. Mention how Paul's file can undergo these four states.

Tell the students that through all this, it can be concluded that the workflow of any user in Git will be in three stages:

- Editing the file in the working directory
- Adding the files to a staging area
- Committing all the changes which are made

Slide 13



Using slide 13, tell the students that Git is present in both the command-line form and the graphical user interface form. Tell the students that both of these forms have their unique features. It is important for all students to have some basic information about the command-line form. Tell the students that the command line services can be accessed through the command-line interface which is present on the operating system. For example, the PowerShell or the command prompt can be used to access the Git command line interface in Windows. However, the terminal is used in Linux or Mac.

Tell the students that there are benefits of using each of these forms of Git. The benefits of using command-line form are:

- It lets you schedule Git commands automatically with the help of a small script
- Certain tasks can only be performed with the help of the command-line for. For example, tell the students that if a user wished to run certain operations then he or she would have no choice, but to use this form of Git
- It allows users to know and control the outcomes of their work precisely

There are also many benefits of using graphical user interface form. Those benefits are:

- It provides users with advanced options that save a lot of their time
- It allows you to simply drag-and-drop the codes which save the user from making any mistake while typing or coding
- It is easier to understand
- The graphical representation of collaborations is better

Ask the students the difference between two forms of GIT. Gather their responses. Then, give them the correct answer,

Ask the students to give an example of each form. Gather their responses.

Refer to following link for more information: <https://www.youtube.com/watch?v=HVsySz-h9r4>

Slide 14



Using slide 14, tell students that Git can be installed as both a command-line interface and its graphical user interface form. Tell the students that to setup Git the students must perform a few steps as follows:

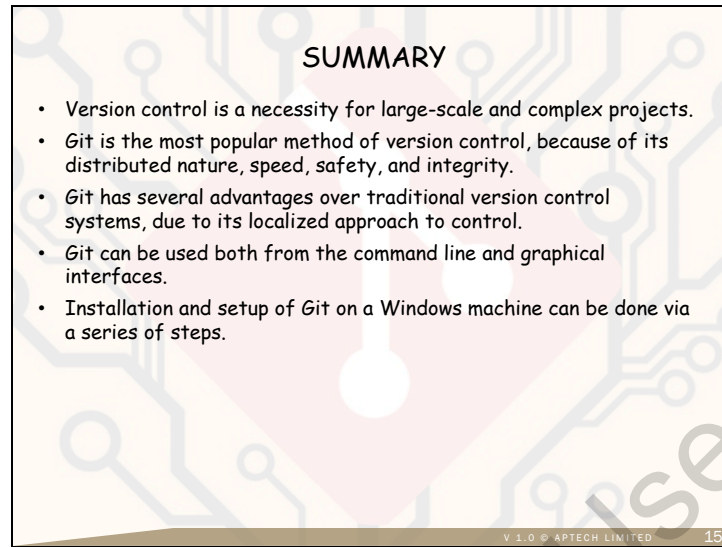
- Download the setup files from the official Web site of the Git project
- Open the installation and follow the prompt to finish the installation
- Once the installation is finished, then open the command prompt to check if Git has been installed. To check that, simply type 'git-version'
- After that, some configuration will still be required before a user can start using Git. To complete those configurations simply add the commands `git config --global user.name 'Your Name Here'` and `git config --global user.email 'Your E-mail ID Here'`

With this, the setup of Git is complete.

View this video to understand setting up GIT:

https://www.youtube.com/watch?v=J_Clau1bYco

Slide 15

The slide has a light orange background with a faint circuit pattern. A large, semi-transparent watermark 'For Aptech Centre Use Only' is oriented diagonally across the slide. The title 'SUMMARY' is centered at the top. Below it is a bulleted list of five points. At the bottom right, there is a small footer containing 'V 1.0 © APTECH LIMITED' and the slide number '15'.

SUMMARY

- Version control is a necessity for large-scale and complex projects.
- Git is the most popular method of version control, because of its distributed nature, speed, safety, and integrity.
- Git has several advantages over traditional version control systems, due to its localized approach to control.
- Git can be used both from the command line and graphical interfaces.
- Installation and setup of Git on a Windows machine can be done via a series of steps.

V 1.0 © APTECH LIMITED 15

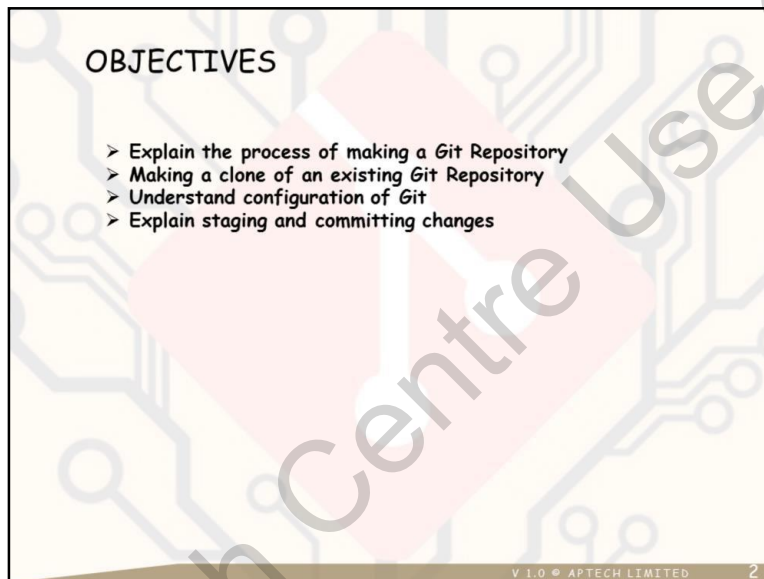
Using slide 15, summarize the session.

Session 2: Working with Git

2.1 Pre-Class Activities

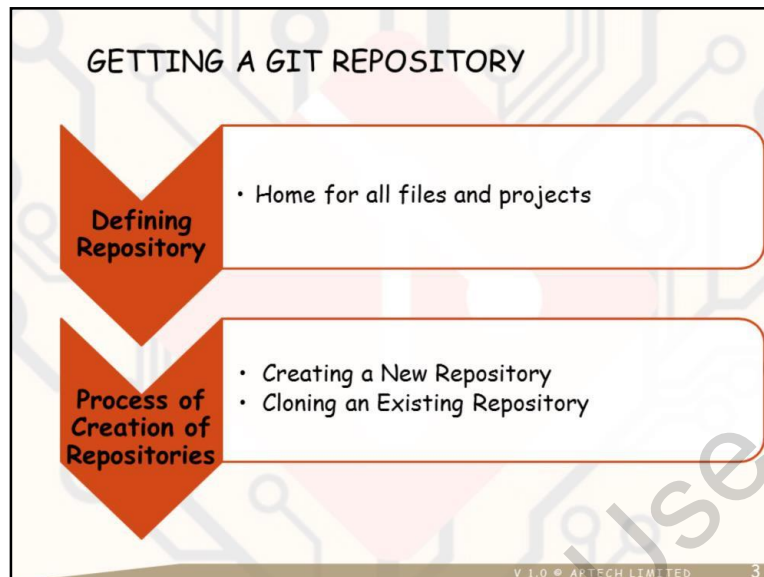
Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

Slide 2



Using the slide 2, give the students an overview about the objectives of the current session. Read out the objectives mentioned.

Slide 3



Using slide 3, tell the students that by now the students must know the steps to successfully setup and configure Git. Tell the students that with the help of this slide, they will know what a repository is and how a repository can be created in Git.

Ask them a question:

Q. Can you think of any real-world analogies for a repository?

Give the students time to think and come up with suitable answers.

Share an example:

Suppose you and a friend are working together on a project about movies. You will have to collect various photos, videos, and documents. Putting all of them together in a folder called 'Movie Project' will help you find them easily whenever required.

Defining Repository

Tell the students that the purpose of Git is to successfully manage files or projects as a number of changes occur in those projects. All the information about these changes is stored in data structures, which are known as repositories. Every repository in Git contains a number of objects.

Two of the most important objects that are present in a repository are:

1. A set of commit objects
2. Heads: These can be defined as the set of references to commit objects A Git repository

can contain all types of files.

Refer to the following link for more information:

<https://www.sbf5.com/~cduan/technical/git/git-1.shtml>

Process of Creation of Repositories Tell the students that there are two different approaches that an individual can follow to create a Git repository. Those two different approaches are:

1. By creating a new repository

To create a new repository with this method, students will have to use the **git init** command. Executing this command will create a new master branch and a new .git subdirectory in the current working directory.

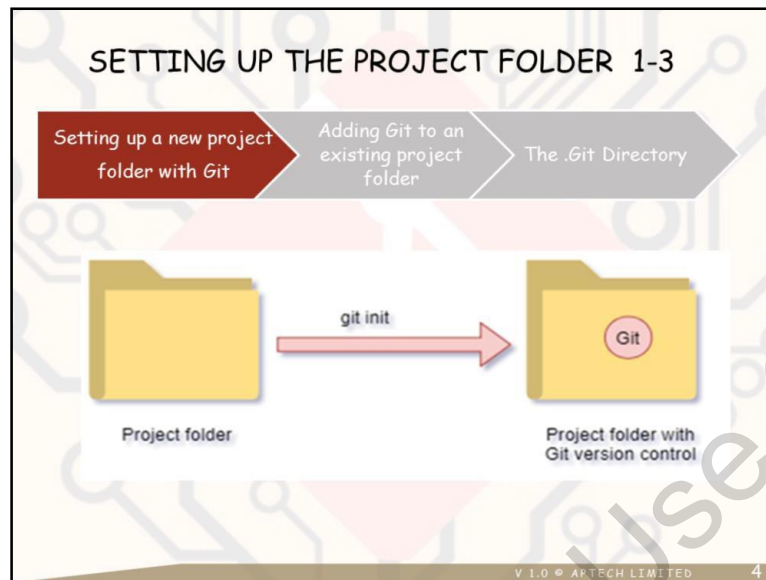
2. By cloning an existing repository

To use this method the students will have to use the **clone** command. This will allow the user to also obtain a local development clone.

Refer to the following link for more information:

<https://www.atlassian.com/git/tutorials/setting-up-a-repository>

Slide 4



Using slide 4, tell students about different ways a Git repository can be created while a project folder is also being setup.

Share an example:

Suppose, you have made a folder on your computer, and your friend has made a similar folder on her computer. The two of you now must share your files with each other. Suppose you send her a mail with your files and she sends you a mail with her files. Now you each have the complete set of files. However, this gets difficult to maintain when you are constantly adding new files. You would have to send multiple mails every day. Here, you can use Git. You can each add Git to your own project folders, and create a main project folder containing each of your files. The contents of the two folders on your two computers will be updated whenever either of you makes a change.

