Lesson 1:

Git Basics

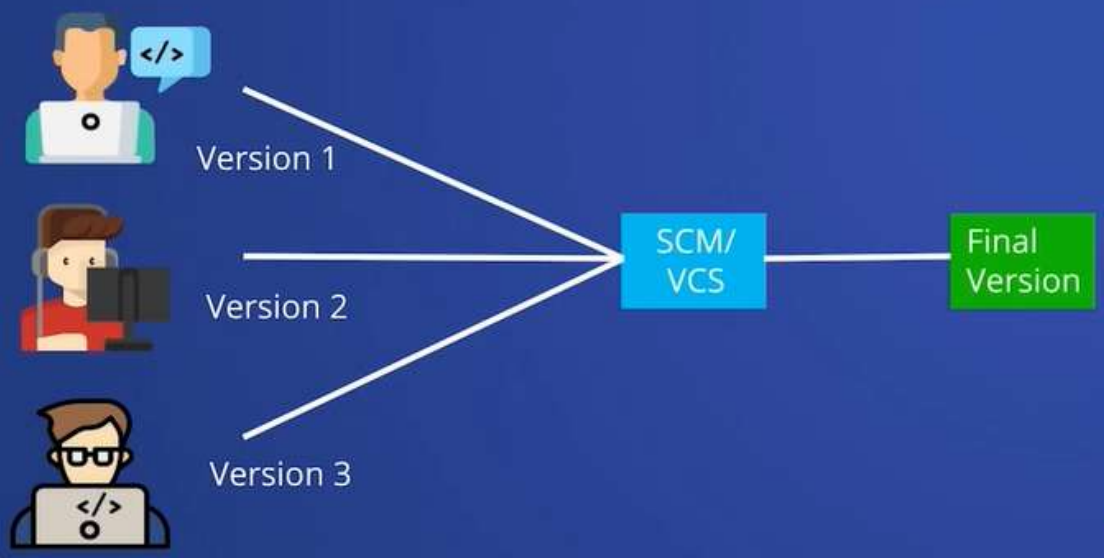
© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Source Code Management (SCM)

To manage the source code

Source Code Management (SCM)
or
Version Control System (VCS)



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Benefits of Source Code Management

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Need for Source Control

Version tracking and branching

Collaboration

Tagging

Audit

Restoring previous version

Easy integration with IDE

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Use of Source Code Management (SCM)

Protects source code

Helps teams to solve problems – Tracking and collaboration

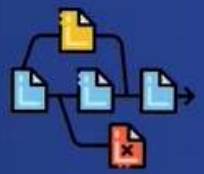
Contains long-term history of changes for each file

© LearnKartS. All rights reserved.

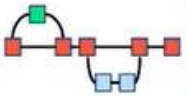
LearnkartS
Certification Prep and Projects

Use of Source Code Management (SCM)

Allows team collaboration



Easily switch on earlier file versions

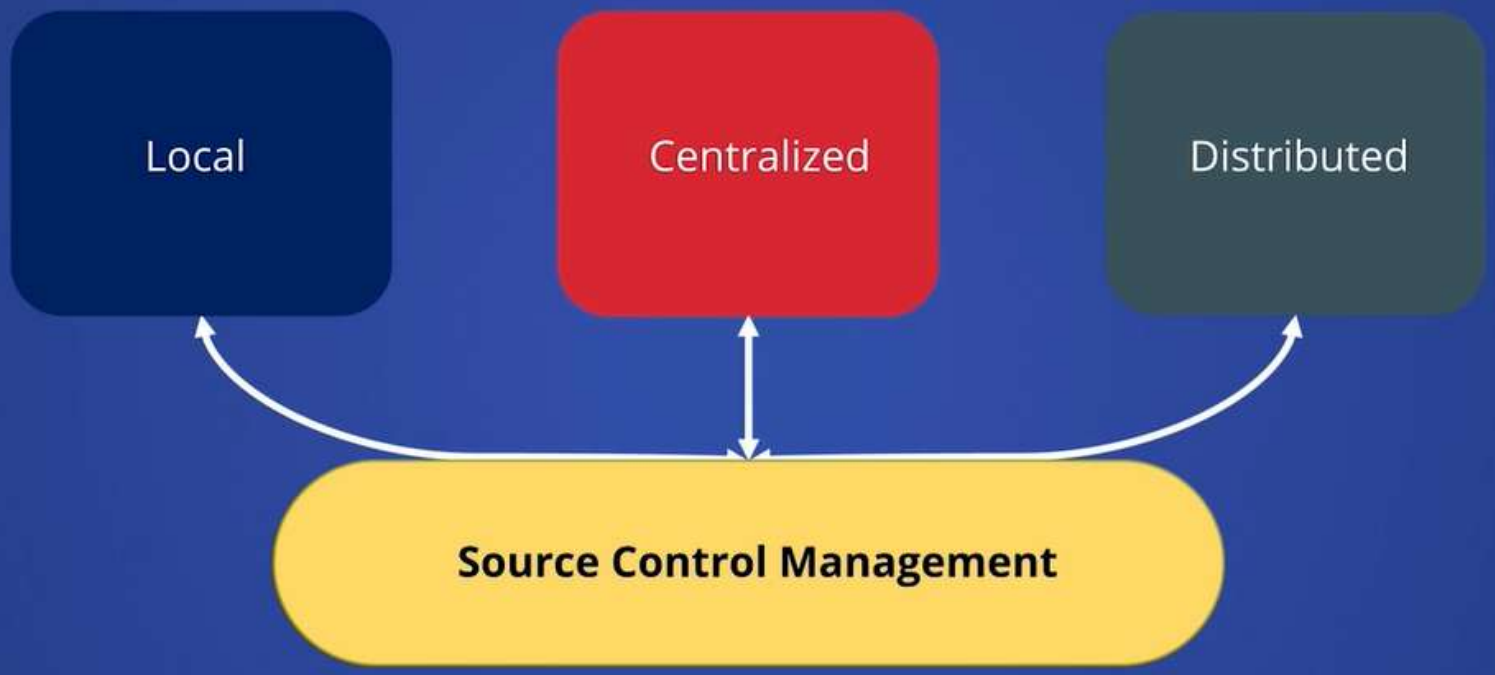


Branching and merging – maintain code as per projects/release/functionality etc.

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

3 Types of Source Control Management



© LearnKartS. All rights reserved.

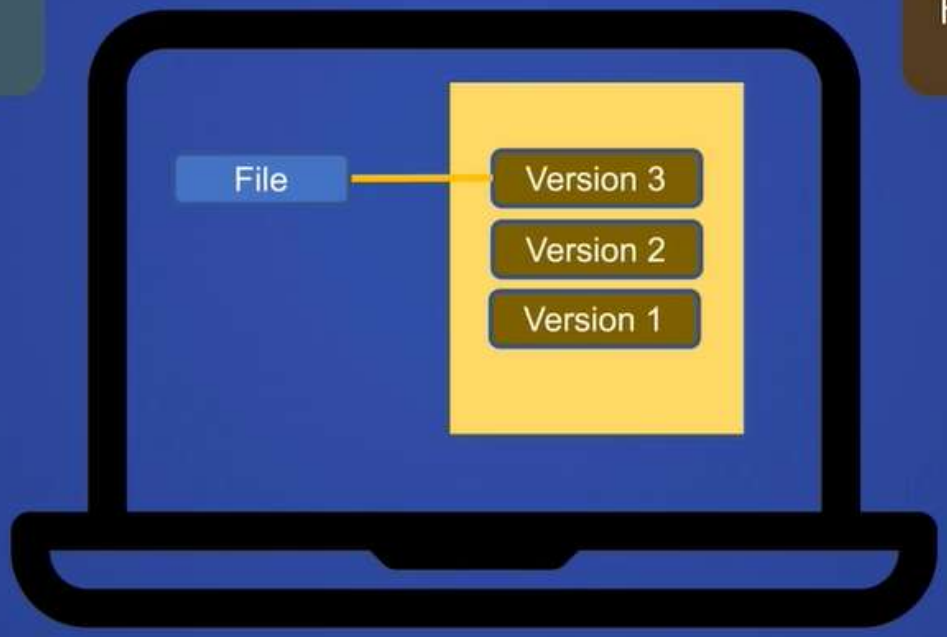
LearnkartS
Certification Prep and Projects

Local Version Control System

A local database located on local computer

File versions are stored locally

Huge risk of loosing database



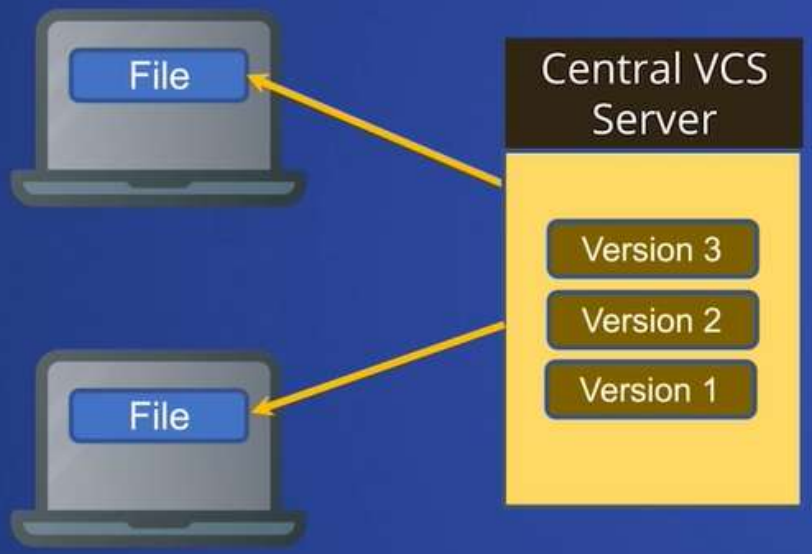
Example: RCS, SCCS

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Centralized Version Control Systems

Single "central" copy of the project on a central server



- Master-slave architecture
- Make changes in local system and then upload to central server
- Need connection with the server
- No risk of loosing database

Only 1 repository

Example: CVS

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Centralized Version Control Systems - Advantages

- Relatively easy to set up
- Provides transparency
- Enables admin to control the workflow

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Centralized Version Control Systems - Disadvantages

If the main server goes down, developers can **NOT** save versioned changes

Dependency on connection with remote

Incase central database gets corrupted, the entire history could be lost

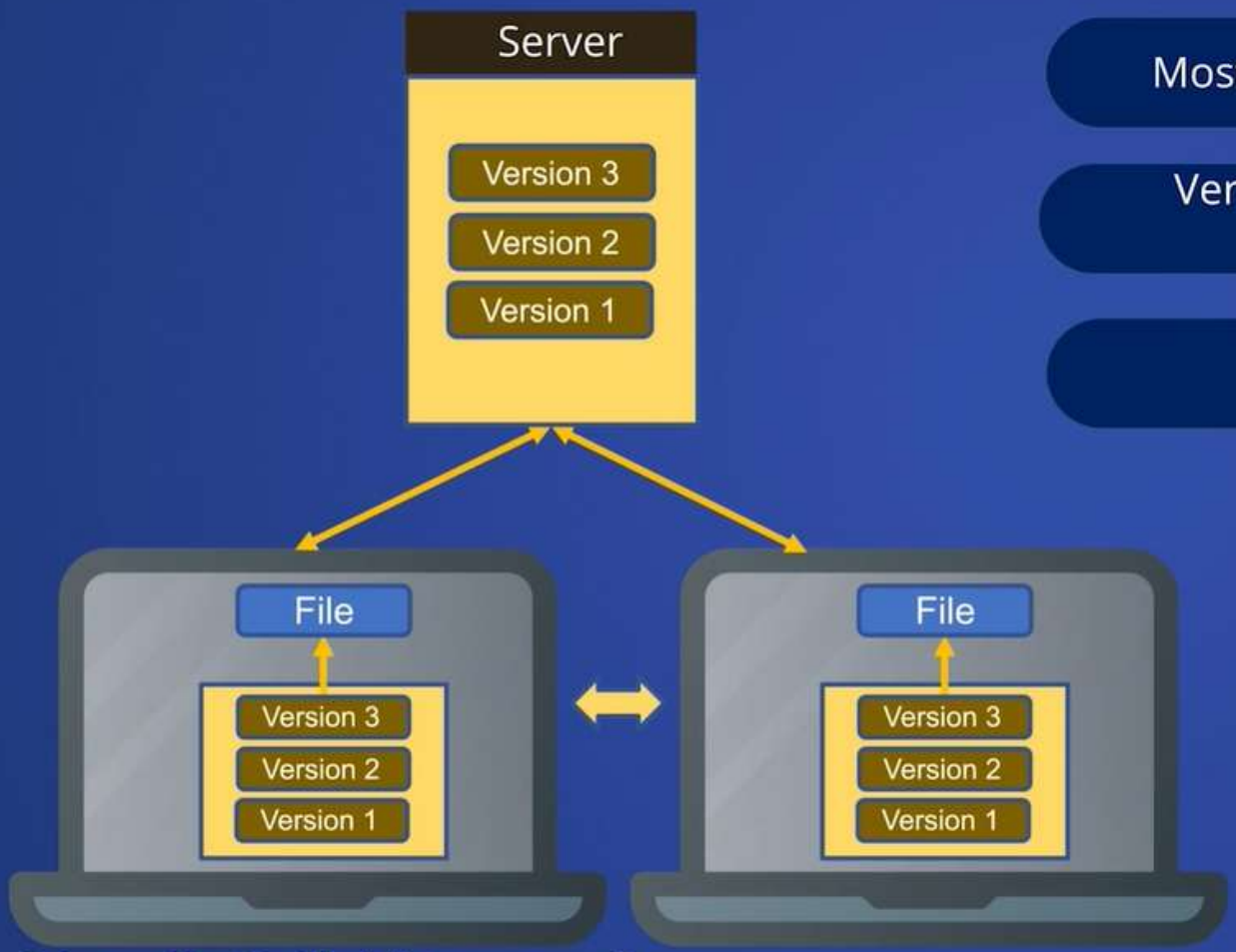
Does **NOT** provide offline access

Slow

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Distributed Version Control Systems



