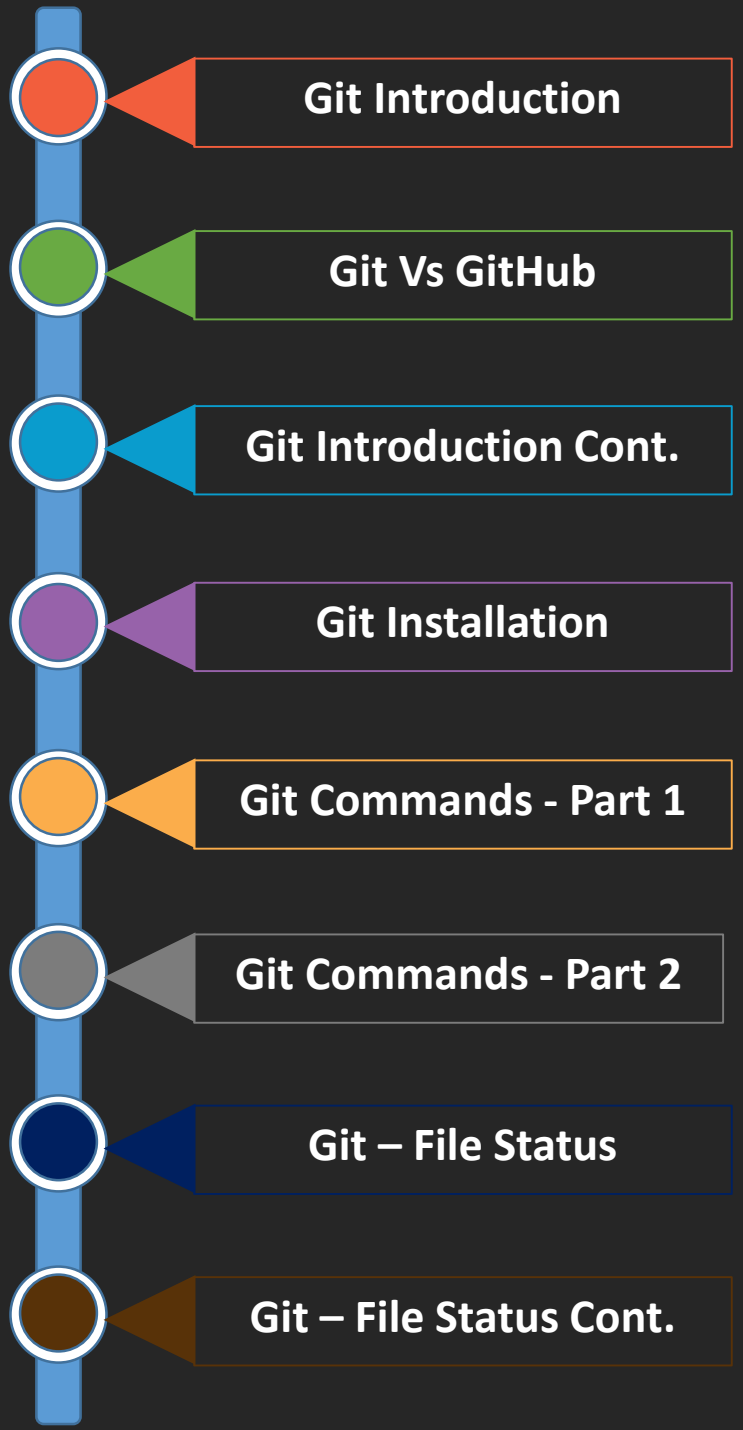
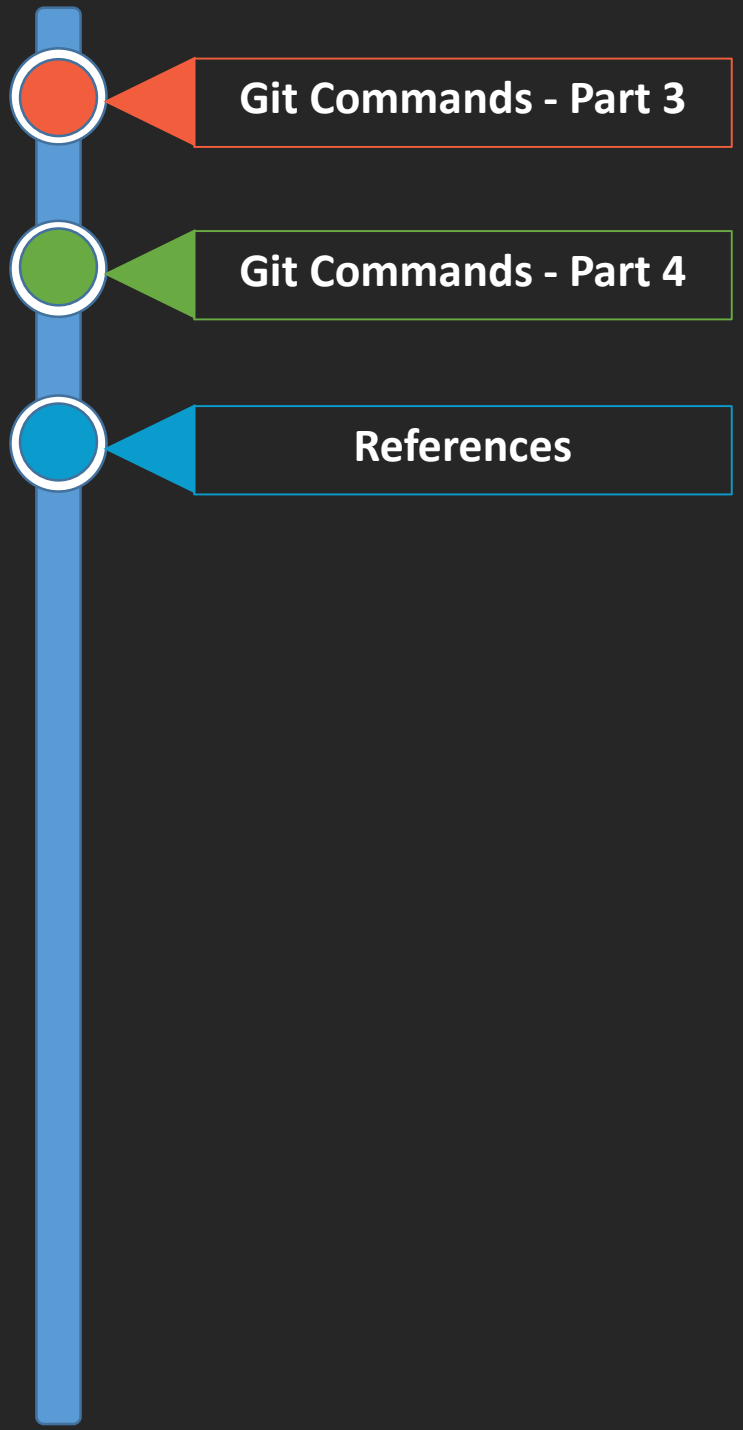
Git Intro
&
Git Commands