Tell the students that, further they will learn about:

1. Setting up a new project folder with Git
2. Adding Git to an existing project folder
3. The .git directory

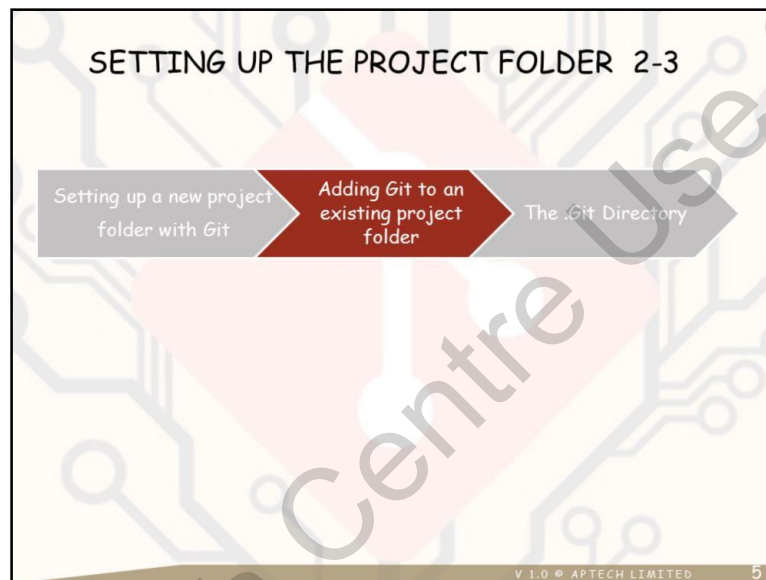
First, discuss how to set up a new project folder with Git.

It is extremely important for a developer to have a properly maintained project folder. This task is taken care of by the version control system in Git. In Git, a folder is also known as a directory. A user would also be required to add version control to a directory.

This task can be accomplished by following a few simple steps:

- Create a new folder and name it with the help of the command line or the GUI of the operating system
- Open the command prompt to navigate to the newly created file
- Execute the command of 'git init' to add the Git version control
- Users can now find a new subfolder named .git. This folder will contain all the files which will be necessary for the maintenance of the entire project

Slide 5

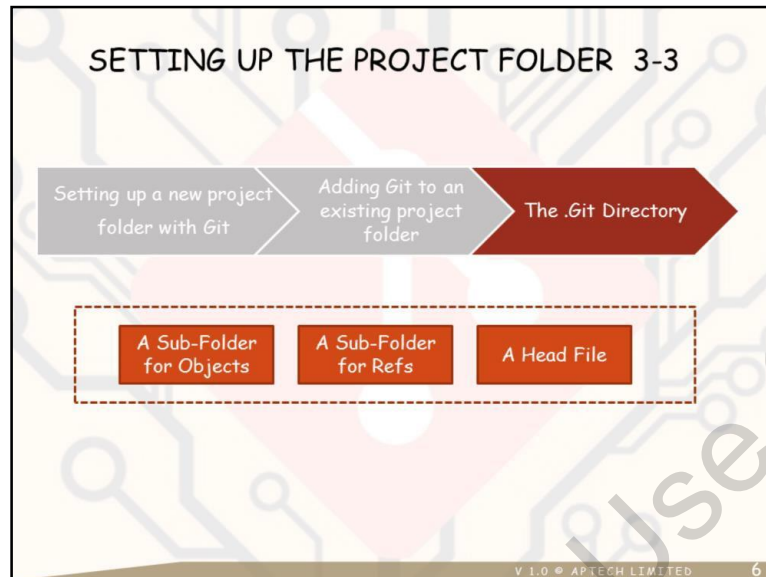


Using slide 5, tell students that now they will learn about adding Git to an existing project folder. Tell the students that sometimes a user might forget to add Git version control to a project. In such a case, Git can still be added even though a project folder might be created along with some of its files. There are a few steps that a user can follow to accomplish this task, as follows:

- Use the command prompt to navigate to the project folder
- Execute the command 'git init' to add git

You can also refer - <https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/>

Slide 6



Using slide 6, tell the students that a git repository is created every single time a user decides to execute the git init command. A git repository contains all kind of vital information that might be required for a project. Some of that information might include the information about remote repository address, commits, and many other types of information. The .git repository also contains a log. This log stores all the commit history that will allow users to roll back and check the history whenever it might be required.

Ask them the question:

Q. Can anyone simply open and change the contents of the .git folder?

A. The .git folder should not be changed unless the user is very familiar with Git. Changing the file incorrectly can damage the contents. That is why the folder is hidden.

Tell the students that three different types of content are stored in these repositories.

Those three different types of content are:

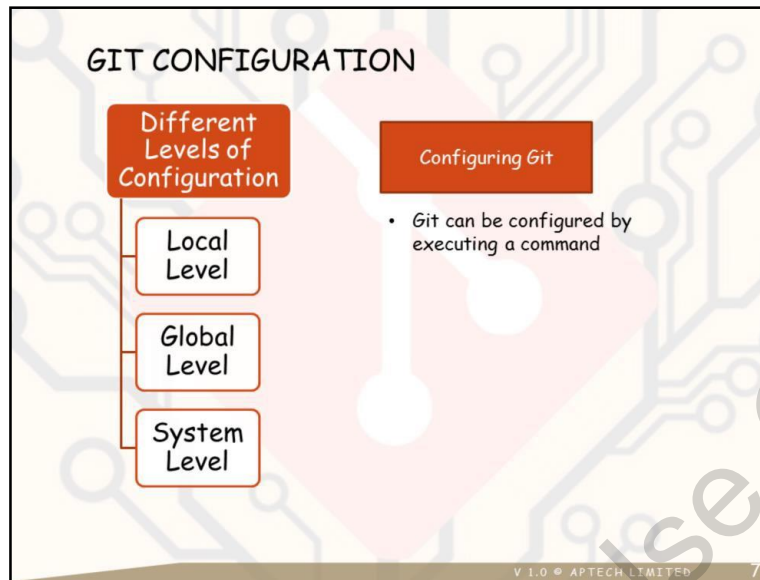
1. A sub-folder for objects
2. A sub-folder for refs
3. A head file

Tell the students that a head file holds all the tracking information for commits.

Refer to the following link for more information:

<https://stackoverflow.com/questions/29217859/what-is-the-git-folder>

Slide 7



Using slide 7, tell students that Git configuration can be defined as a set of possible actions performed to customize the Git version control system.

The most basic use case of Git configuration is when users can set up username and email as per the preference of the user.

Share an example:

Imagine that you live in a building that has many apartments. You may decide to change the lights in your own balcony. This would be a local level configuration. Imagine that you change the lights on the entire floor. This would be a global level configuration. Now, imagine that you change them for the entire building. This would be like changing the system level configuration.

Tell the students that similar to this example, there are also three different levels of configuration in Git. Those three different levels of configuration are:

- 1. Local Level:** If no configuration option is selected then, this option is selected by default. This level of configuration is often applied to different context repositories that get invoked in the Git config. The values of the local configuration are also stored in a file. This file can be found in the .git repository
- 2. Global Level:** This level of configuration is user-specific. The term user-specific refers to the fact that all configurations are applied to the operating system user. The values of global level configuration are all located in a file. This file can be located in the home directory of the user
- 3. System Level:** The system level configurations are applied throughout the entire machine. The

file of the system level configuration is present in a gitconfig file. This file can be found off the system root path

Tell the students that this is also the order of priority for the configuration levels.

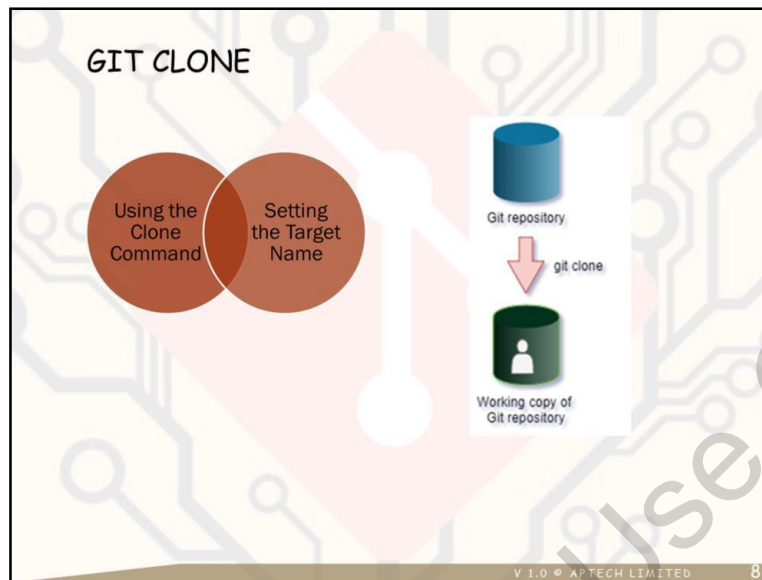
Tell the students that Git can be configured by executing a command: `git - [level] [option] [argument]`

Refer to the following link for more information:

<https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config>

For Aptech Centre Use Only

Slide 8



Using slide 8, tell the students that Git clone is a type of Git command line.

Ask them the question:

Q. What occurs when you clone something?

A To clone something means to create an exact copy of it. Hence, a copy is created.

Share an example:

Suppose you accidentally delete entire folder of your project from your computer. You will then must acquire a folder containing your work as well as your friend's. You can simply clone the folder from Git, without losing any of your contents.

Git clone is a type of Git command line. This command line can help a user in creating a clone of a pre-existing repository. This cloning procedure will help the user in creating working on a project without tampering with the main line of work. There are two different methods of creating a clone of a repository. These methods are:

1. By using the clone command

To use this method, a user would have to follow a few simple steps:

- Find the existing repository that has to be cloned. To accomplish this task, users can use the URL of the repository
- Execute the command 'git clone <url>'. Tell the students that <url> must be replaced with the requires URL
- The user can now find a .git directory

This method will help the user in creating a repository that is exactly similar to the pre-existing repository.