- Most of the mechanism is same as centralized
- Version database copy exists on individual's developer system as well
- Low risk of loosing data

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Reasons to Use Distributed Version Control Systems

No single point of failure

Allows developers to clone the repository and work on that

Entire project on local system hence faster

Work in off-line mode

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Distributed Version Control Systems

Advantages

1. **Working on branches is easier** in DVS, developer faces lesser conflicts
2. If server is down, **developer can work using local copies**
3. **Faster** as mostly user deals with local copy without hitting server every time
4. Systems are workable offline - **Entire repository on local machine**

Disadvantages

1. Difficult for beginners, multiple commands needs to be remembered
2. Git is slow for large files like 2-3gb

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Features

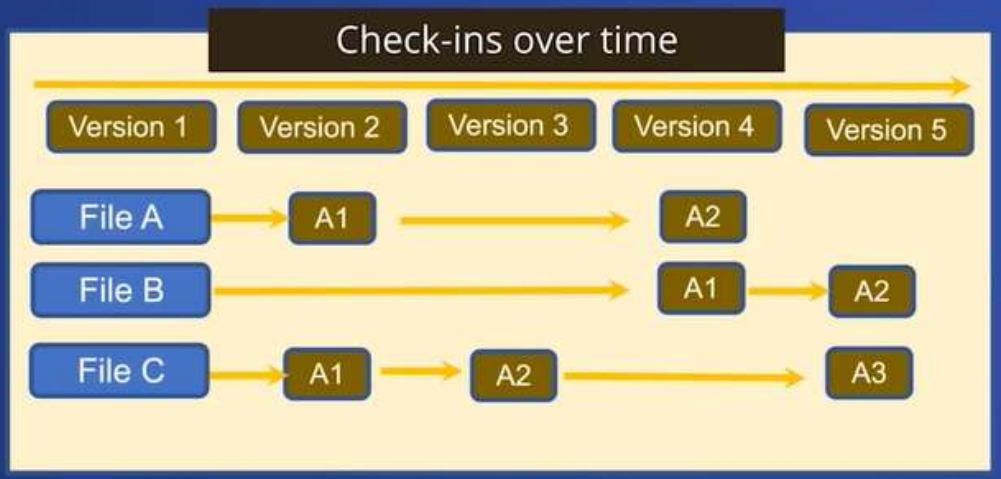
- Distributed
- Non-linear development
- Speed
- Open source
- Easy to integrate
- Compatible
- Lightweight
- Simple design
- Secure
- Able to handle large projects

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git vs Other Version Control System

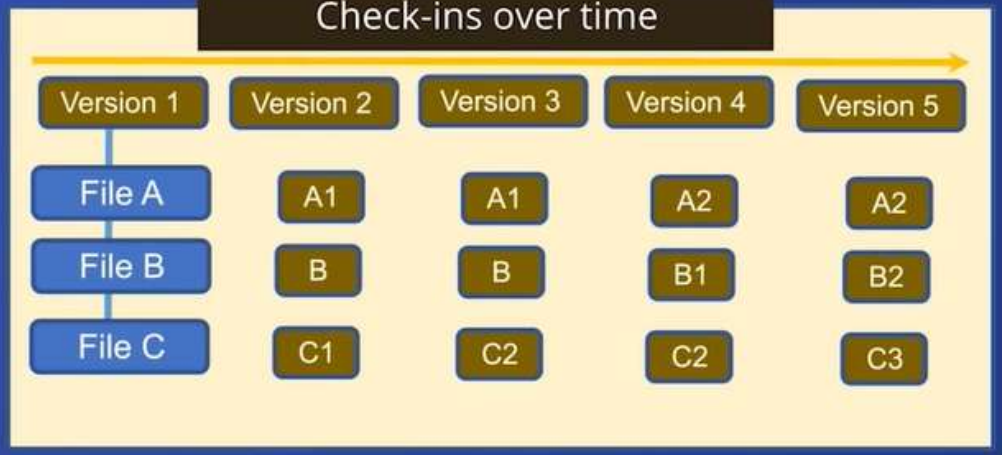
Other Version Control System



Stores data as changes to base version of each file

Git

Git considers data as **stream of snapshots**



Stores **data as snapshots** of the project over time

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Sections of a Git Project

Folder location where developer is making changes in the code

* Ready for commit files are stored
* Staging **provides a unique** id to each change set added by the working area
* Commit moves file from staging area to repository

* .git directory => **Hidden folder**
* Stores project metadata and object database

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

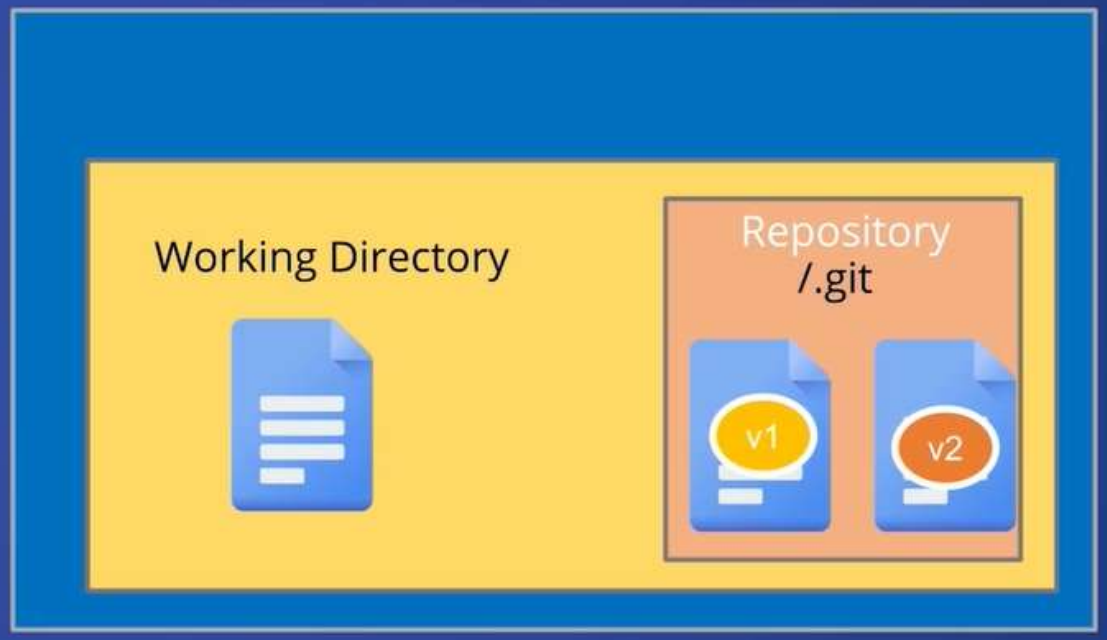
Repository

Git Repository/.git directory/Local Repository

Repository is a data structure
To **store metadata** of files and directories

Different projects have different repositories

Multiple branches related information will also be stored in repository



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Working Tree

Helps to see how files are present

Users can change the files in the working tree by modifying, creating and removing files

Using “git status” one can see what exactly is there in working tree

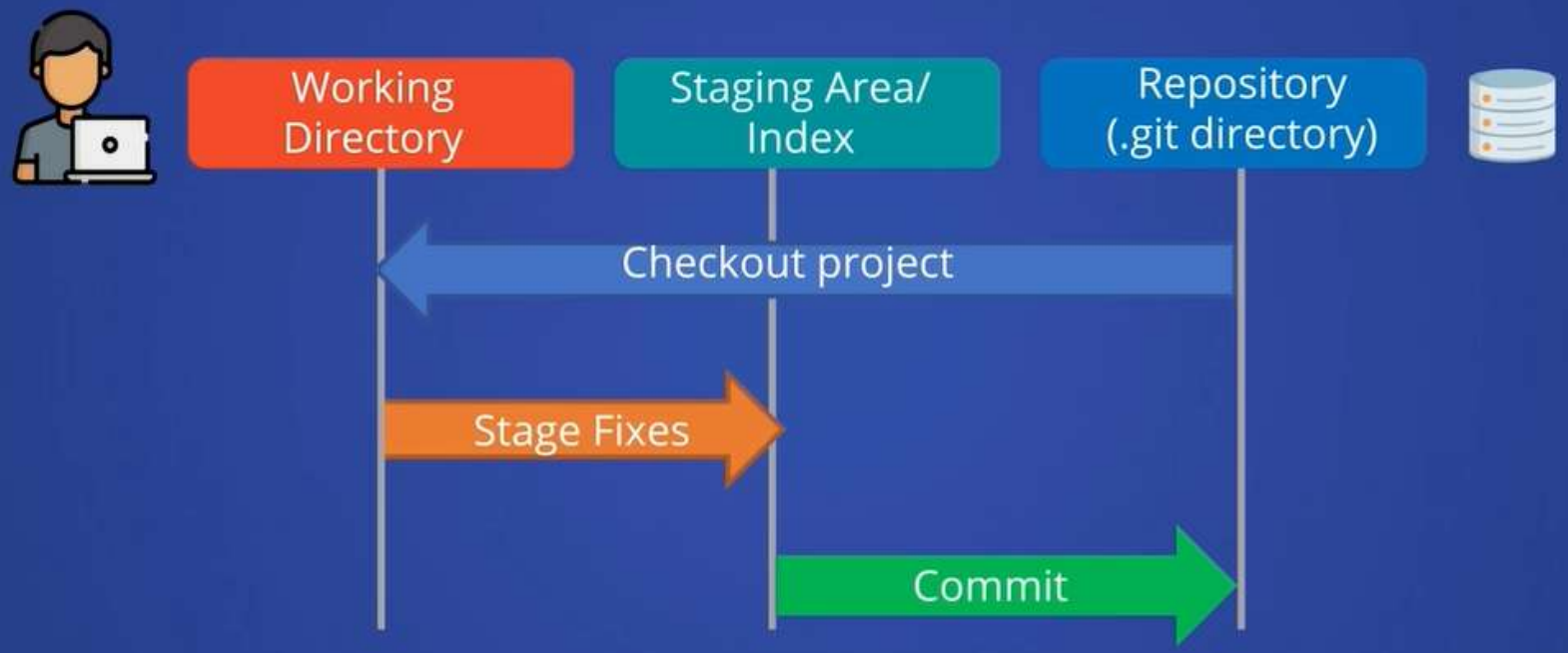
```
git status
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Commit

Commit is a collection of content



To move file from working directory to staging area

To move file from staging area to repository

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Tag

Tag is used to mark the important commit phase

Tags define state of a branch in specific repository

After development activity, one can recover the branches from tags

Two different types of tags

Light-weighted tag

Annotated tag

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

URL

Git remote

Location of the remote repository

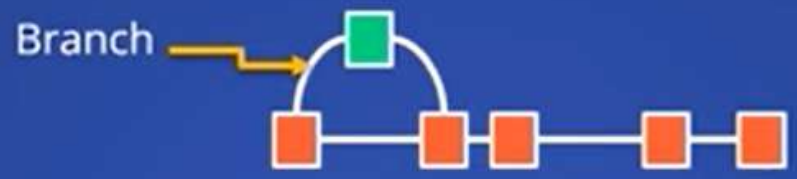
Used for PUSH and PULL operation

URL can be shared with other developers

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Branch



Different version of the repository from the main project

Helps to perform parallel development

Developers work independently on branches and then merge the code

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Fork and Checkout

Fork

To import repository from server-to-server communication

Checkout

To switch between branches in a repository

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Clone and Merge

Clone

To make a copy of remote repository to local repository

Helps to collaborate on the repository

Merge

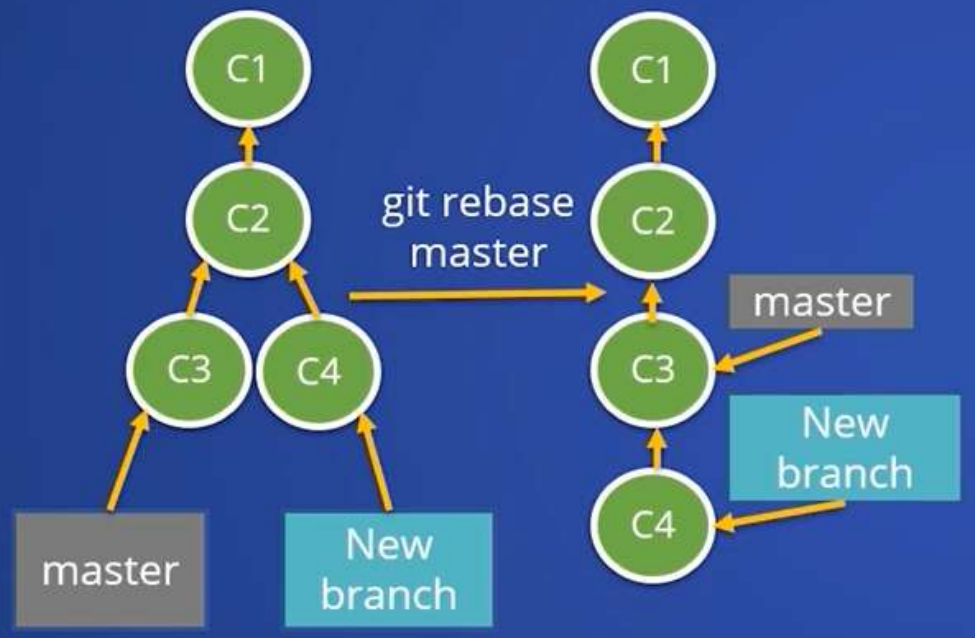
Used to collect all code done by git branch command and clubbed together into single branch

Between development branches, feature branches and master branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Rebase



- Rebase is used to merge content between two branches and convert to single branch
- To have a clean history
- To reduce number of branches

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Lesson 2:

Working with Git

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Initialization and Configuration

Author details