<created_by> Mahesh M </created_by >



- Agenda -

<created_by> Mahesh M </created_by >



- Agenda -

<created_by> Mahesh M </created_by >

Git Introduction

- **What is Git**

  - Git is **free and open source software for distributed version control**: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

- **Creator of Git**

  - Git was originally authored by **Linus Torvalds** in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Since 2005, Junio Hamano has been the core maintainer.

- **Why we use Git**

  - As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server. **Git is free and open-source software distributed under the GPL-2.0-only license.**

<created_by> Mahesh M </created_by >

Git Vs GitHub

Git	GitHub
Git is a version control system that lets you manage and keep track of your source code history.	GitHub is a cloud-based hosting service that lets you manage Git repositories. If you have open-source projects that use Git, then GitHub is designed to help you better manage them.
Software	Service
Command Line Tool	GUI
Git is installed on local system	GitHub is hosted on the web
Git is maintained by Linux.	GitHub is maintained by Microsoft.
Git is a version control system to manage source code history.	GitHub is a hosting service for Git repositories.
Git is open-source licensed.	GitHub includes a free-tier and pay-for-use tier.
Git competes with CVS, Azure DevOps Server, Subversion, Mercurial, etc.	GitHub competes with GitLab, Git Bucket, AWS Code Commit, etc.