2. By setting the target name

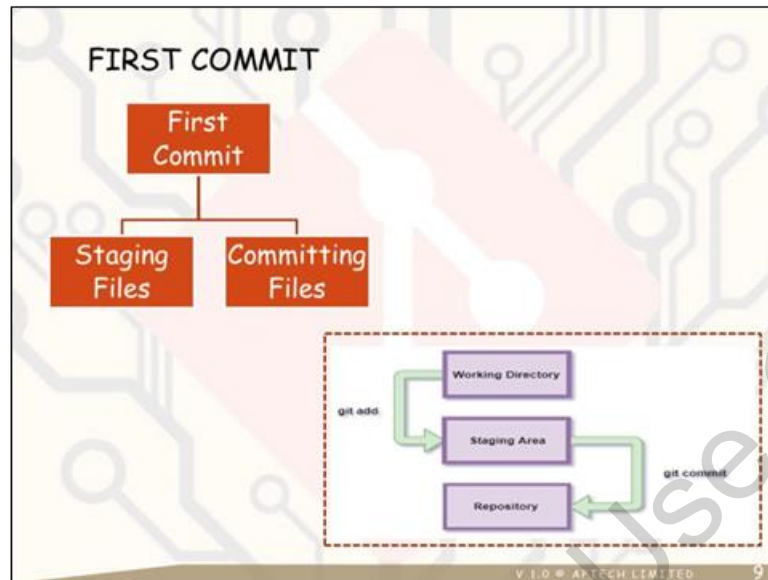
This method can help a user in cloning a repository with a different name and to use this method, users will be required to execute a command 'git clone <url><repo name>

To understand the application of what was just explained ask the students to imagine a scenario. In this scenario, tell the students that there is a developer named Jane D. Jane D is taking over the work of some other colleague who has just left the organization. Here, the work of the colleague is assigned to Jane D and Jane D does not wish to overwrite on the work of her colleague. To handle this case, the best solution for Jane D is to create a clone of the repository of the colleague. This would allow Jane D to work without any kind of a hassle and would further make sure that the work of the colleague is also not overwritten. Tell the students that this is one of those situations where the Git clone feature can be extremely useful.

Refer to the following links for more information:

<https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone> <https://git-scm.com/docs/git-clone>

Slide 9



Using the slide 9, tell the students that this slide will help the students in understanding how to commit changes to the project. Tell the student that a repository can be created by a user and eventually, the user will also start working on the files that might be present in that repository. There will also be some changes that will be made in those files. To preserve these changes, a user would must commit those changes in the files. This commit option can also be considered as a type of save option, which is also available in other software. There are two main steps, which must be followed if a commit must be made successfully. Those steps are:

1. Staging the files

Tell the students that before a change can be committed in git that change first has to be staged. To accomplish the procedure of staging a file command must be executed. That command is 'git add'. This command can be executed to stage a set of files, a single file, or even a sub-repository. This step allows the file or the files to be made ready for being committed.

2. Committing the files

To commit a file successfully to git a command is required to be executed. That command is 'git commit'. Once this command is executed the editor will pop up which will ask for the commit message. Tell the students that users can also choose to add the commit message with the commit command too in the format of 'git commit -m 'commit message''. This 'commit message' can be replaced by any other desired message.

Ask students a question:

Q. What is the difference between staging and committing files?

A. When a file is staged, you are telling Git to commit it whenever the next commit takes place.

Thus, several files can be staged, and they can get committed together.

Share an example:

Suppose your task is to collect information regarding Hindi movies. You may put several images in your folder. You can stage these immediately. You may then find a few videos and put them in your folder, staging them too. After deciding that you have collected all the information you require, you can commit your changes. The images as well as the videos will get committed. This saves you the time taken to commit it for every change.

Slide 10

SUMMARY

- Git provides commands to enable creating and cloning of a Git repository.
- It is easy to create and set up a Project folder with Git repository
- You can add Git to an existing folder
- Git configuration can be customized at various levels
- Git clone command enables you to create a copy of a repository
- The process of committing changes in Git is like saving changes

V 1.0 © APTECH LIMITED 10

Using slide 10, summarize the session.

Session 3: Git Branching

3.1 Pre-Class Activities

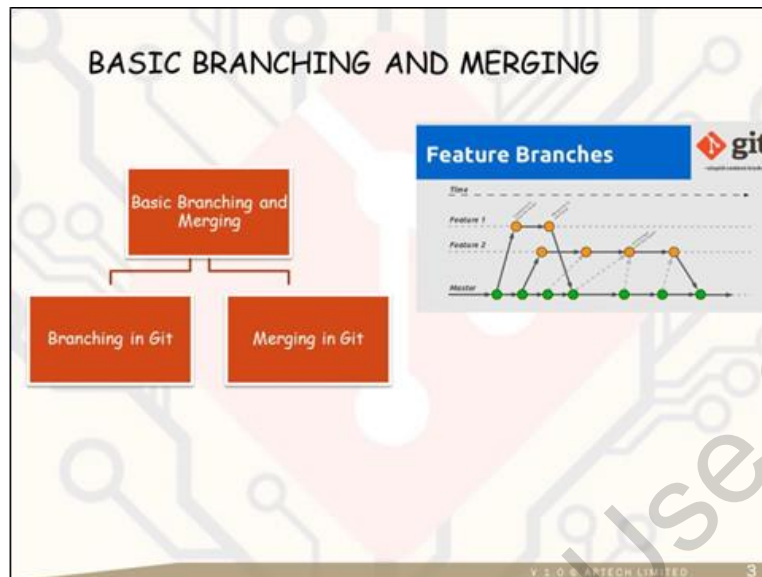
Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

Slide 2



Using the slide 2, give the students an overview about the objectives of the current session. Read out the objectives mentioned.

Slide 3



Using slide 3, explain to the students the concept of branching and merging.

Ask them the question:

Q. What is the difference between branching and merging?

A. Branching and merging are practically opposites. Branching consists of splitting one workflow into two, while merging consists of joining two workflows into one.

Branching

Branching lets you work on a new feature without affecting the main codebase.

Merging

Git merge will combine multiple sequences of commits into one unified history. In the most frequent use cases, git merge is used to combine two branches.

Share this example with them:

Suppose you and your friend are working together on a project. You have written some code. Your friend may want to propose a more efficient way of doing something. They could change your code, but that is a risk. If their idea does not work out, you will have to lose your working code as well. In such a case, you can ask your friend to create a new branch of your folder. They can try making their changes on that. If the idea works, you can merge their branch with yours.

Refer to the following links to know more:

<https://confluence.atlassian.com/bitbucket/branching-a-repository-223217999.html>

<https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging>

Slide 4



Using slide 4, tell the students how to create a new branch in GIT.

Ask them the question:

Q. What does checkout do?

A. Checkout is the Git term for moving to a new branch.

git checkout -b newbranchname

You will be switched to a new branch where you can make changes to your project. In order to merge your changes with the master branch, you would have to switch back to the master branch with the command.

git checkout master

This command will switch you back to the master branch where you can merge your changes using the command.

git merge newbranchname

The changes made on the branch *newbranchname* have now been added to the master branch and can be committed.

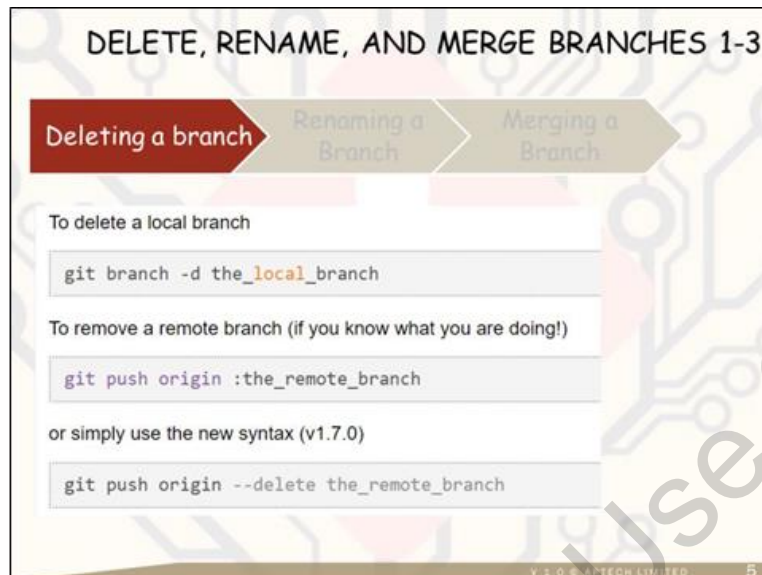
Share this example:

Consider that your friend has created a new branch and made changes on it. They will ask you to see if the changes are acceptable. You can then view these changes by checking out their branch.

Use this example to gain a better understanding of the concept - <https://backlog.com/git-tutorial/branching/create-branch/>

For Aptech Centre Use Only

Slide 5



Using slide 5, tell the students how to delete branches.

Example:

Share this example:

Suppose the changes your friend made to the code are not working. They will then delete their branch and start over. The '-d' stands for delete.

How to delete local branches in Git?

`git branch -d`: Using the '-d' flag, inform 'git branch' which item to delete. Use '-f' flag if you are trying to delete a branch that contains *unmerged changes*. **How to delete remote branches in Git?**

To delete a remote branch, do **not** use the 'git branch' command - but instead 'git push' with the '--delete' flag:

```
git push origin --delete feature/login
```

Deleting both a local and a remote branch

Remember that local and remote branches have nothing to do with each other. They are separate objects in Git.

Refer to the following link to know more: <https://www.atlassian.com/git/tutorials/using-branches>

Slide 6



Using slide 6, tell the students how to rename branches.

Consider a project you are working on, creating multiple feature branches. Then, your dev team decides to change their feature branch naming convention. How do you proceed? Can you easily accommodate your team's plan? It is super simple to rename your branches.

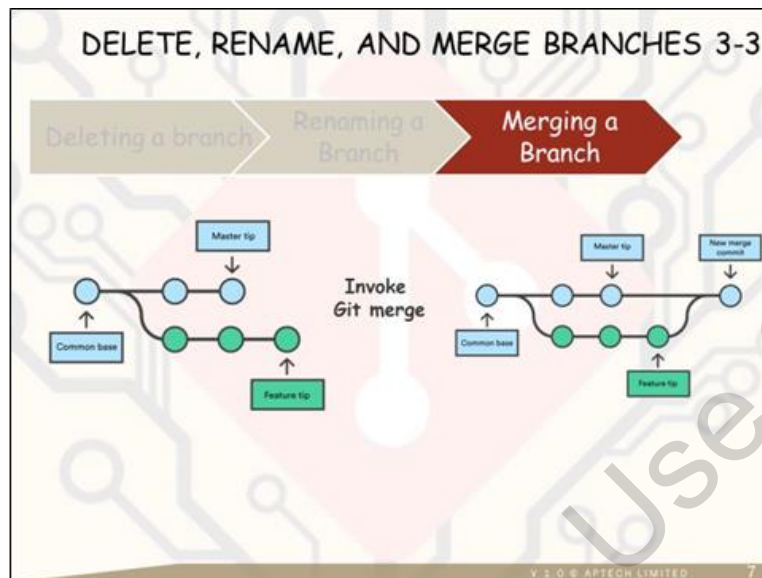
Let us say you have an existing branch called 'My_Feature,' and you want to rename it to 'Your_feature'.

First, you want to change your local branch. This is easy:

```
git branch -m my_feature your_feature
```

Then, you change the remote branch. This is a bit more complex, because you cannot directly rename the remote branch. Instead, you have to delete the remote branch with the old name and recreate it with the new name.

Slide 7



Using slide 7, tell the students how to merge branches.