```
git config --global user.name "[name]"
```

```
git config --global user.name "learnkarts"
```

```
git config --global user.email "[email address]"
```

```
git config --global user.email "git@example.com"
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Add Files to Staging Area/Start Tracking

To add all non-tracked files in current repository directory

```
git add *
```

```
git add -A
```

Adds currently tracked files which have been modified to the staging area and also checks if those are deleted or not

```
git add -u
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Ignoring Files to Add to Staging Area

What Files Should be Ignored?

Add such file to a file **.gitignore**

- Runtime files such as log, lock, cache, or temporary files.
- Files with sensitive information, such as configuration files passwords or API keys
- Compiled code, such as .class, jar files, .o, so files
- Dependency directories, such as /vendor or /node_modules
- System files like .DS_Store or Thumbs.db
- IDE or text editor configuration files

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

.gitignore File Matching Patterns

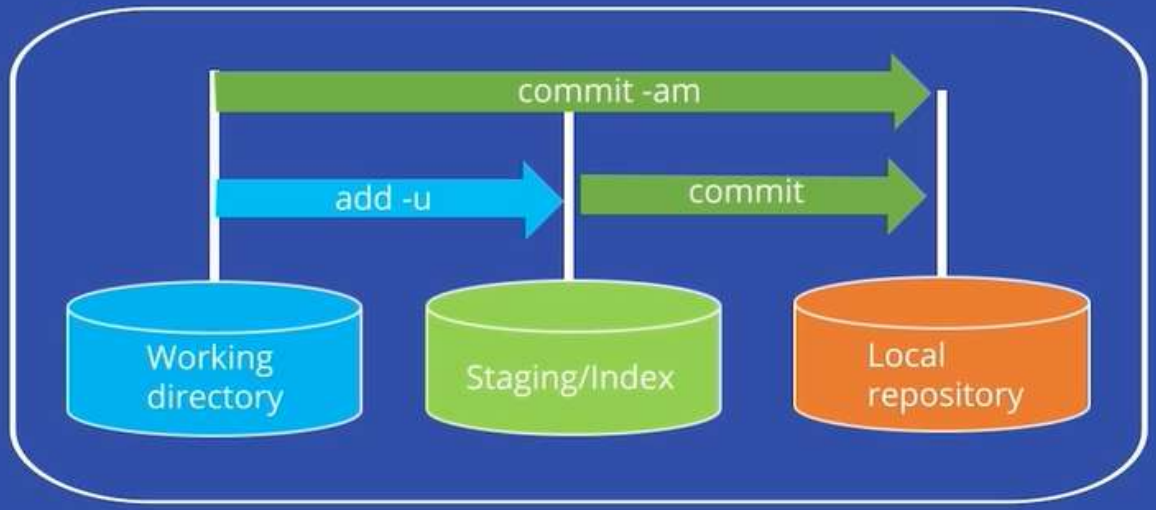
Pattern	Example Matches
/access.log	access.log
access.log	access.log logs/access.log var/logs/access.log
build/	build
*.log	error.log logs/debug.log build/logs/error.log
logs/**	Everything inside logs directory

Pattern	Example Matches
**/build	var/build pub/build build
foo/**/bar	foo/bar foo/a/bar foo/a/b/c/bar
access?.log	access0.log access1.log accessa.log
foo??	fooab foo23 foo0s

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Commit Bypassing Staging



Commit directly from working directory to repository (skipping staging)

```
$ git commit -a -m "commit bypassing staging"
```

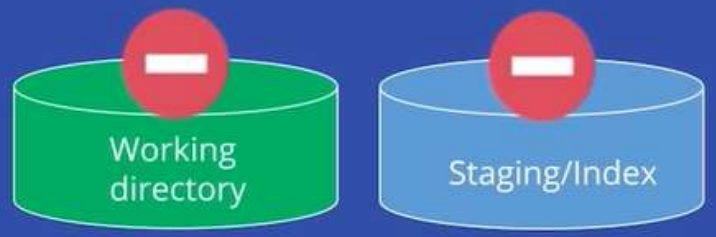
© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

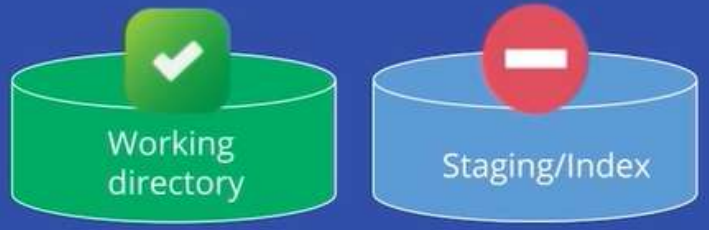
Remove Files

Remove one or more files from current tracked files

```
git rm <filename>
```




```
git rm --cached <filename>
```



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Restoring Removed Files

Deleted

To restore removed files

```
git restore <filename>
```

Previous code will be checked, if any kind of changes are done, those will be rolled back

```
$ rm Readme.md

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    Readme.md

no changes added to commit (use "git add" and/or "git commit -a")

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ git restore Readme.md

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ git status
On branch master
nothing to commit, working tree clean

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ ls -alrt
total 20
drwxr-xr-x 1 Lenovo 197121 0 Sep  5 10:14 ../
-rw-r--r-- 1 Lenovo 197121 0 Sep  5 10:25 Readme.md
drwxr-xr-x 1 Lenovo 197121 0 Sep  5 10:25 ./
drwxr-xr-x 1 Lenovo 197121 0 Sep  5 10:25 .git/
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Move Files

'git mv' command is used to rename a file/ move file-

```
git mv <filename>
```

1. Moves a file
2. Runs 'git add' on the new file
3. Runs 'git rm' on the old file

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Clean Files

```
$ git clean
```

- To remove unwanted files from working directory
- Includes removing temporary build artifacts or merge conflict files

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

What 'git clean' will NOT Remove?

Files which are being tracked

Files that are committed

Files listed in .gitignore file

To forcefully remove files listed in .gitignore

```
git clean -x
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Important Points

```
git clean -force
```

```
git clean -n
```

Does a dry run to show what it will do

```
git clean -i
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Typical Usage

`git clean -n`	Perform a dry run first
`git clean -f`	Clean untracked files
`git clean -x`	Remove files listed in .gitignore

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Check Files Difference

Commands to see how the file differs from the repository version

To see details differences

```
$ git diff HEAD
```

Latest commit

Difference between working directory and staging area

```
$ git diff
```

Difference between staging area and the last commit (head commit)

```
$ git diff --staged
```

Difference between two commits

```
$ git diff master <branch_name>
```

© LearnKartS. All rights reserved.

LearnKartS
Certification Prep and Projects

Resetting a File

Used to roll-back the changes

Resets all the file in repository

```
git reset -- hard
```

Reset a specific file

```
git reset -- hard command <filename>
```

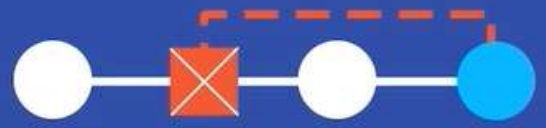
```
$ rm Readme.md
g
Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    Readme.md

no changes added to commit (use "git add" and/or "git commit -a")

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ git reset --hard
HEAD is now at f2df5ea First Commit

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ git status
On branch master
nothing to commit, working tree clean
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Revert a Commit

Reverting



```
$ git revert <commitHash>
```

Revert a commit by using commit hash

Use 'git log' to get the commit hash

```
$ git log
commit f2df5ea65b9befc32da2bf58133af54c5b9d9fe1 (HEAD -> master)
Author: Anuj Sharma <Anuj.Sharma@company.com>
Date:   Sun Sep 5 10:20:15 2021 +0530

    First Commit

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git log

To view commit history

- Shows all commits
- Author details
- Commit hash
- Commit date and time
- Commit message

```
$ git log
```

```
$ git log
commit f2df5ea65b9befc32da2bf58133af54c5b9d9fe1 (HEAD -> master)
Author: Anuj Sharma <Anuj.Sharma@company.com>
Date:   Sun Sep 5 10:20:15 2021 +0530

    First Commit

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Log

To see specific number of commits

```
$ git log -p -<number of commits>
```

To see last commit

```
git log -p -1
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Show

To get the inside information of a specific commit

```
$git show [<options>] [<object>…]
```

```
$ git log
commit f2df5ea65b9befc32da2bf58133af54c5b9d9fe1 (HEAD -> master)
Author: Anuj Sharma <Anuj.Sharma@company.com>
Date:   Sun Sep 5 10:20:15 2021 +0530

    First Commit

Lenovo@AnujSharma MINGW64 ~/Desktop/gitrepo (master)
$ git show f2df5ea65b9befc32da2bf58133af54c5b9d9fe1
commit f2df5ea65b9befc32da2bf58133af54c5b9d9fe1 (HEAD -> master)
Author: Anuj Sharma <Anuj.Sharma@company.com>
Date:   Sun Sep 5 10:20:15 2021 +0530

    First Commit

diff --git a/Readme.md b/Readme.md
new file mode 100644
index 0000000..e69de29
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Configuration

To configure git use 'git config

```
$ git config
```

```
$ git config -h
```

```
$ git config -help
```

```
$ git config
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-pattern]
    --get-all             get all values: key [value-pattern]
    --get-regexp          get values for regexp: name-regex [value-pattern]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value-pattern]
    --add                 add a new variable: name value
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Configuration Levels

01 Local Configuration

Only related to a local repository

02 Global Configuration

Accessible only to a specific user not to other users

03 System Configuration

Accessible at the global and the local level also

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Local Configuration

Configurations stored into current repository

While working on a git repository, a ".git: folder is present

Able to do the changes, perform modifications like to modify the files, add files and remove files

These configurations are NOT accessible at the system or the global level

In .git folder , a configuration file is present as-

```
.git / config
```

All the local configurations are saved

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Global Configuration

```
git config --global --list
```

Author details

Shared across multiple projects managed by a current user

User level configuration

```
$ git config --global --list
user.email=anuj@company.com
user.name=Anuj Sharma
merge.ff=false
merge.ff=true
alias.last=log -1 HEAD

Lenovo@AnujSharma MINGW64 ~/Desktop
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

System Configuration

Highest level of configuration level

Stored in **/etc/gitconfig** file

Settings can be **overridden by global and local settings**

Configuration can be **inherited** by the local and global configuration levels

```
$ git config --system --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
credential.helper=manager-core
pull.rebase=false
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master

Lenovo@AnujSharma MINGW64 ~/Desktop
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Configurations