<created_by> Mahesh M </created_by >

Git Introduction Cont.

- **Types of repos in Git**
  - Remote repository
  - Local repository

- **Sections in Local Git**
  - Local repo consists of three section -
    1. Working Area
    2. Staging Area
    3. Committed Files

- **Commit flow in local repo**
  - Commit flow in local repo is like below

    Working Area -> **Staging Area** -> **Committed Files**

<created_by> Mahesh M </created_by >

Git Installation

- **Debian / Ubuntu -**
  - Please use below commands to install git on Debian / Ubuntu -
  sudo apt update
  sudo apt install git

- **Fedora / Red Hat Linux (RHEL) -**
  - Please use below commands to install git on Fedora / RHEL -
  sudo yum update
  sudo yum install git

- **Windows -**
  - Please follow below steps to install git on Windows -
  1. Download latest git setup from - **https://git-scm.com/download/win**
  2. Git Setup wizard screen. Follow the Next and Finish prompts to complete the installation.

<created_by> Mahesh M </created_by >

Git Commands - Part 1

- **Command to check Git Version-**
  `git --version`

- **Command to check display help information about Git**
  `git --help`

- **To get help on specific git command first install git-man and then use git help command**
  `sudo apt-get install git-man`

  - Now you can use below command to check help regarding specific git command
  For example –

  `git help init`

<created_by> Mahesh M </created_by >

Git Commands - Part 2

- **Initialize Git repository -**
  - To initialize git repo, please follow below steps.
  Step 1 – Go to your working directory where you want to initialize your repo.
  Step 2 – Now execute below command to initialize git repo.

```
git init
```

- **Git command to check what changes happened**

```
git status
```

  - Git status will show you below info
    - Default or current branch
    - What commits we did
    - Untracked files
    - Status like nothing added to commit but untracked files present

<created_by> Mahesh M </created_by >

Git - File Status

- **Git Sections –**

  1. Working Area

  - When you create a file, that change will persist in working area.

  2. Staging Area

  - When you want to move your file/files changes to staging area, you need to use below command.

    To add specific file to staging area

    git add filename

    To add files present in current directory to staging area

    git add .

  3. Committed Files

  - When you want commit your changes, you need to use below command.

    git commit -m "meaningful message regarding change you made"

    When you add meaningful message with your every commit, it becomes easier for you to identify what change you actually made just by seeing that commit message

    Note - Commit saves a copy of entire file/files into .git folder (.git is hidden directory/folder).

<created_by> Mahesh M </created_by >

Git - File Status Cont.

- **Git Sections –**

  - **Unstage/Revert file change from Staging area -**

  - What if you want to unstage the file or you want to revert the changes you made in staging area for a file. To revert changes from staging area, you need to use below command.

    git restore --staged filename

    - And now your file will be restored which is present in .git folder.

    File change state transition -

    Staging area -> Working area

  - **Git to ignore tracking of specific file -**

    - Let's configure git to ignore this file permanently. To do so use below command

    echo filename >> .gitignore

    for example

    echo notes.txt >> .gitignore

To see which files are being ignored by git you can just simply see content of .gitignore file - **cat .gitignore**

Note - As you might have noticed the .gitignore file itself may be listed as untracked. It is a good practice to track the .gitignore file with git.

Git Commands - Part 3

- **Git Config Commands –**

  - To track which user made the commit, we should configure user name and email
  - To configure user name

    git config user.name “username“

  - To configure user email

    git config user.email “abc@xyz.com"

  - To display git config variables

    git config -l

<created_by> Mahesh M </created_by >

Git Commands - Part 4

- **Git log Command –**

- It shows info for commit hash, branch name, author, date, commit message.

```
git log
```

- To show git logs as one line then use below command. It shows info for commit hash, branch name, commit message.

```
git log --oneline
```

- To list changed file names in output of git log command, you can use below command. It shows info for commit hash, branch name, author, date, commit message and filename.

```
git log --name-only
```

<created_by> Mahesh M </created_by >

References

- https://www.geeksforgeeks.org/difference-between-git-and-github/
- https://www.atlassian.com/git/tutorials/install-git#windows

<created_by> Mahesh M </created_by >