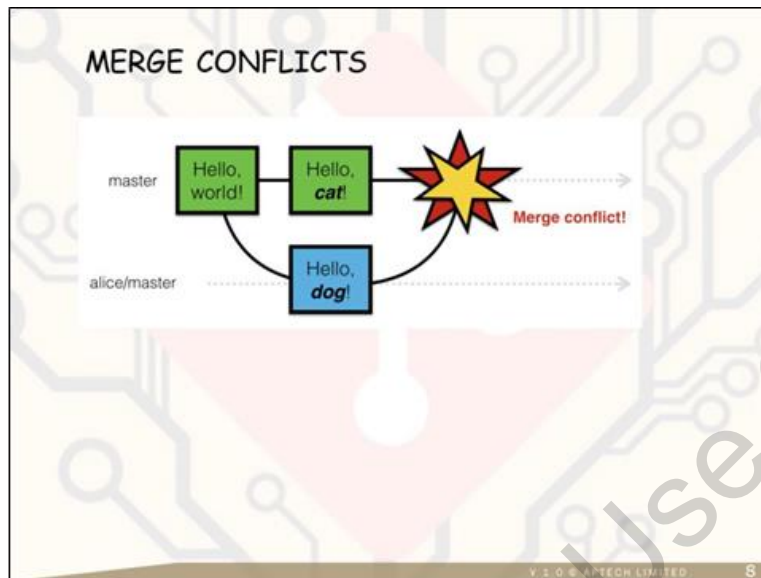
Merging is Git's way of putting a forked history back together again. The `git merge` command takes the independent lines of development created by git branch and integrate them into a single branch. The merge will combine multiple sequences of commits into one unified history. In the most frequent use cases, **git merge** is used to combine two branches.

Share this example:

When you have checked the changes that your friend made to the project, and if they are acceptable, you can merge their branch with yours.

Refer to the following link to know more: <https://www.atlassian.com/git/tutorials/using-branches/git-merge>

Slide 8



Using slide 8, tell the students about Merge Conflicts and how to resolve them.

Merge conflicts happen when you merge branches that have competing commits, and Git requires your help to decide which changes to incorporate in the final merge.

Git can often resolve differences between branches and merge them automatically. Usually, the changes are on different lines, or even in different files, which makes the merge simple for computers to understand.

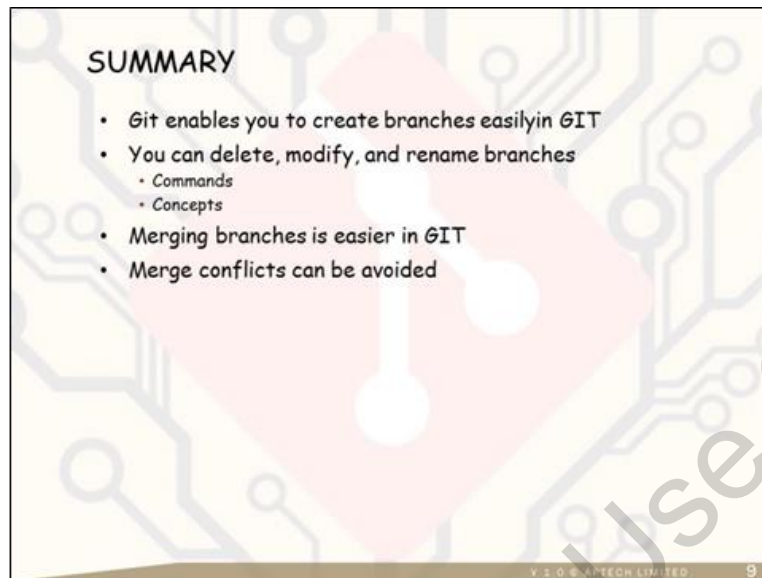
Share this example:

Suppose your friend did not add new code at the end of yours, rather they changed a line in your code. The Git system will face a conflict as it will not know which version is correct and what should get stored in that particular line in the merged version.

Here, you will have to tell the Git system what to do and resolve the conflict.

Refer to the following link to know more: <https://help.github.com/articles/about-merge-conflicts/> <https://www.git-tower.com/learn/git/faq/solve-merge-conflicts>

Slide 9



SUMMARY

- Git enables you to create branches easily in GIT
- You can delete, modify, and rename branches
 - Commands
 - Concepts
- Merging branches is easier in GIT
- Merge conflicts can be avoided

V 1.0 © APTECH LIMITED 9

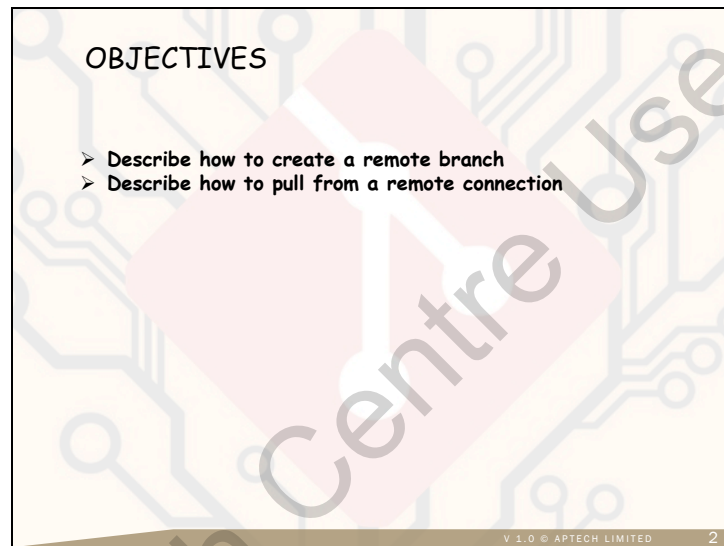
Using the slide 9, give the students a summary of the session.

Session 4: Different Approaches to Git

4.1 Pre-Class Activities

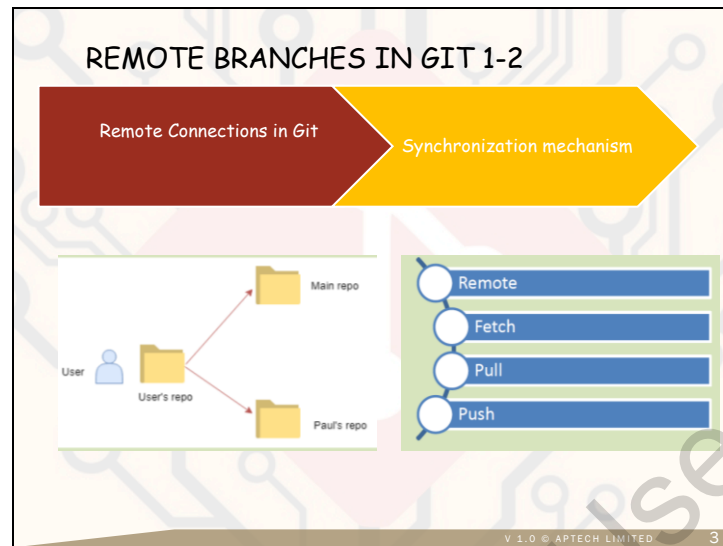
Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

Slide 2



Using slide 2, give the students an overview about the objectives of the current session. Read out the objectives mentioned.

Slide 3



Using slide 3, tell the students what remote connections are and why they are useful.

Ask the students this question -

Q. What is a remote connection?

- A remote connection is a link between your repository and one on a remote computer.

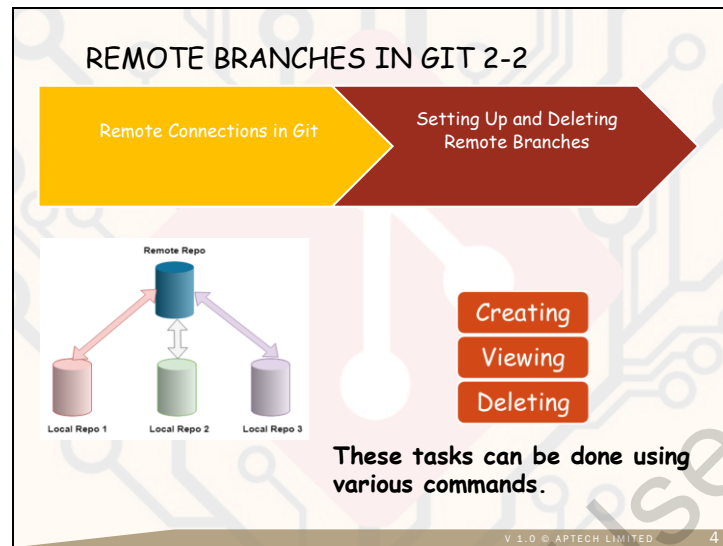
Defining Remote

Tell the students that the purpose of a remote connection is to create a link from their own repository on a local machine to any other repository on a local (that is, a remote) machine. Tell them that each Git repository has a configuration file, which is stored in the .git repository. Also, tell them that this file keeps track of all the remote connections. Show them how to use commands for the following:

- Viewing remotes
- Adding remotes
- Renaming remotes
- Deleting remotes

Refer to following link for more information: <https://www.atlassian.com/git/tutorials/syncing>

Slide 4



Using slide 4, tell the students how remote branches function.

Ask them the question:

Q. What is a branch?

A. A branch is a movable pointer to one of the older commits in the history of your project.

Tell them that a remote branch is one that exists in a remote repository. The same git branch command used to perform operations on local branches can be used for remote branches as well.

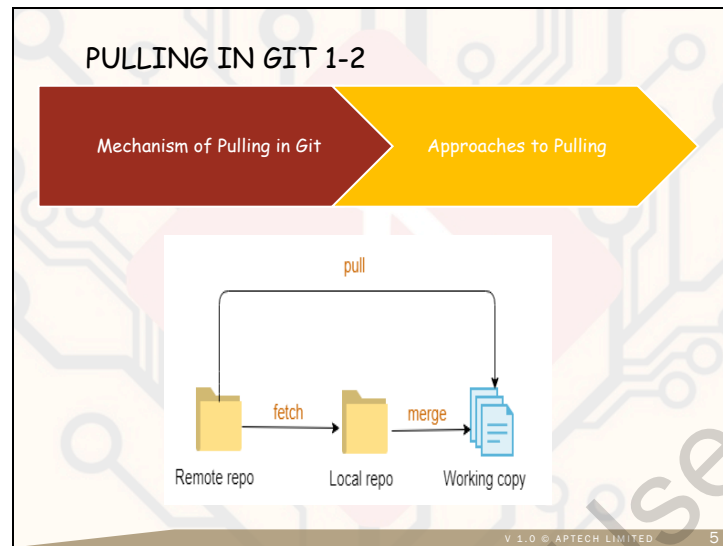
Show the students how remote branches are:

- Created
- Deleted

Help them remember the commands by practicing the syntax.