List Local Configuration

```
$ git config --local --list
```

List Global Configuration

```
$ git config --global --list
```

List System Configuration

```
$ git config --system --list
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

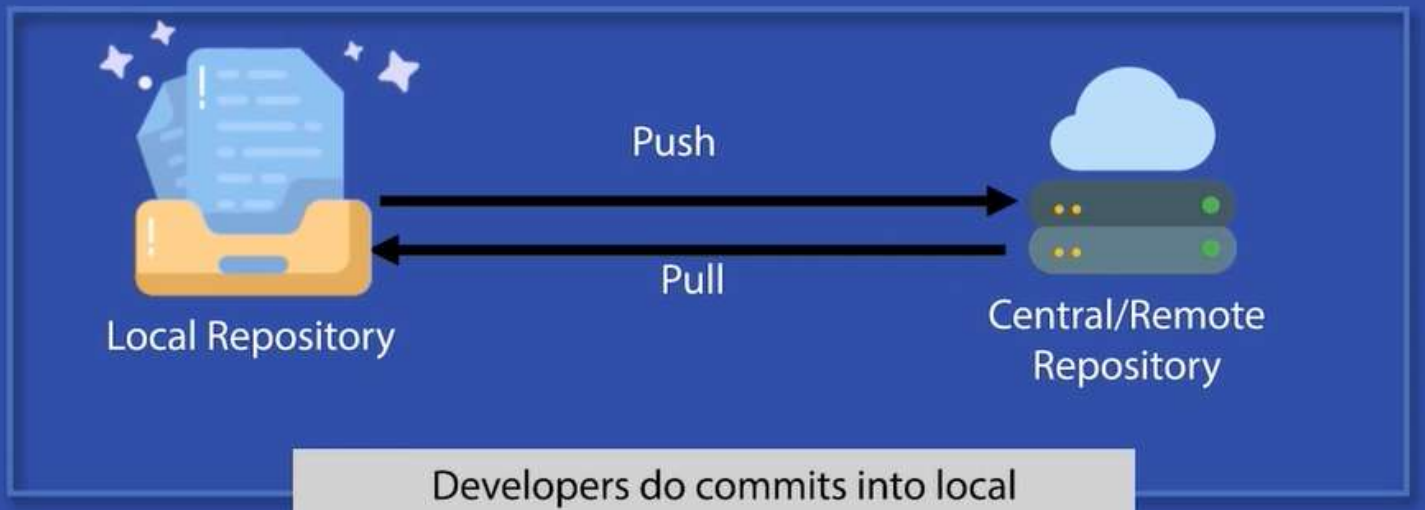
Lesson 3:

Git Repositories

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Repository Types



Developers do commits into local repository and moves the changes to remote/central repository

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Local Repository vs Remote Repository

Remote Repository

- Hosted onto a specific server
- Accessible to all the team members
- Not different from the local repository
- Clone/download remote to local
- All the activities like branches, commits, tags, merging, rebasing perform on local
- After changes, push changes to remote repository

Local Repository

- Installed on team members' computers
- Local repository has an associated working directory

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Local Repository vs Remote Repository

Local Repository

- Developers first checks actual work into the local repository
- Actual project work is carried out in local repository
- All changes must be made and committed locally

Remote Repository

- Does not have working directory
- Login to remote repository, browse source code, do changes and all activities are done at remote server
- Directly doing the changes to the remote repository by accessing remote

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Terminology - Origin

Makes referencing easier

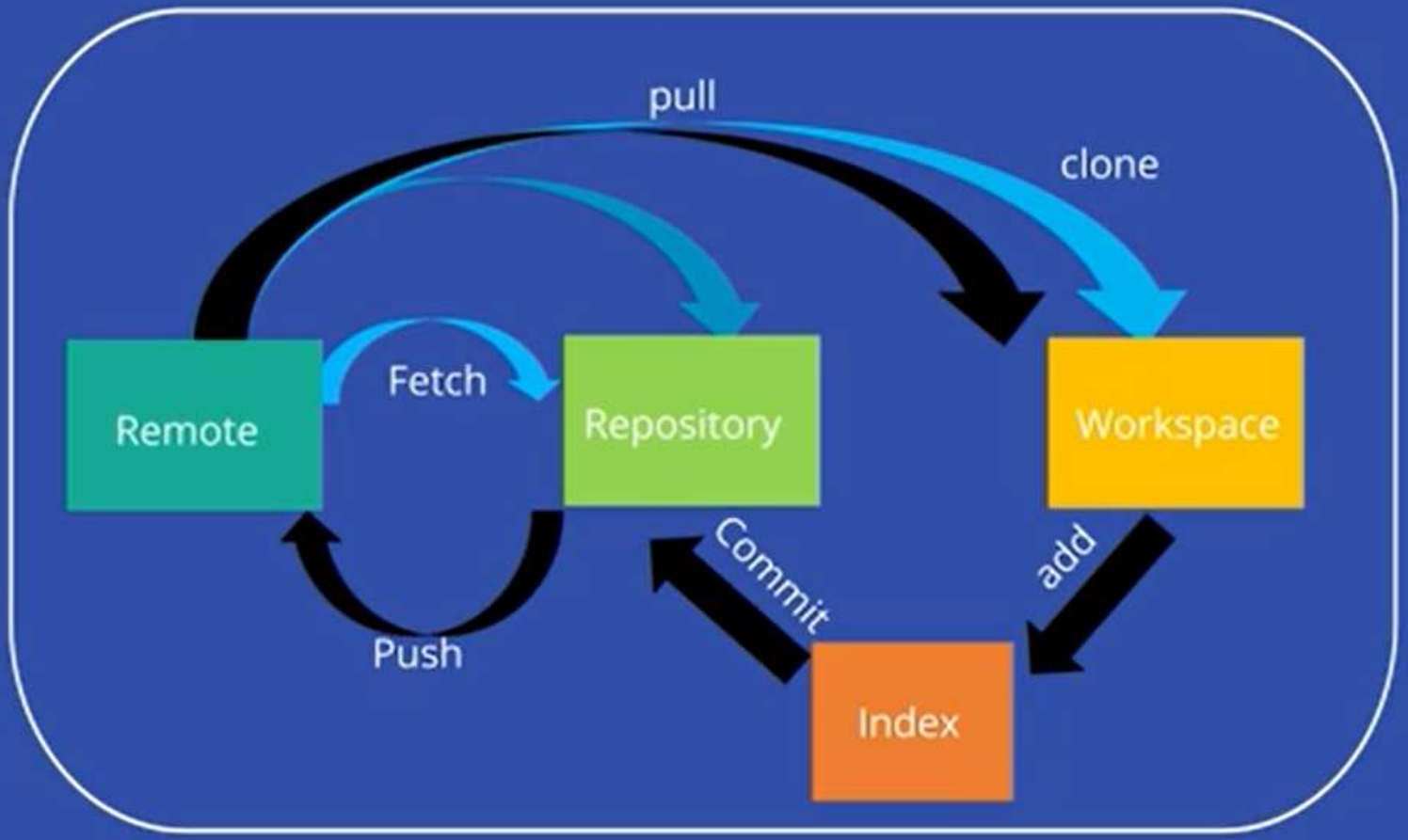
In Git, "origin" is a shorthand name for the remote repository

Used instead of original repository's URL

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

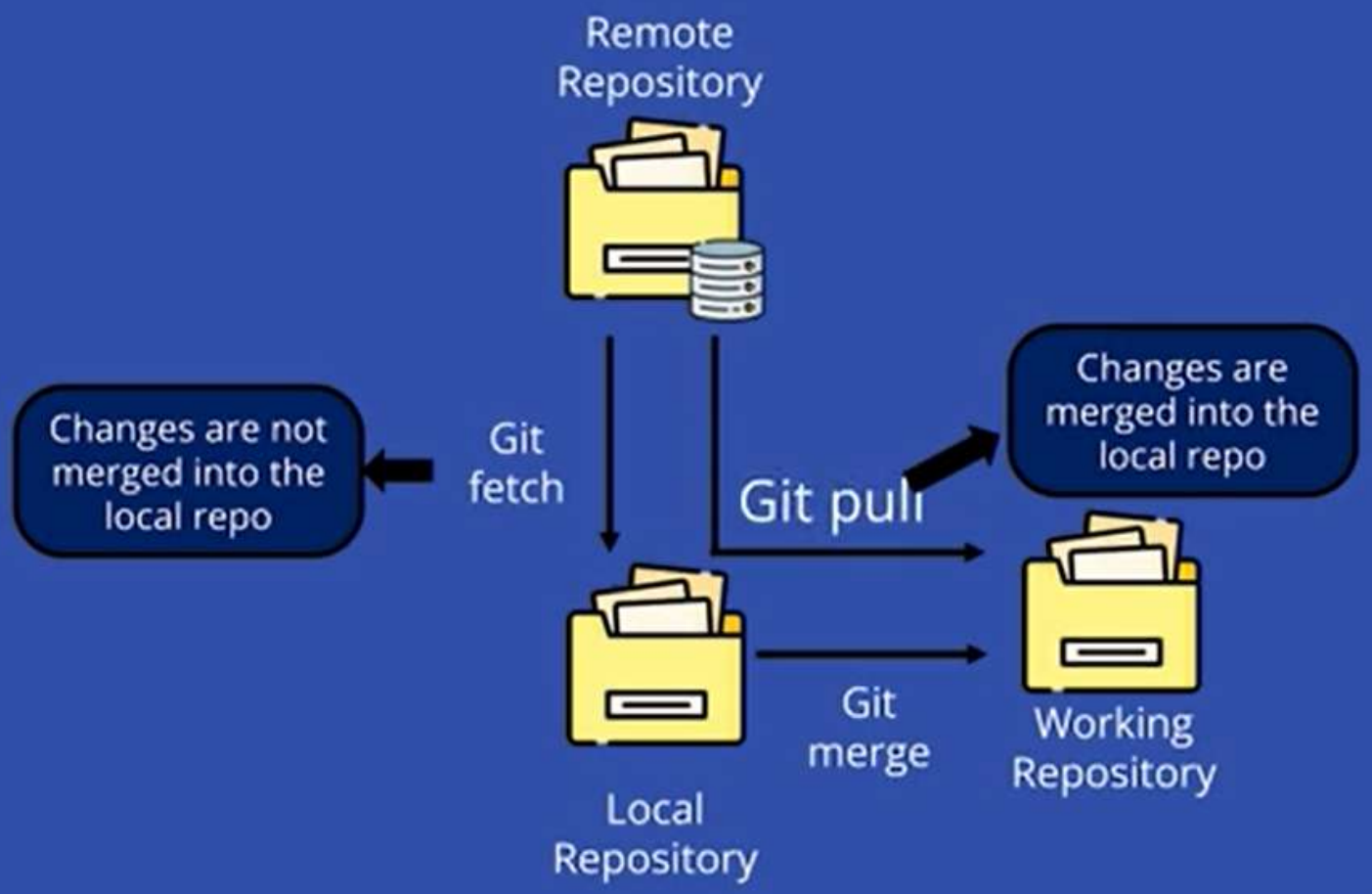
Difference between Clone, Fetch, Push and Pull



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Difference between "Git Pull" and "Git Merge"



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

GitHub

- Cloud-based service
- Graphical interface and CLI
- Provides access control
- Wikis and basic task management tools
- Lightweight and fast

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

GitHub Features

Effective Team Management

GitHub supports and organizes all team members on one page

Team can collaborate with each other

Accessibility

Can create numbers of public and private repositories

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

GitHub Features

Integration

Easy to Integrate with other DevOps tools like Jira, Jenkins

Improved Code Writing

Assists code review

Assists code approval

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

GitHub Features

Increased Code Safety

- Identifies vulnerabilities in the code
- Enables collaboration to secure end-to-end

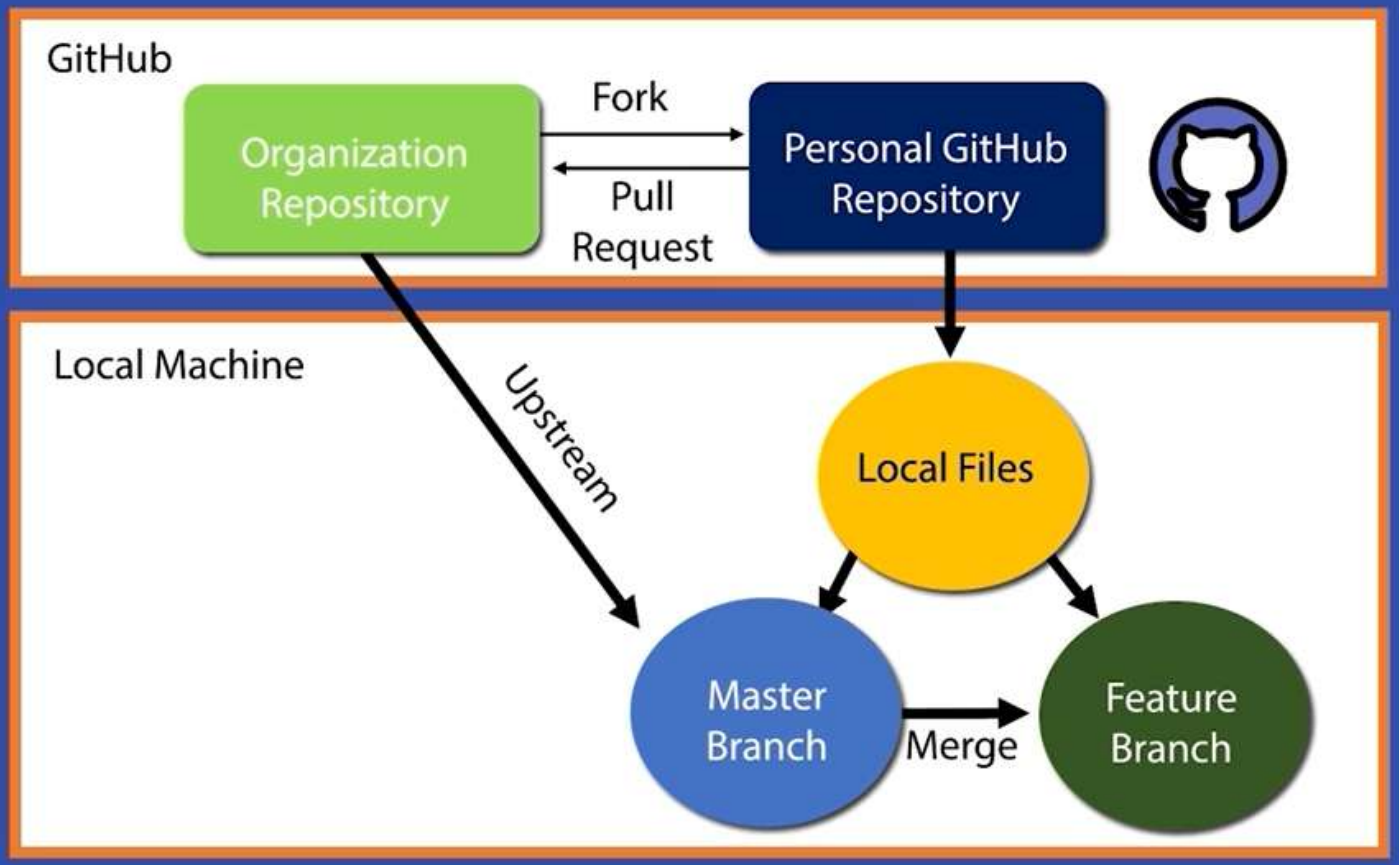
Easy Code Hosting

- Single location for all the code and documentation
- Easy GUI

© LearnKartS. All rights reserved.

LearnKartS
Certification Prep and Projects

GitHub Flow



Fork based mechanism

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Configure SSH Key

SSH Keys

To connect and authenticate to remote servers and services

With SSH keys, one can connect to GitHub without supplying username and personal access token at each visit

For password less communications, SSH can be used

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Setup SSH

Check for existing SSH keys