Refer to following link for more information: <https://git-scm.com/book/en/v1/Git-Branching-Remote-Branches>

Slide 5



Using slide 5, tell the students about pulling in Git. Connect this to the previous explanation about remotes by telling them that while the remote operation is used to create and maintain connections between repositories in Git, users will copy the changes made in a remote repository into their own local repository. This is done by pulling.

Explain that pulling is essentially the output of two commands:

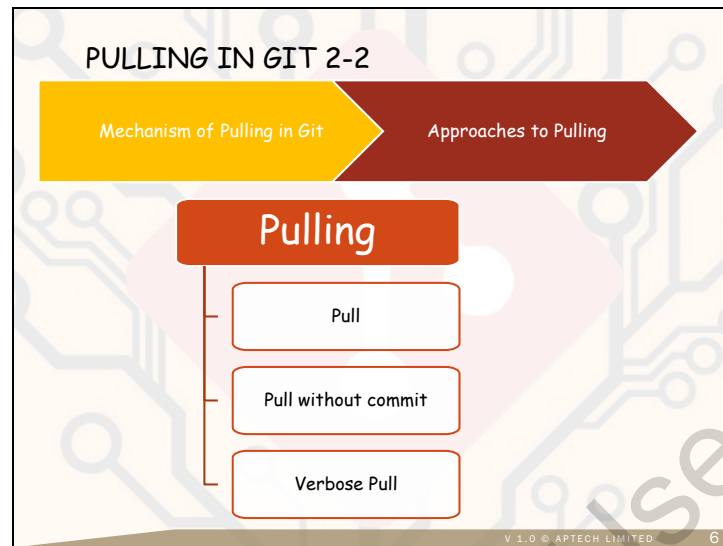
- Fetch
- Merge

The 'git fetch' command downloads commits, files, and refs from a remote repository into your local repo. Fetching is what you do when you want to see what everybody else has been working on.

Refer to following link for more information:

<https://www.atlassian.com/git/tutorials/syncing/git-fetch>

Slide 6



Using slide 6, explain about pulling.

Pulling is important for synchronizing the work done in different locations on the same project. It helps people work together and get updated with any changes that others have made.

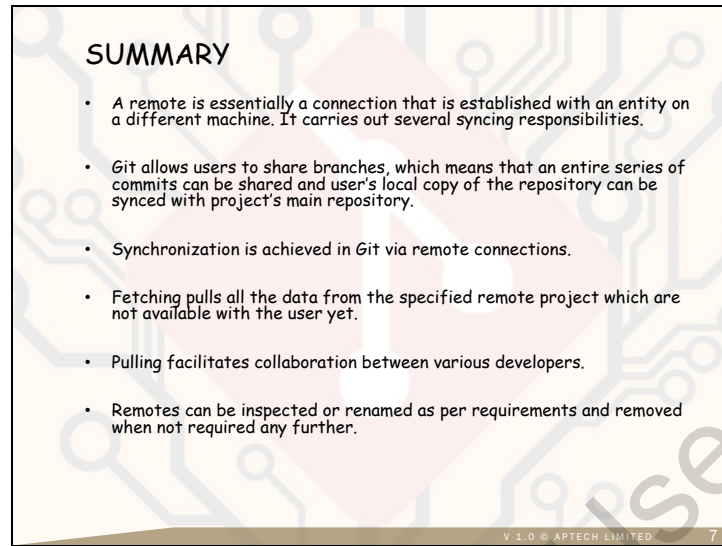
Tell the students that there are three major ways in which pulling can be carried out:

- Normal pull
- Pull without a commit
- Verbose pull

Explain each of these with the commands and help them remember the commands through repetition.

Also, explain the difference in the options and when each of them should be used.

Slide 7



The slide is titled "SUMMARY" and contains a bulleted list of six points. The background is a light yellow with a faint circuit pattern. A large, diagonal watermark reading "For Aptech Centre Use Only" is overlaid across the slide. At the bottom right, there is a small footer that reads "V 1.0 © APTECH LIMITED" followed by the number "7".

SUMMARY

- A remote is essentially a connection that is established with an entity on a different machine. It carries out several syncing responsibilities.
- Git allows users to share branches, which means that an entire series of commits can be shared and user's local copy of the repository can be synced with project's main repository.
- Synchronization is achieved in Git via remote connections.
- Fetching pulls all the data from the specified remote project which are not available with the user yet.
- Pulling facilitates collaboration between various developers.
- Remotes can be inspected or renamed as per requirements and removed when not required any further.

V 1.0 © APTECH LIMITED 7

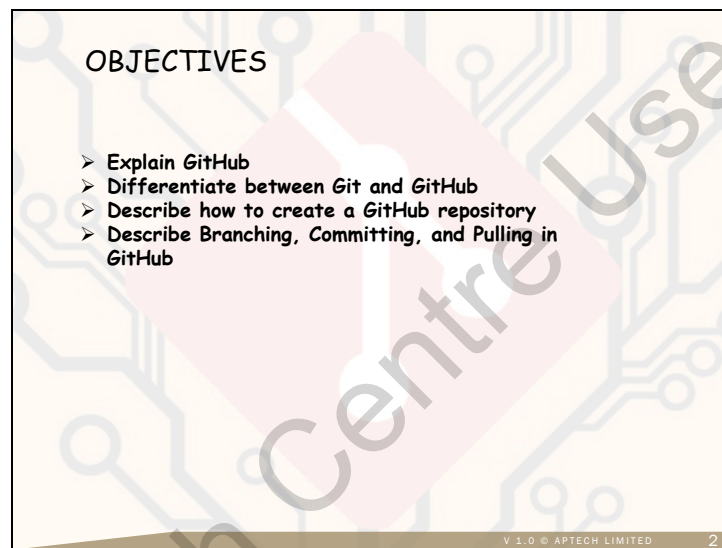
Using slide 7, give the students a summary of the session.

Session 5: GitHub as a Project Repository

5.1 Pre-Class Activities

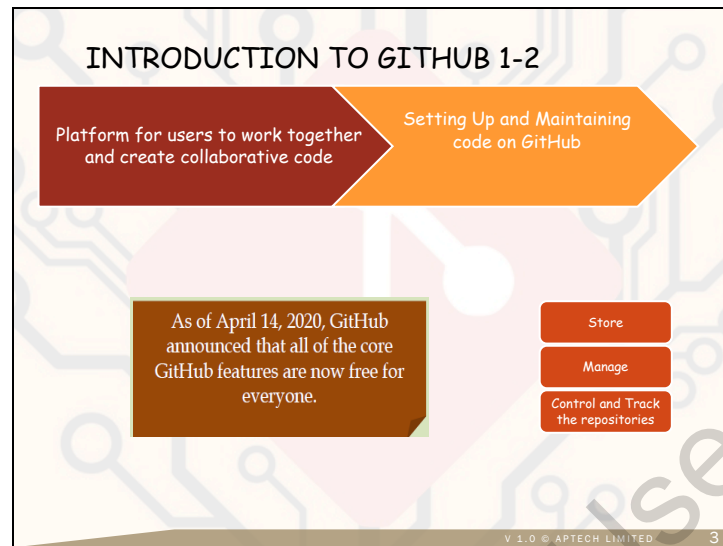
Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

Slide 2



Using slide 2, give the students an overview about the objectives of the current session. Read out the objectives mentioned.

Slide 3



Using slide 3, introduce about GitHub.

GitHub is a Website that is built on the Git version control system. It is thus, a platform for users to work together and create collaborative code. It has all the functionalities offered by the distributed Git version control system, as well as a few additional features. The user will thus, be able to set up and maintain code on GitHub.

GitHub is a code-hosting platform for collaborating with the end users. It lets you and others work together on projects from anywhere. GitHub is world's largest coding community. Developers can build code, track changes, and innovate solutions to problems that might arise during the site development process simultaneously. Non-developers can also use it to create, edit, and update website content. It is a social networking site for programmers that many companies and organizations use to facilitate project management.

The Website is hosted on <https://github.com>.

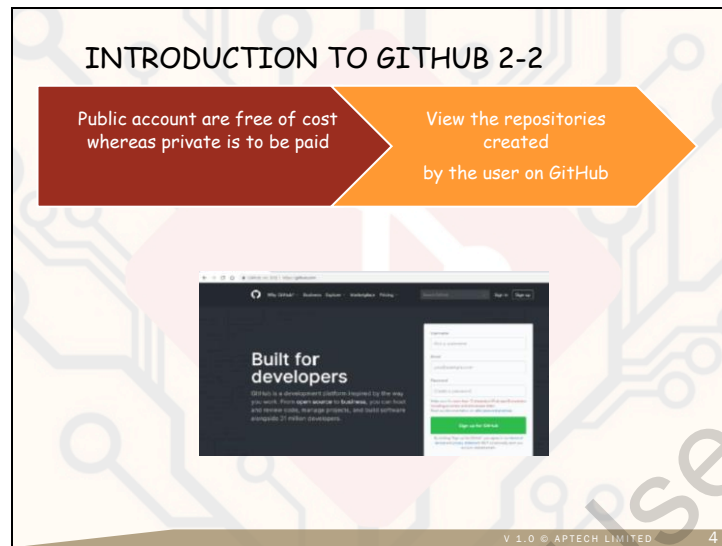
GitHub is a platform that helps users to store, manage, and control the repositories, track them and their status in order to know if any problem has occurred and solve accordingly.

Refer to following links for more information on GitHub:

<https://www.simplilearn.com/tutorials/git-tutorial/what-is-github>

<https://digital.gov/resources/an-introduction-github/>

Slide 4



Using slide 4, explain how to create GitHub Account.

- The user will create a GitHub account on the GitHub home page.
- The public accounts on GitHub are free of cost.
- These can be viewed by anyone. This means that anyone can view the repositories created by the user on GitHub.
- To get a private repository, the user has to have a paid account on GitHub.

Slides 5 and 6

GITHUB WEB INTERFACE 1-2

Collaboration of users

Extend and Explore GitHub



Marketplace menu displays tools/apps on GitHub



Enables users to check topics, trending repositories, and more

V 1.0 © APTECH LIMITED 5

GITHUB WEB INTERFACE 2-2

Cloning Mechanism

Process of Pushing

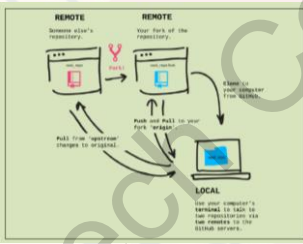


Image Courtesy: http://lord.us/git-it/challenges/forks_and_clones.html



Image Courtesy: <https://levelup.gitconnected.com/how-to-sync-forked-repositories-using-git-or-github-2933e497fa16>

V 1.0 © APTECH LIMITED 6

Using slides 5 and 6, tell the students about GitHub Web Interface.

GitHub Web Interface helps in collaboration and extending features. Tell students that by using the GitHub Web Interface, users can browse the changes, commit the required changes, create branches, and so on.