```
$ ls -al ~/.ssh
```

Public keys

id_rsa.pub
id_ecdsa.pub
id_ed25519.pub

Generate a new SSH keys

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

Algorithm (RSA, ed25519)

If keys exist, one can overwrite it

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Push and Fetch Branch to/from GitHub

```
$ git remote set-url origin git@github.com:<username>/< local repo name>.git
```

```
git push -u origin master
```

Pushes the local content on the master branch of the remote location

```
git push
```

```
git pull
```

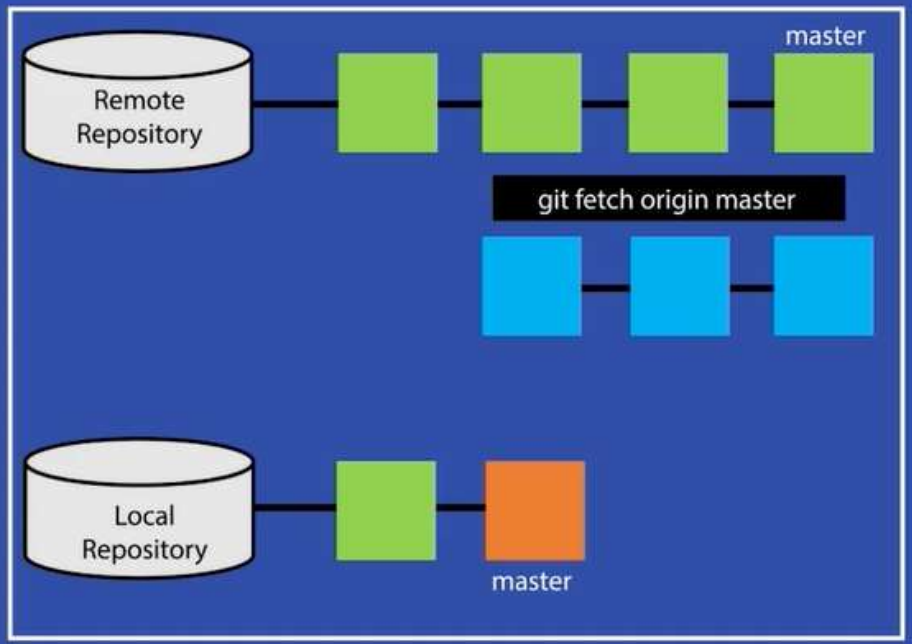
```
git fetch
```

By default, takes the current branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Understand Fetch



Git "fetch" downloads commits, objects and refs from another repository from remote repo

Not merged into local repo and preserved into the origin/master as a kind of a remote branch

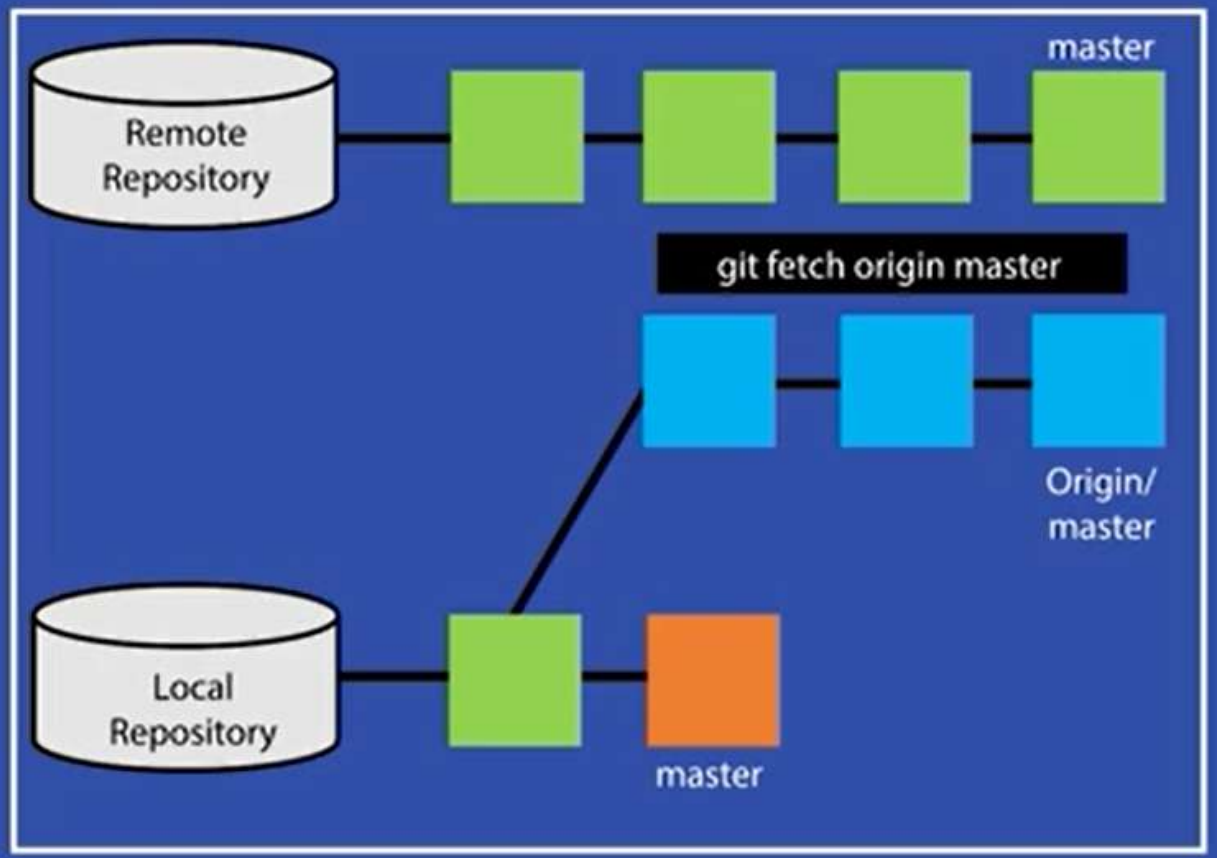
To merge into the local branch master, one have to do a pull activity

When working on to a local repository , its required to keep local repo up to date

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Understand Fetch




```
git fetch < repository URL>
```

```
git fetch <branch URL><branch name>
```

```
git fetch -all
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git vs GitHub

Git	GitHub
1 Software	1 Service
2 Command-line tool and GUI	2 Graphical User Interface (GUI)
3 Maintained by Linux	3 Maintained by Microsoft
4 Focused on version control and code sharing	4 Focused on centralized source code hosting platforms
5 No user management feature	5 Built-in user management feature

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

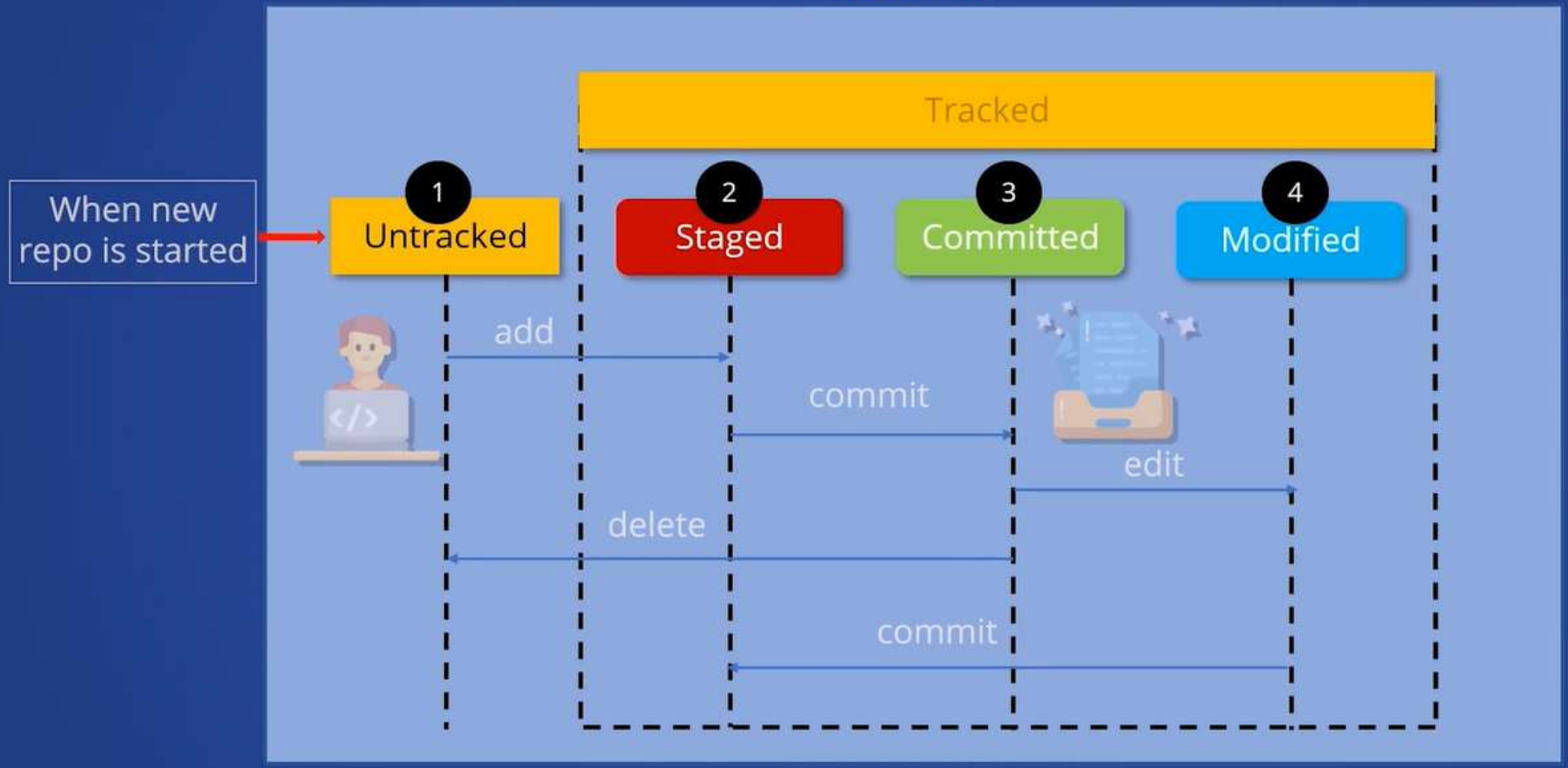
Lesson 4:

Git Life Cycle

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

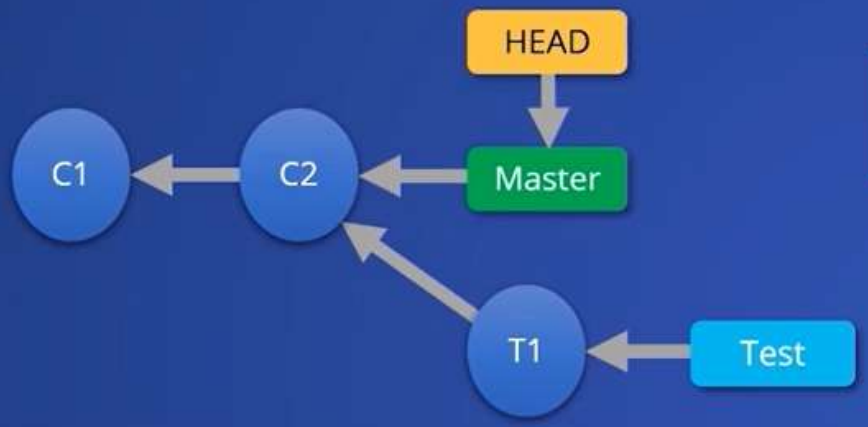
Flow of Git Files States



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

HEAD



Purpose	To point to the last commit

HEAD is a reference variable for the latest commit of the recent checkout branch

```
$ git show HEAD
commit 00fcce523529c269692271f406baf828a5112230 (HEAD -> edited, origin/edited)
Author: ImDwivedi1 <52317024+ImDwivedi1@users.noreply.github.com>
Date:   Sat Sep 14 16:46:19 2019 +0530

    Update Index.jsp

    you have not mentioned the contact info, so i have edited this file.

diff --git a/index.jsp b/index.jsp
index 371223b..236e116 100644
--- a/index.jsp
+++ b/index.jsp
@@ -27,4 +27,6 @@
```

```
$ git show HEAD
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

HEAD

HEAD is a pointer → Points to the latest commit

* As new commits occur, HEAD keeps on moving to point to new commits
* If a specific version is checked out, HEAD will move to point to the same
* HEAD can move forward and backward by doing checkout

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Know HEAD Position

To check HEAD

```
cat .git/HEAD
```

Output

ref: refs/heads/master

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Check File Status

To see the state of files

```
git status
```

```
git status -s
```

* Tracked files
* Un-tracked files
* Branch information
* Ready-for-commit files
* Current active branch
* Modified files

Provides summary

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Check File Status

Commands to see how the file differs from the repository version

To see details differences

```
git diff HEAD
```

To see summary differences

```
git diff --summary
```

To compare changes with latest commit

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

To Move File to Staging

To add any files

```
git add <file name>
```

```
git add -u
```

To add multiple files

```
git add file1 file2 file3 and so on
```

To add all the files

```
git add -A
```

```
git add --all
```

git add does not affect the remote repository

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Move File from Staging to Local Repository

To preserve the changes

```
git commit
```

```
git commit-m "message"
```

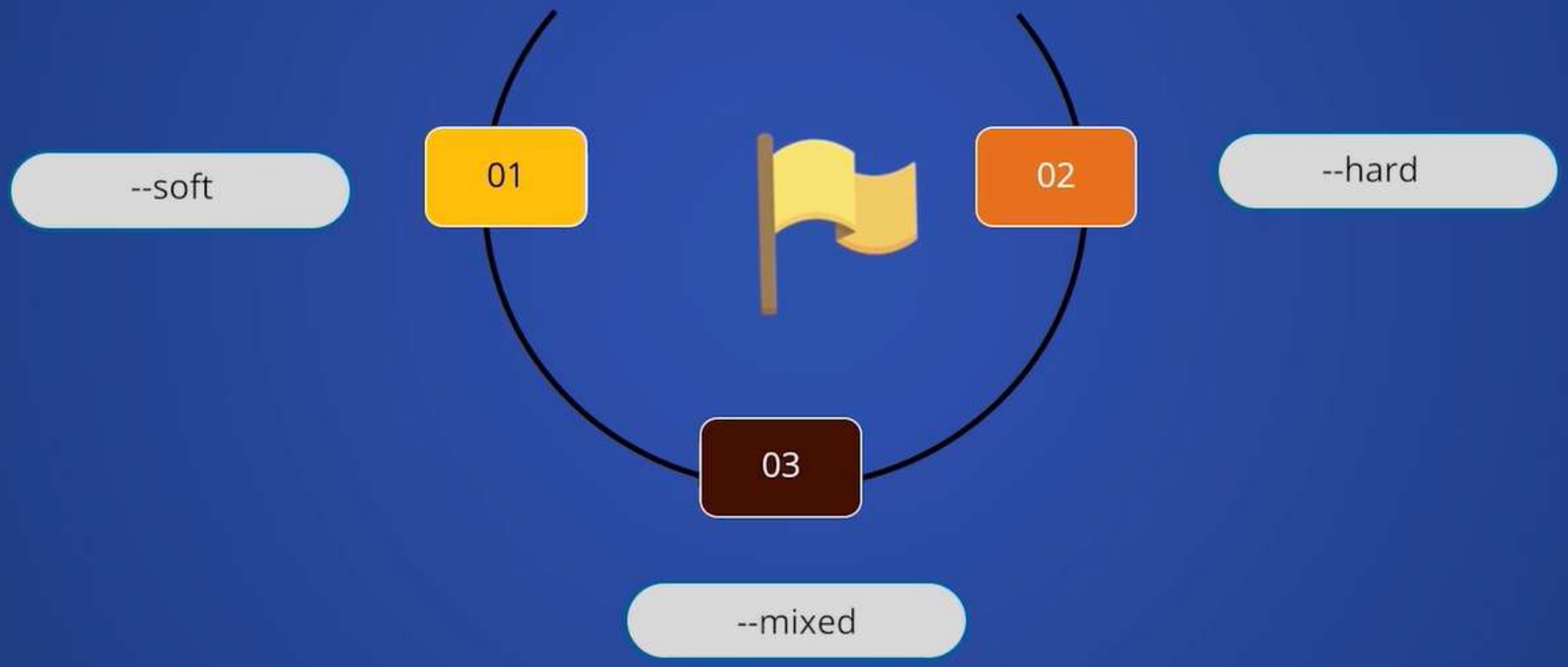
To publish the changes to remote repository

```
git push origin <branchName>
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Reset Flags

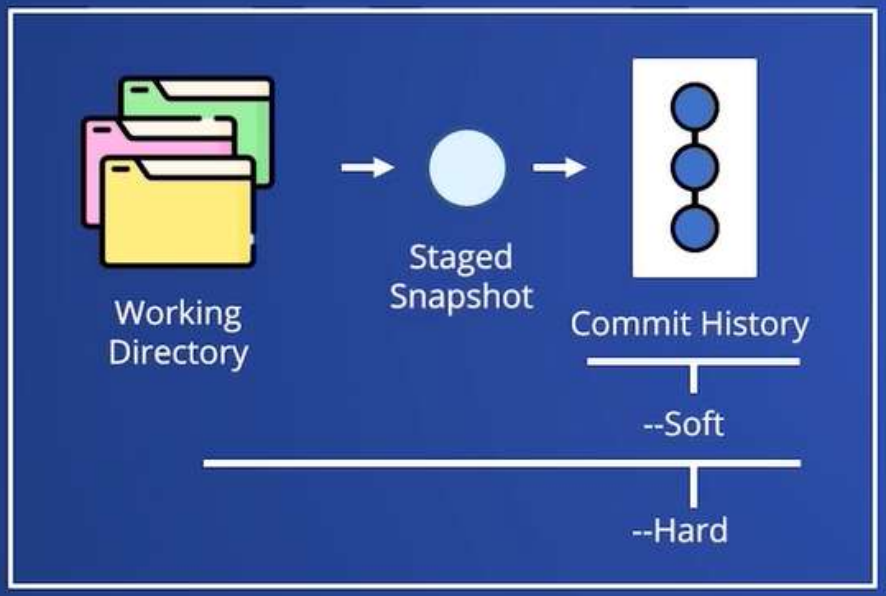


© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Soft Reset

--soft



Staging Index and the working directory are left untouched

```
Git reset--soft <commit-id>
```

* Any file which is in the staging area will NOT be rollback

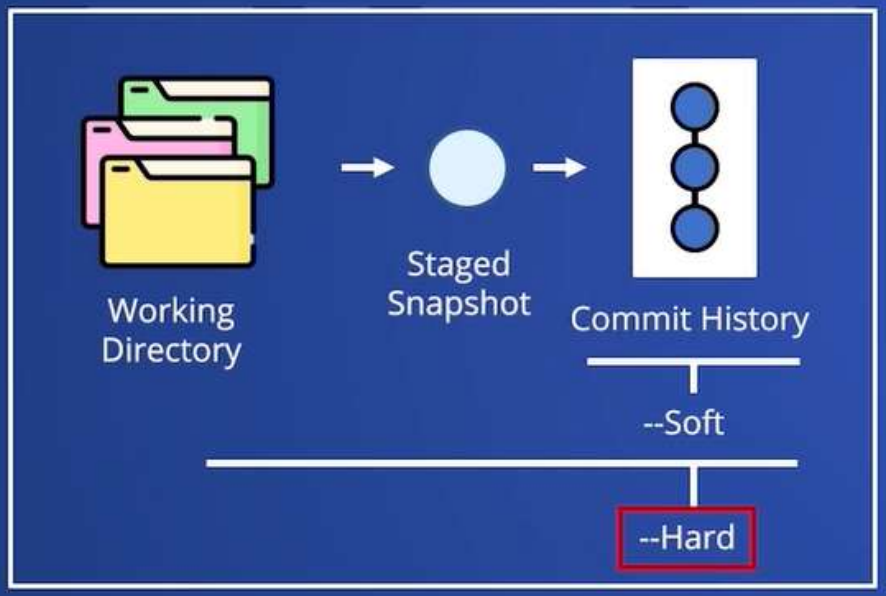
* Only removes the file from local repository

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Hard Reset

--hard



Staging index and working directory are reset to match that of the specified commit

```
git reset --hard
```

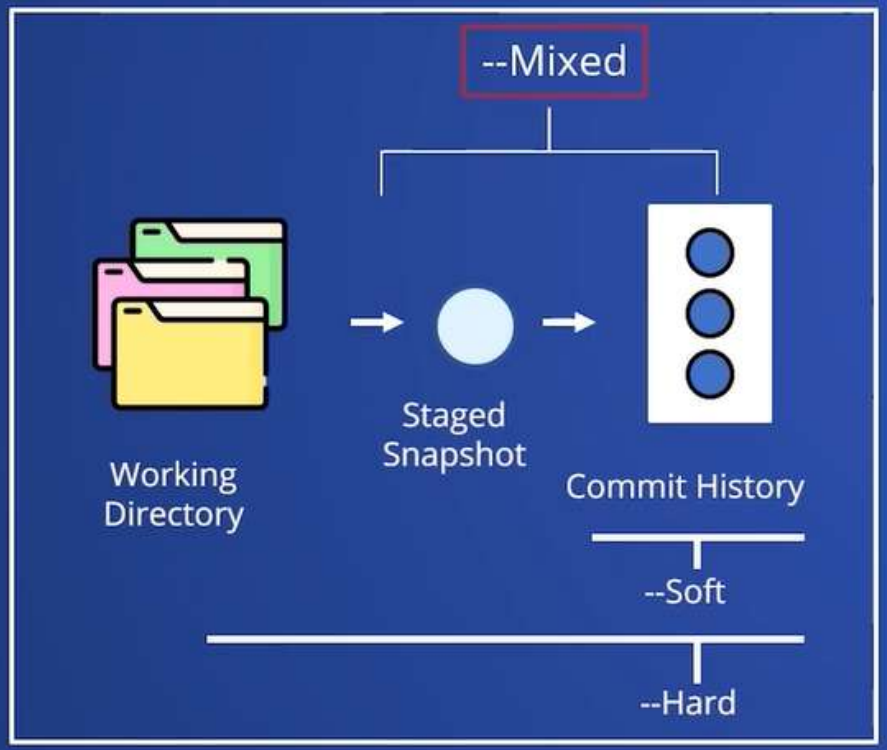
* All the changes made are completely removed
* Moves HEAD to backward [one or more commits back]
* Use if changes needs to be eliminated from repository