Collaboration is a process where it allows different users to work on the same project, thereby completing the work faster. Collaboration involves the following:

- Ask for username
- Navigate the main page of repository
- Click Settings- under the repository name

- Click 'Manage the access'
- Invite the collaborator and start with the required task

Refer to the following link for more information: <https://coderefinery.github.io/github-without-command-line/creating-using-web/>

Using slide 6, explain about cloning and pushing process.

To create a local copy of the forked GitHub repository on your local computer, you can use the clone feature. Cloning enables you to create a local copy.

Pushing is a process of sending committed changes from local repository to a remote repository on GitHub.com.

Cloning helps to do the following:

- Contribute to Organizational projects
- Use Open Source Repositories

The git push command takes two arguments:

- Remote name
- Branch name

Refer to following links for more information:

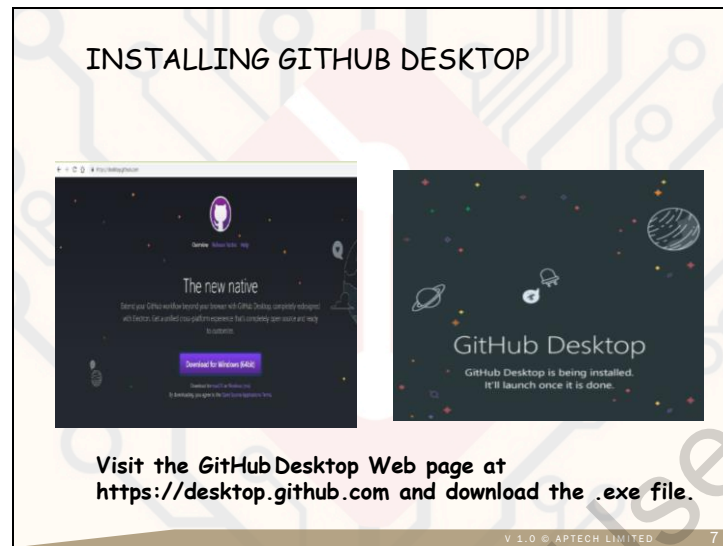
<https://digitalthoughtdisruption.com/2020/09/09/clone-repository-push-changes-using-git-bash/>

<https://www.toolsqa.com/git/git-clone/>

<https://heardlibrary.github.io/digital-scholarship/manage/control/github/clone/>

<https://docs.github.com/en/get-started/using-git/pushing-commits-to-a-remote-repository>

Slide 7



Using slide 7, explain steps to install the GitHub Desktop.

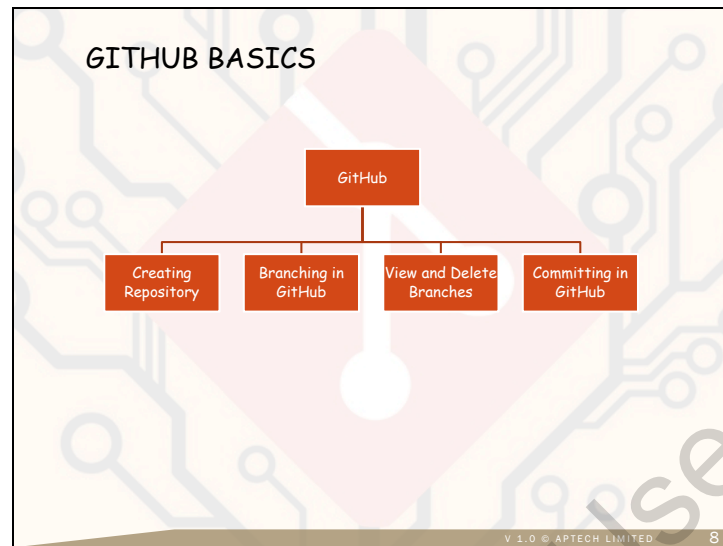
The installation for GitHub Desktop consists of few simple steps.

First, visit the GitHub Desktop Web page at <https://desktop.github.com> and download the .exe file.

Steps for installing the GitHub Desktop are as follows:

- Visit the GitHub Desktop <https://desktop.github.com> and download the .exe file.
- Run the file and follow the installation prompts.
- Navigate to options in the File menu.
- Add the GitHub account using the Sign in Option.
- Add the Git configuration options.
- GitHub Desktop will be ready to use.

Slide 8

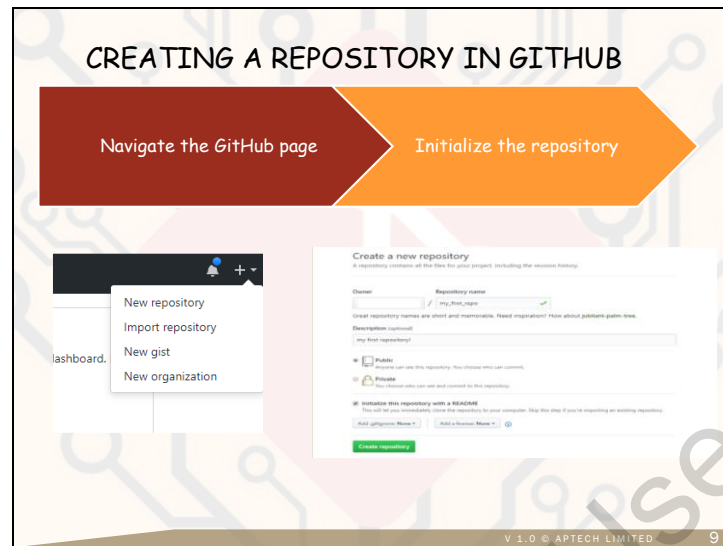


Using slide 8, explain students the GitHub Basics.

- **Creating Repository:** Repositories are created to store and collaborate the user's projects files, manage the name and location.
- **Branching in GitHub:** Branches allow users to fix bugs and develop features. This feature is used when interacting with the Version Control System (VCS), and for writing and managing code.
- **View and Delete Branches:** The branches that are newly created can be viewed to ensure the correct working. Also, if any branch is no longer required then, it can be deleted.
- **Committing in GitHub:** Commit refers to the changes that is required, when, what, and by whom.

Help them understand each point in detail.

Slide 9



Using slide 9, introduce students on how to create repository in GitHub.

Guide them through the process of navigation and initializing the repository.

Ask students this question -

Q. What is the purpose of a GitHub repository?

A. GitHub is a code-hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

Refer to following links for more information: <https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository>

<https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/overview/creating-your-first-repository-using-github-desktop>

Slide 10

BRANCHING IN GITHUB

Switch branches/tags
my_first_branch
Create branch: my_first_branch from master

View the Default and Active branches

GitHub Statistics In 2021

- Total Number of GitHub Users: **73+ million!**
- Number of repositories hosted on GitHub: **200+ million!**
- Number of Organizations: **4+ million!**

V 1.0 © APTECH LIMITED 10

Using slide 10, explain students the process of branching in GitHub.

Branch management is an important part of the Git workflow. Branches are a GitHub mechanism designed to allow users to create features, fix issues, or explore further development within their repository.

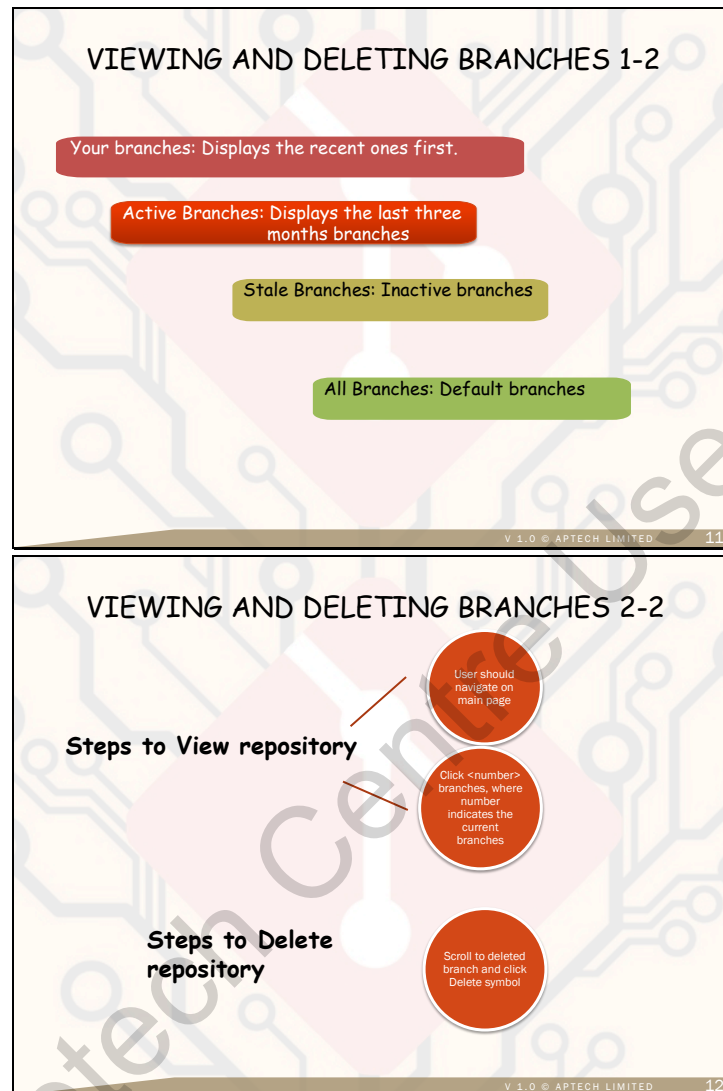
When a user creates a repository with content on GitHub.com, the repository has a single (autogenerated) branch. This first branch in the repository is the default branch. The default branch is the one that others see when they visit that repository. This is also the initial branch which Git will check out locally when someone clones that repository. The default branch of a repository is the base branch for any new pull requests and code commits.

GitHub names the default branch as main in any new repository. This can be modified by the user to a name of his/her choice. A repository can have multiple other branches. Developers can merge a branch into another branch through a pull request.

The git branch command allows users to create, list, rename, and delete the branches.

Refer to following links for more information: <https://docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop/making-changes-in-a-branch/managing-branches>
<https://thenewstack.io/dont-mess-with-the-master-working-with-branches-in-git-and-github/>
<https://www.javatpoint.com/git-branch>

Slides 11 and 12



Using slides 11 and 12, explain how to view and delete branches in GitHub.