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Mixed Reset

--mixed



Mixed = Hard + Soft

* Mixed reset does NOT reset the working directory
* All files in the directory are available, but the changes in staging area are removed
* Example scenario : When bad commits happened, but one wants to keep all the work so that it can be recommitted

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Revert All Changes to a File

```
git checkout <fileName>
```

Discards all changes made in a file

Check modifications in a file

```
git diff <fileName>
```

Check the differences in a file before and after the modifications

Discard all changes on the branch

```
git reset --hard
```

Reset all changes made on the branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Reset Comparison

	HEAD	Staging/ Index	Working directory	Working Directory Safe?
Commit Level				
`reset -soft [commit]`	REF	NO	NO	YES
`reset [commit]`	REF	YES	NO	YES
`reset -hard [commit]`	REF	YES	YES	NO
`checkout <commit>`	HEAD	YES	YES	YES
File Level				
`reset [commit] <paths>`	NO	YES	NO	YES
`checkout [commit] <paths>`	NO	YES	YES	NO

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Understanding "^ (caret)" and "~ (tilde)"

Used to identify specific commit

HEAD~n

n commits before HEAD

nth generation ancestor of HEAD

HEAD~ /HEAD~1

Commit's first parent

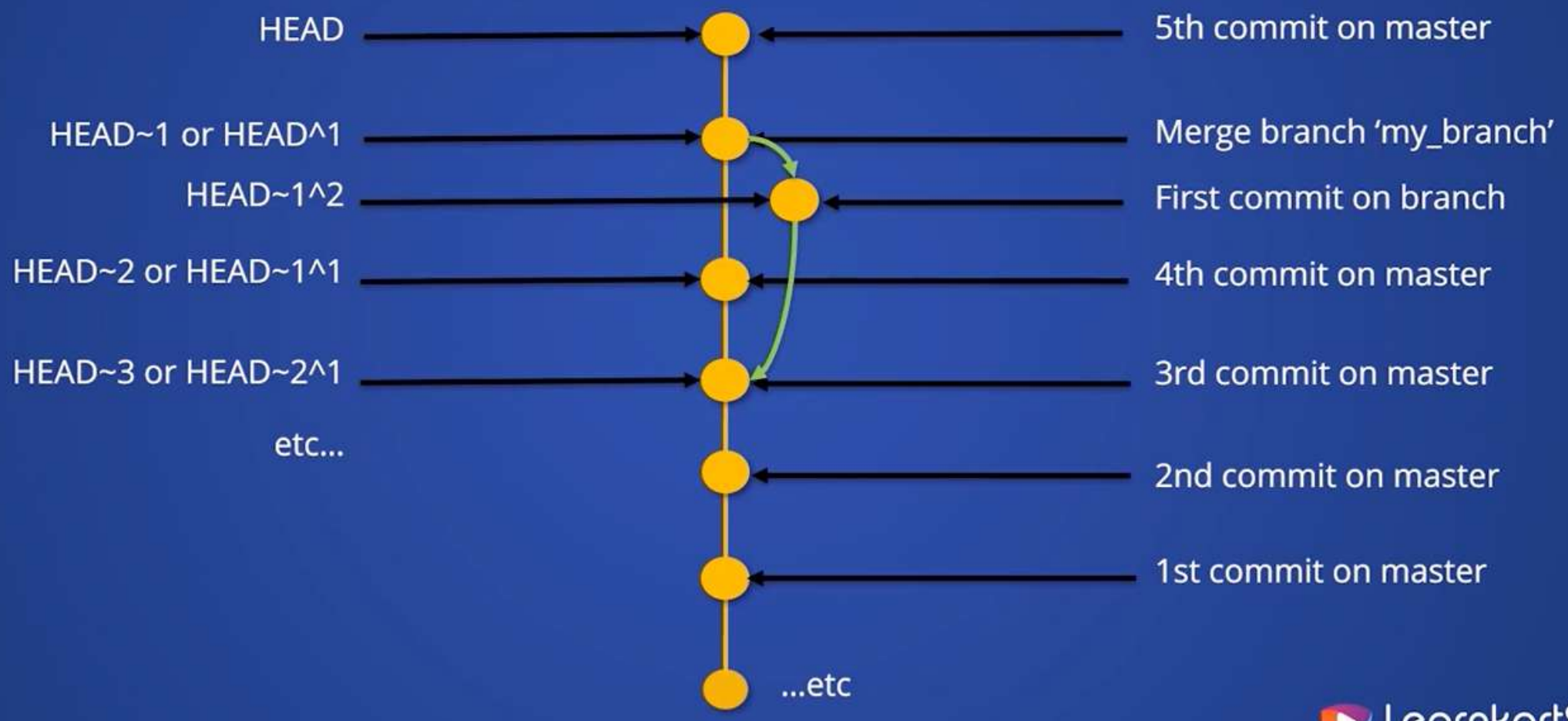
HEAD~2

Commit's first parent's first parent (2 commits backward)

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Understanding "^ (caret)" and "~ (tilde)" - Example



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Objects

- Simple key-value data store
- Maintained at remote and local repository
- Any type of content can be inserted in a Git repository

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Objects

All files are committed in form of git objects in .git folder

Repository

".git" folder should be secured

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Objects

blob	Contents of a file
tree	A directory listing (of **blobs** and **trees**)
commit	Snapshot of the working tree

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Objects - Blob

blob

* IIn git, the contents of files are stored in objects called blobs, (Binary Large Objects)
* Secured
* Every blob in git is identified by its SHA-1 hash

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Objects - Blob

File

Files also contain meta-data

Example.txt

Contains all details

Blob

Blob doesn't register its creation date, its name, or anything but its contents

Blobs contents—binary streams of data

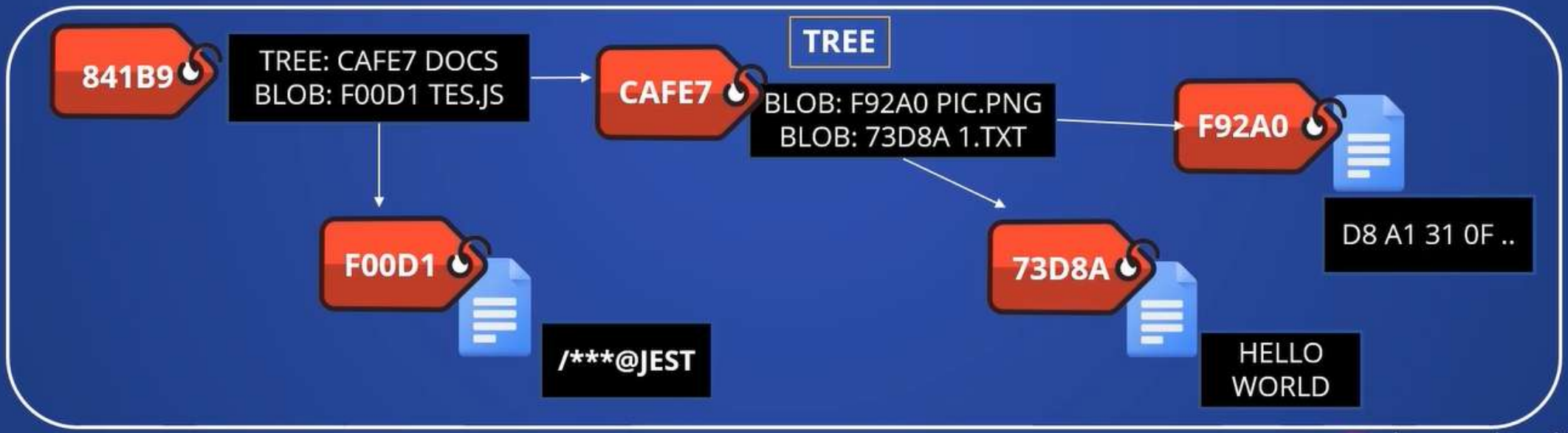
© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Objects - Tree

tree

* Directory structure
* Refers to blobs as well as other trees
* Trees are identified by SHA-1 hashes



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Objects - Commit

commit

* Commit is a snapshot

* HEAD points to the latest commit in branch

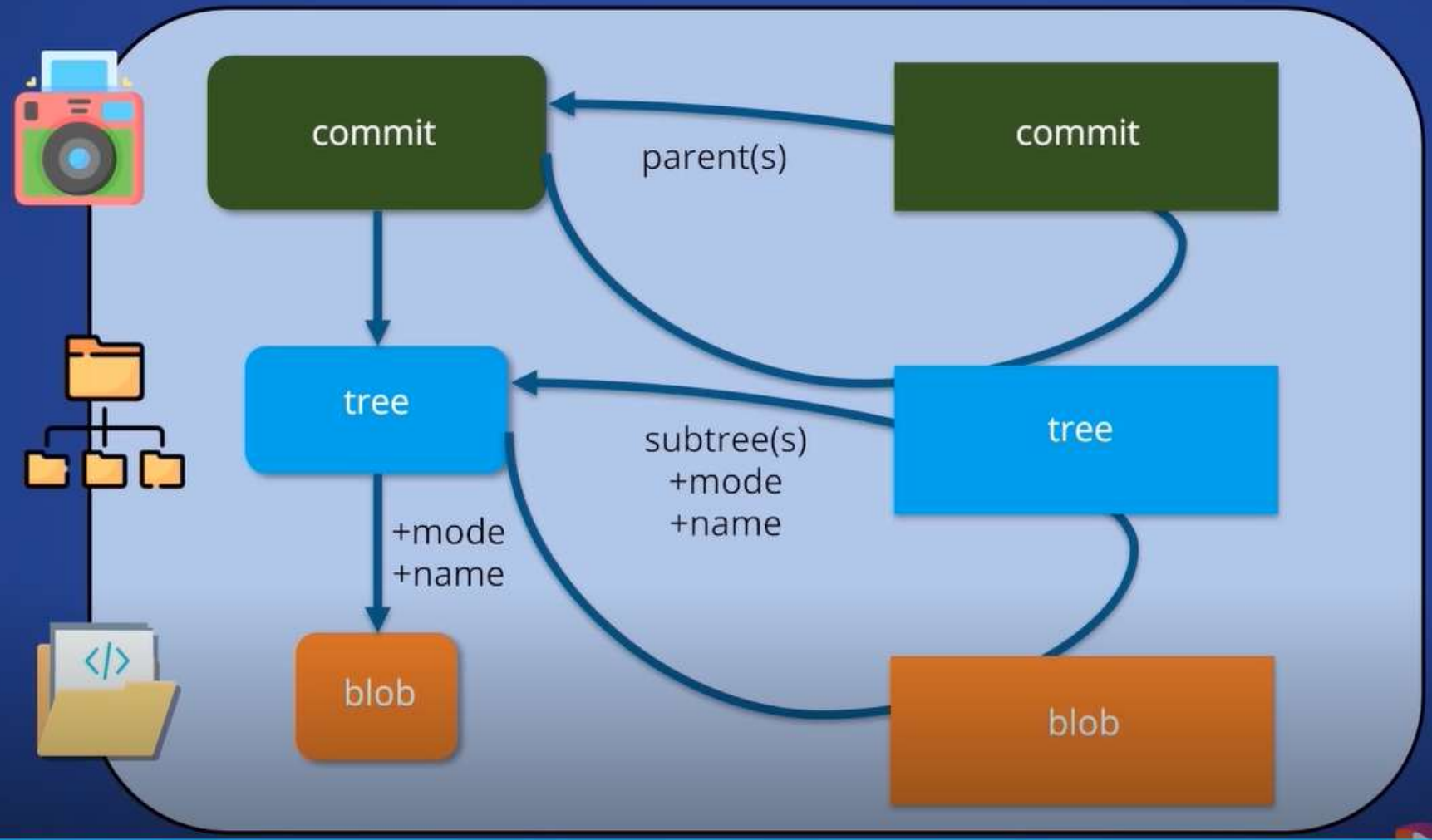
* Commit objects are also identified by SHA-1 hashes

* Every commit holds the entire snapshot, not just diffs from the previous commit(s

© LearnKartS. All rights reserved.