Following are steps to view a repository:

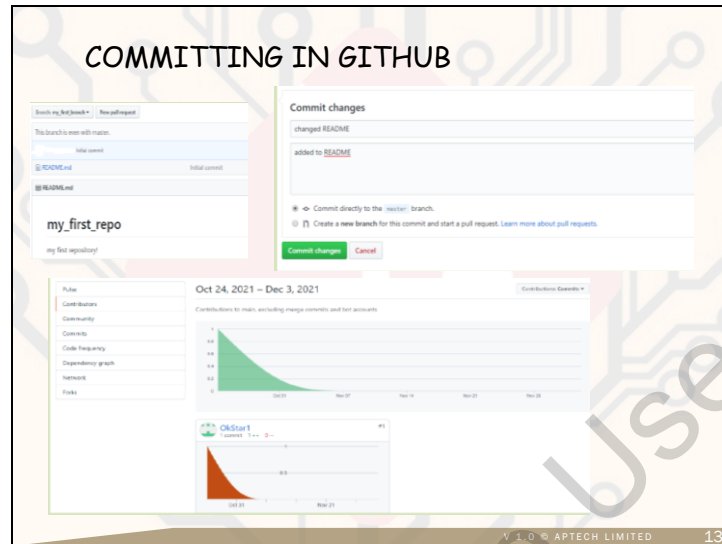
1. The user should navigate to the main page of the repository on GitHub.
2. Then, above the list of files, click <Number> branches, where Number indicates the current number of branches existing in the repository.
3. Navigate to the branches.

Following are the steps to delete branches that are no longer required:

1. The user should navigate to the main page of the repository on GitHub.
2. Then, above the list of files, click <number> branches, where number indicates the current number of branches existing in the repository.

3. Scroll to the branch that is to be deleted, then click the Delete symbol.

Slide 13



Using slide 13, explain how to commit in GitHub.

A commit represents a revision or an individual change made to a file (or set of files). Commit is a core operation in a GitHub project and enables users to save changes to the repository.

Common usages for Git Commit are as follows:

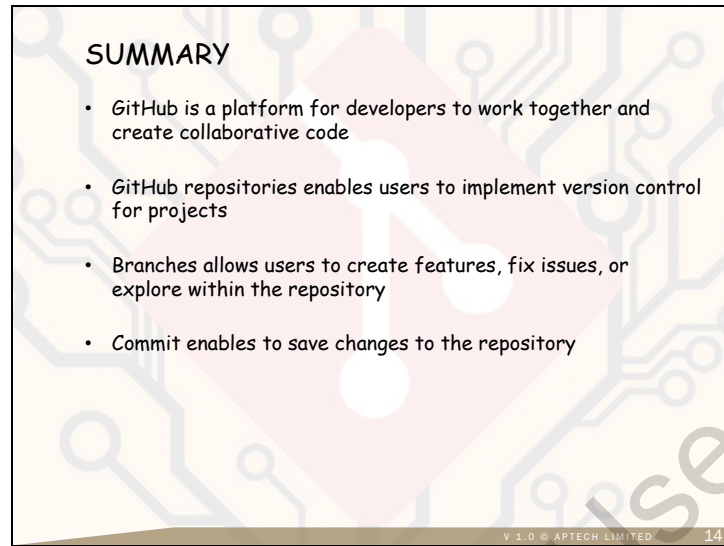
- `git commit` – Used to start the commit process
- `git commit -m` – Allows to include commit message at the same time
- `git commit -am` – Allows to skip the staging process
- `git commit --amend` – Replaces the most recent commit with a new commit

Refer to following links for more information:

<https://docs.github.com/en/pull-requests/committing-changes-to-your-project/creating-and-editing-commits/about-commits>

<https://docs.github.com/en/pull-requests/committing-changes-to-your-project>

Slide 14



SUMMARY

- GitHub is a platform for developers to work together and create collaborative code
- GitHub repositories enables users to implement version control for projects
- Branches allows users to create features, fix issues, or explore within the repository
- Commit enables to save changes to the repository

V 1.0 © APTECH LIMITED 14

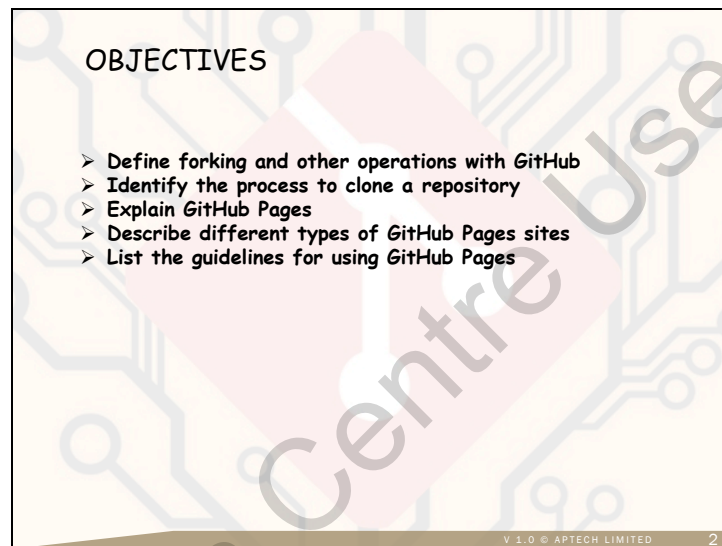
Using slide 14, give the students a summary of the session.

Session 6: More on GitHub and overview of GitHub Pages

6.1 Pre-Class Activities

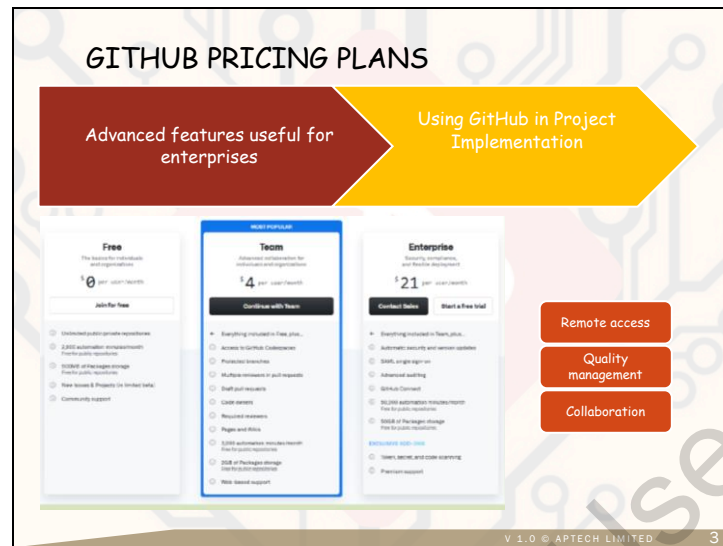
Before you commence the session, you should familiarize yourself with the topics of this session in-depth. Prepare a question or two that will be a key point to relate the current session objectives.

Slide 2



Using slide 2, give the students an overview about the objectives of the current session. Read out the objectives mentioned.

Slide 3



Using slide 3, tell the students about GitHub pricing plans.

GitHub is free for individual user accounts and basic organizational accounts. However, it also offers paid accounts with advanced features that are useful for enterprises. As per the GitHub site, the Team account is the most popular.

GitHub Enterprise Cloud users can perform the following: Actions, Advanced Security, Advisory Database, Codespaces, Dependabot Preview, GitHub Enterprise Importer, Learning Lab, Packages, and Pages.

GitHub Enterprise users can perform Actions, Advanced Security, Advisory Database, Connect, Dependabot Preview, GitHub Enterprise Importer, Learning Lab, Packages, Pages, and SQL Server Images.

GitHub AE users can perform Actions, Advanced Security, Advisory Database, Connect, Dependabot Preview, GitHub Enterprise Importer, Packages, and Pages.

Users can use the following link for comparing the features list and pricing of various editions: <https://github.com/pricing>

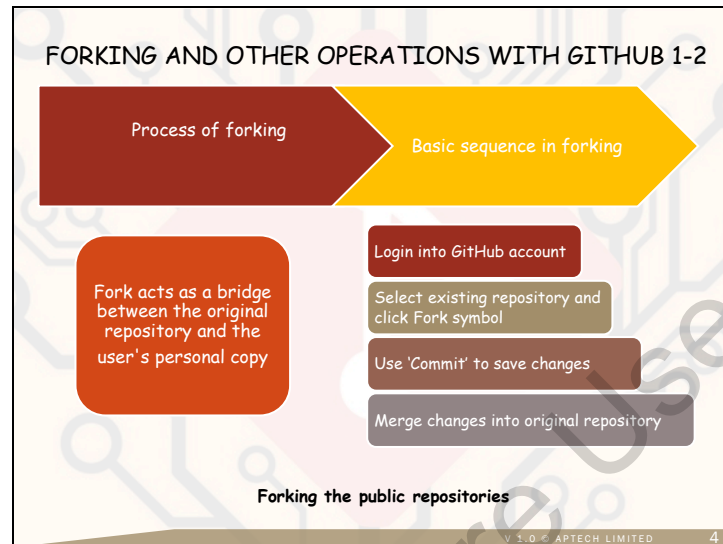
Also, explain the GitHub's feature at individual and organizational level: It helps in Remote access, Quality management, and Collaboration.

Refer to following links for more information:

<https://www.g2.com/products/github/pricing>

<https://www.getapp.com/it-management-software/a/github/pricing/>

Slide 4



Using slide 4, explain forking and other operations with GitHub.

Forking is a process that creates a copy of the repository from the server into the user's own account. Modifications can then be done as per the user's requirement, without having any effect on the original project. A fork acts as a bridge between the original repository and the user's personal copy where the user can contribute back to the original project using 'Pull Requests'.

The fork() function is used to create a new process by duplicating the existing process from which it is called.

The basic sequence in forking involves the following:

- Login in GitHub Account
- Click fork symbol by selecting the existing repository
- Use 'Commit' to save changes
- The changes performed can be now Merged

Refer to the following links for more information:

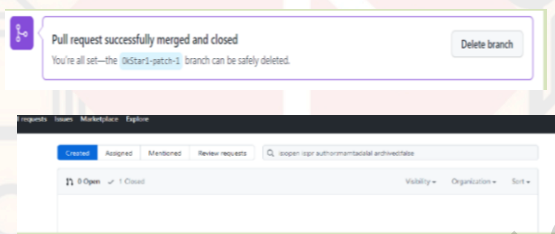
<https://docs.github.com/en/get-started/quickstart/fork-a-repo>

<https://stackoverflow.com/questions/24939843/what-does-it-mean-to-fork-on-github>

Slide 5

FORKING AND OTHER OPERATIONS WITH GITHUB 2-2

Deleting a local branch Restoring a Deleted branch



The screenshot shows a GitHub notification: "Pull request successfully merged and closed. You're all set—the `0kstar1-patch-1` branch can be safely deleted." with a "Delete branch" button. Below it is a screenshot of the GitHub pull request interface, showing a table with columns for "Created", "Assigned", "Mentioned", "Review requests", and a search bar. The table lists "0 Open" and "1 Closed" pull requests. Below the table, it says "Merging and deleting using Pull requests".

Merging and deleting using Pull requests

V 1.0 © APTECH LIMITED 5

Using slide 5, tell the students how deleting a branch can be performed.

Users can delete a local branch, after the pull request is successfully performed and the original repository has been merged with their changes. Users should remember that they cannot delete branches that are associated with open pull requests.

Slide 6

CLONING A REPOSITORY

Add/Remove files and fix Merging issues

Copying the repository from GitHub.com to the local machine

```
git clone https://github.com/nysagit/git.git
Cloning into 'git'...
remote: Counting objects: 17468, done.
remote: Compressing objects: 100% (52057/52057), done.
remote: Total 17468 (delta 113196), reused 16688 (delta 113196)
Receiving objects: 100% (17468/17468), 41.16 MiB | 1.84 MiB/s, done.
Resolving deltas: 100% (113196/113196), done.
Checking out files: 100% (2570/2570), done.
```

Quick setup — if you've done this kind of thing before

- ...or create a new repository on the command line
- ...or push an existing repository from the command line
- ...or import code from another repository

V 1.0 © APTECH LIMITED 6

Using slide 6, tell the students about Cloning a repository.

Cloning is a process that allows the user to add or remove files and fix any existing merge issues. It can also help to push larger commits. Cloning involves copying the repository from GitHub.com to the local machine. This can be done via command line or GitHub.com GUI.

Refer to following links for more information:

<https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository>

<https://www.howtogeek.com/451360/how-to-clone-a-github-repository/>

Slide 7



Using slide 7, introduce the GitHub Pages.

GitHub Pages is a really useful resource provided and hosted by GitHub.

GitHub Pages is a static site hosting service that takes HTML, CSS, and JavaScript files straight from a repository on GitHub, optionally runs the files through a build process, and publishes a Website.

GitHub is a 'social coding' site. It allows users to upload code repositories for storage in the Git version control system. Users can collaborate on code projects and the system is open-source by default, meaning that anyone in the world can find their GitHub code, use it, learn from it, and improve on it. Users can host their site on GitHub's github.io domain or their own custom domain.

Ask students this question -

Q. Why to use GitHub pages?

The main purpose is to create Websites for personal and professional use. The importance of using GitHub pages are mainly to ensure the collaboration between the programmers and users.

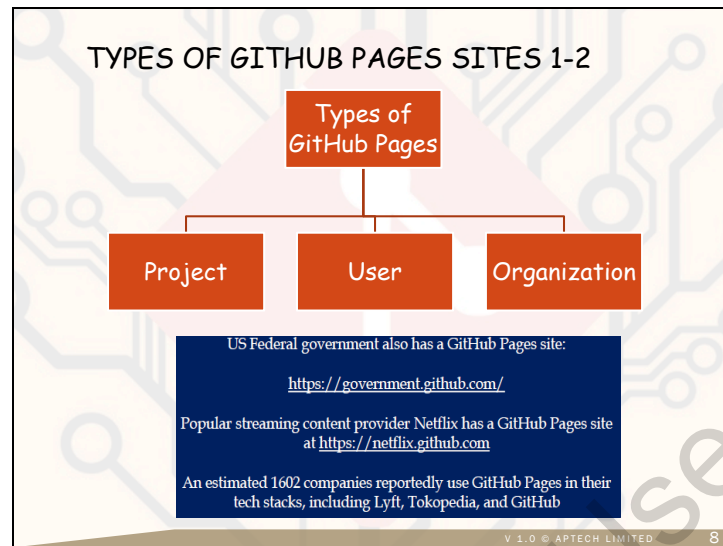
Refer to the following links for more information:

<https://lab.github.com/githubtraining/introduction-to-github>

<https://docs.github.com/en/pages/getting-started-with-github-pages/creating-a-github-pages-site>

<https://lab.github.com/githubtraining/github-pages>

Slide 8



Using slide 8, explain students the types of GitHub Pages sites. These are as follows:

- Project
- User
- Organization

Help them understand what each is useful for. **Project sites** are connected to a specific project hosted on GitHub, such as a JavaScript library or a recipe collection. **User** and **organizations sites** are connected to their specific accounts on GitHub.com.

User and organization sites are connected to a specific account on GitHub.com. To publish a user site, they must create a repository owned by their user account that is named **<username>. github.io**.

To publish an organization site, user must create a repository owned by an organization that is named **<organization>**.

Slide 9



Using slide 9, explain the steps to add theme to a Page site. These are as follows:

- Once the repository is created, users can add themes to their Pages to make it look attractive. Click Settings and go to GitHub Pages.
- The Source page is disabled hence, users can now Enable it using their Username. Click 'Choose a Theme'.
- Once the Theme is selected, user can Edit as per their requirements.
- Verification will be performed once the modification is done.

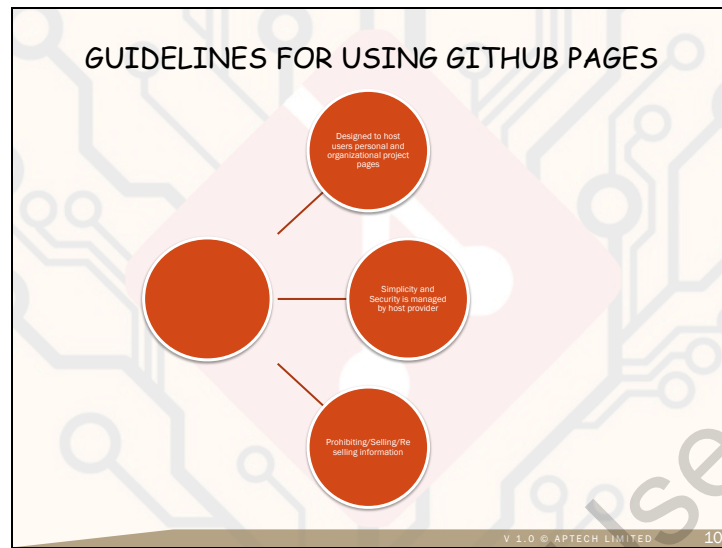
Steps for adding theme with the theme chooser:

- On GitHub, navigate to user site's repository.
- Under the user's repository name, click Settings.
- In the 'Code & operations' section of the sidebar, click Pages.
- Under 'GitHub Pages', click **Choose a theme** or **Change theme**.
- On the top of the page, click the theme the user wants, then click **Select theme**.
- User may be prompted to edit their site's *README.md* file.
 - To edit the file later, click **Cancel**.
 - To edit the file now, see "Editing files."

Refer to following link for more information:

<https://docs.github.com/en/pages/getting-started-with-github-pages/adding-a-theme-to-your-github-pages-site-with-the-theme-chooser>

Slide 10

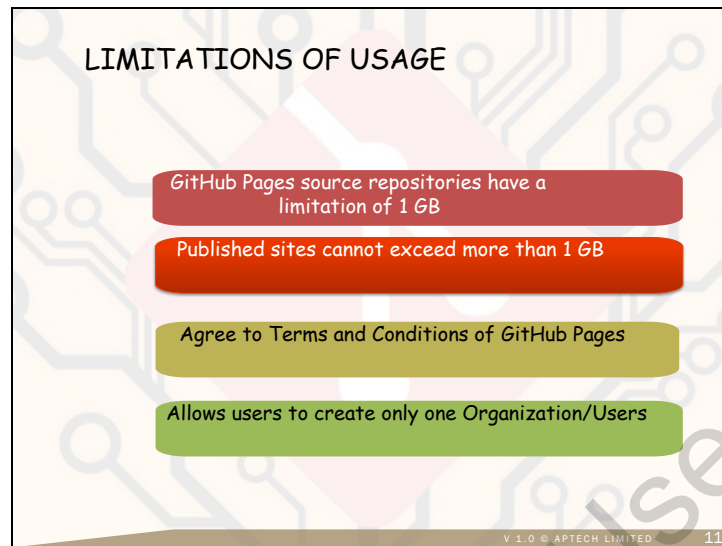


Using slide 10, explain the guidelines for using GitHub Pages. These are as follows:

- GitHub Pages are public Web pages hosted and published through GitHub, and the quickest way is by using Jekyll.
- GitHub Pages are designed to host the users' personal and organizations project pages.
- GitHub Pages simplicity is a great feature where the users are not required to be a well proficient user.
- Security is managed by the host provider.
- GitHub Pages are subject to GitHub Terms and Conditions that also include prohibiting/selling/reselling of information.

Refer to following link for more information: <https://docs.github.com/en/github/site-policy/github-community-guidelines>

Slide 11



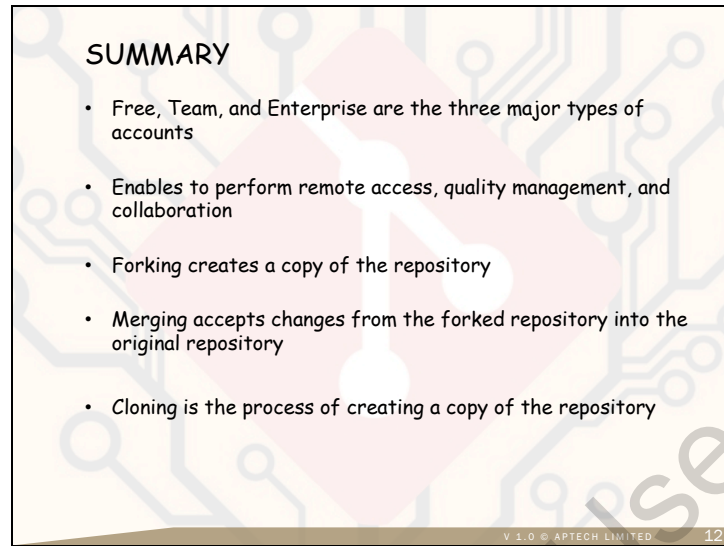
Using slide 11, explain the limitations for GitHub Pages. The limitations are as follows:

- GitHub Pages source repositories have a limitation of 1 GB.
- Published sites cannot exceed more than 1 GB.
- Agree to Terms and Conditions of GitHub Pages.
- Allows users to create only one Organization/Users.

Help to understand each of the limitation in detail. Read out the content displayed on slide 11.

Refer to following link for more information: <https://docs.github.com/en/enterprise-server@3.1/pages/getting-started-with-github-pages/about-github-pages>

Slide 12

The slide has a light orange background with a faint circuit pattern. A large, semi-transparent watermark with the text "For Aptech Centre Use Only" is oriented diagonally across the slide. The title "SUMMARY" is in bold black text. Below it is a bulleted list of five items. At the bottom right, there is a small footer containing the version "V 1.0 © APTECH LIMITED" and the slide number "12".

SUMMARY

- Free, Team, and Enterprise are the three major types of accounts
- Enables to perform remote access, quality management, and collaboration
- Forking creates a copy of the repository
- Merging accepts changes from the forked repository into the original repository
- Cloning is the process of creating a copy of the repository

V 1.0 © APTECH LIMITED 12

Using slide 12, give the students a summary of the session.