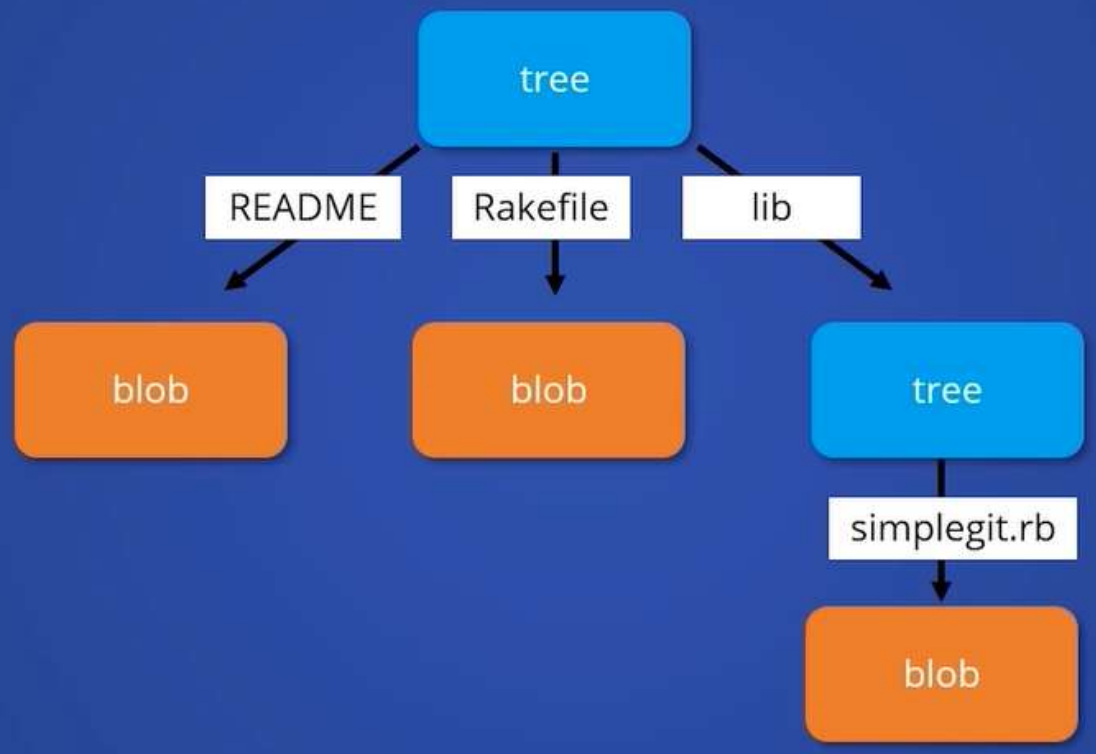
LearnkartS
Certification Prep and Projects

Git Objects



© LearnKartS. All rights reserved.
LearnkartS
Certification Prep and Projects

Git Data Storage



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Lesson 5:

Git Branching and Merging

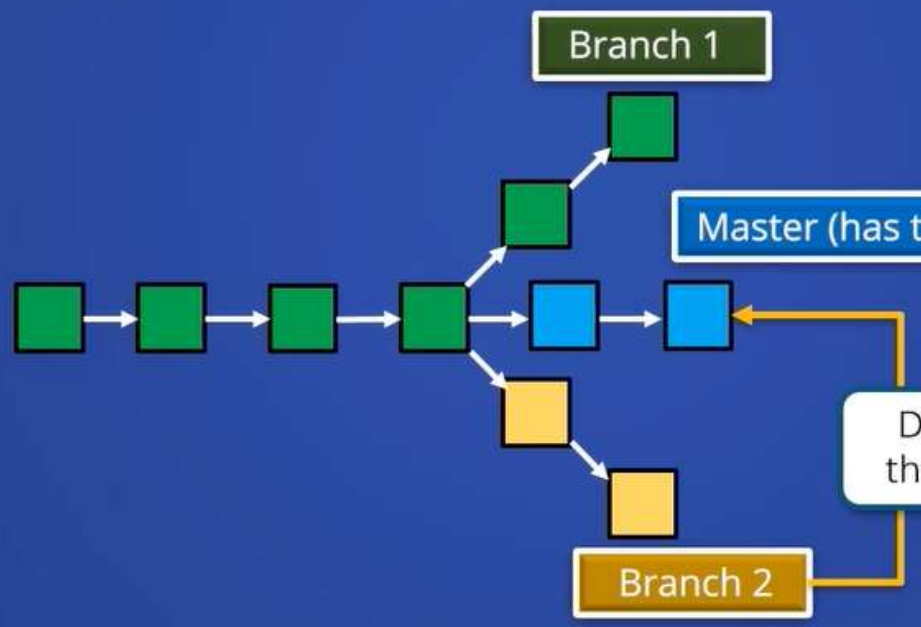
© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Branch

Independent development line

Branch is a basic approach followed in any version control system



Default branch which gets created at the time of git initialization

Do the changes in branch and then merge it to master branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Need for Branch

Most of the development projects have two mainstream branches

Master branch
Used for keeping the production ready source code
[which we don't want to change]

Final source code is deployed from the master branch

Dev branch
Used for development and testing purposes
[bug fixes and alteration]

New changes for feature development or testing are done in the dev branch, and then it is merged into the master branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Branching Main Operations

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Check Branch

```
$ git branch
```

```
$ git branch --list
```

Show the available branches

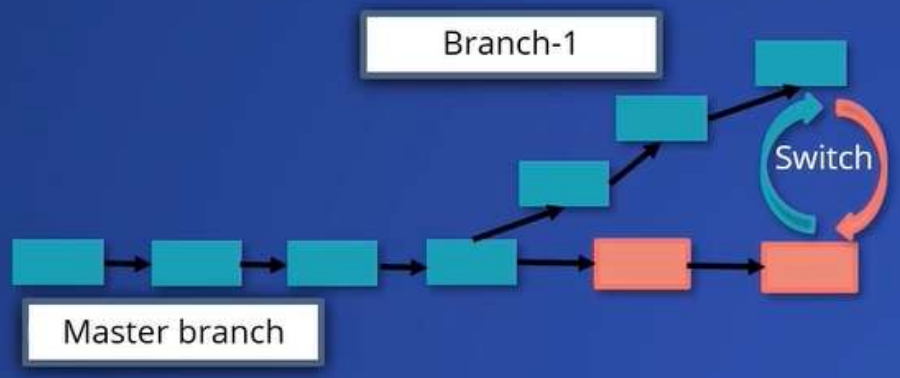
Show the current branch

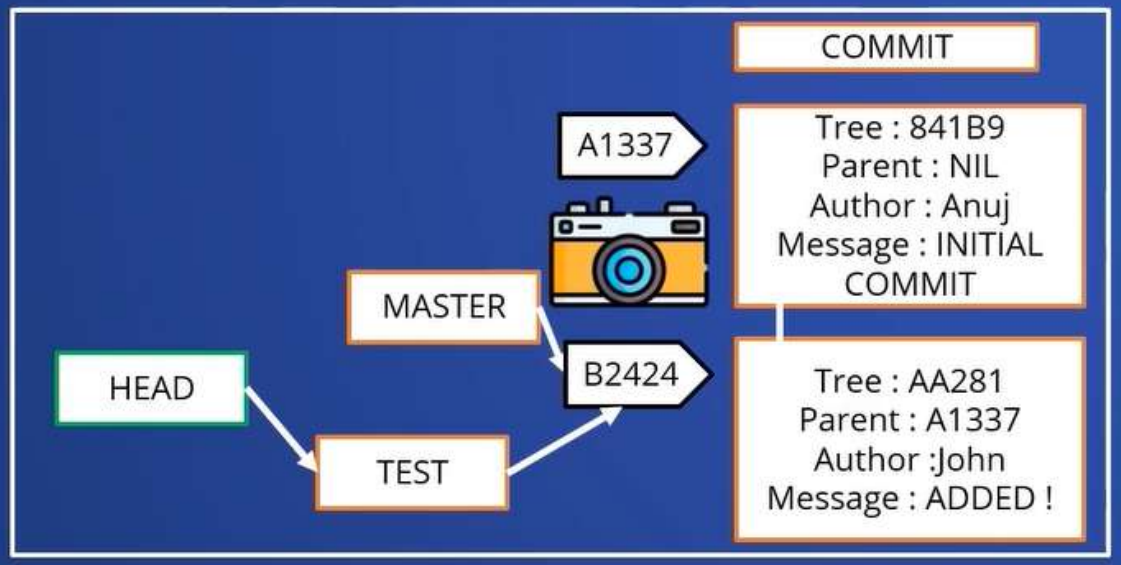
```
Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (master)
$ git branch -l
* master

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (master)
$ git branch
* master

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (master)
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Switch Branches





Switching/Checking Out branch

```
$ git checkout <other-branch>
```

```
$ git checkout test
```

HEAD points to latest commit on test branch now

```
$ $ git branch -m master
```

Switch back to master branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Create and Switch Branch

```
$ git checkout -b <new branch name>
```

Create a new branch
+
Switch to the new branch

```
$ git checkout -b feature
Switched to a new branch 'feature'

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (feature)
$ git branch -l
  develop
* feature
  master
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Rename a Local Branch

```
$ git branch -m <new-name>
```

Renames the current checked out branch

```
$ git branch -m <old-name> <new-name>
```

Renames the specific branch with the new name

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Rename a Remote Branch

Renaming a Remote Branch

1. Remove the old branch from our local repository

```
$ git push origin --delete <old-name>
```

2. Push the new local branch one up

```
$ git push -u origin <new-name>
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Track Branches

Tracking a Remote Branch

```
$ git branch -u
```

```
$ git branch -vv
```

Provides different commits and related information

```
$ git push -u origin <local-branch>
```

Push branch to remote repository and enable tracking

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Track Branches

Tracking a Remote Branch

```
$ git branch --track <new-branch> origin/<base-branch
```

Helps create a proper track using a branch

```
$ git checkout --track origin/<base-branch>s
```

Helps a tracking and communicate with a branch

```
$ git branch -vv
```

Helps to visualize tracking related information

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Delete a Local Branch

2 methods to delete

'-d'

Safe option : If the branch has unmerged commits it will not be deleted

```
$ git branch -d my-new-branch
```

'-D'

To forcefully delete

```
$ git branch -D my-new-branch
```

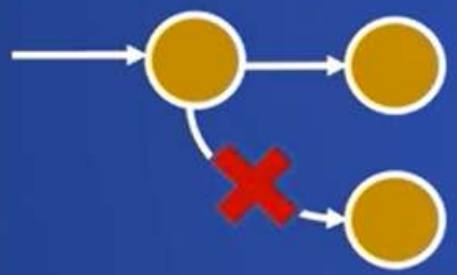
```
$ git branch -D feature
Deleted branch feature (was a3f18f1).

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (develop)
$ git branch -l
* develop
  master

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (develop)
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Delete a Remote Branch

```
$git push origin --delete <branch name>
```



This command will push the commit and it will delete the remote branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Difference between Branches

Compare branches

```
$git diff <branch name>
```

```
$ git branch -l
  develop
* feature
  master

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (feature)
$ git diff develop

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (feature)
$ git log --oneline
a3f18f1 (HEAD -> feature, master, develop) First Commit

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (feature)
$ git checkout develop
Switched to branch 'develop'

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (develop)
$ git log --oneline
a3f18f1 (HEAD -> develop, master, feature) First Commit

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (develop)
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Pushing the Local Branch

When creating a new branch, it will be on local machine and will not be accessible to teammates until it is pushed to the remote repository

To collaborate or share push local branch to remote repository

```
git push origin <branchName>
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Commit Changes to Remote Repository

List of branches

```
git branch -v
```

Push the current branch

```
git push <remote>
```

Push a specific branch

```
git push <remote> <branch name>
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Branches Types

Master Branch
Contains the production ready source code

Feature Branch
Perform development and add features

Develop/ Integration Branch
Multiple features are integrated

Release Branch
A finalized branch where merging occurs

Hotfix Branch
A temporary branch for source code debugging

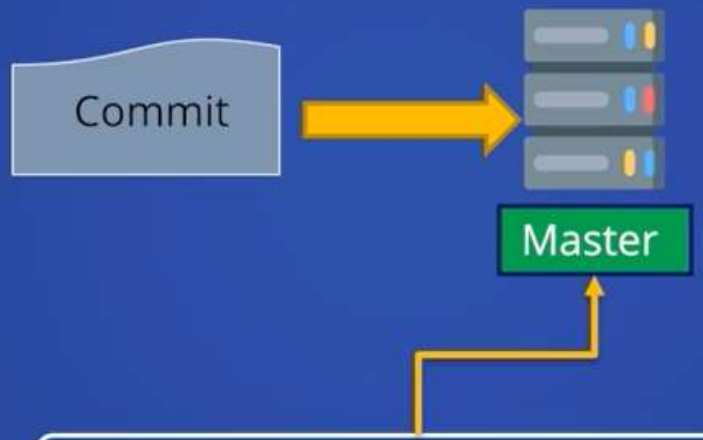
© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Master Branch

Git automatically creates a master branch when first commit in a repository is made

All commits by default will be done in master unless a new branch is created explicitly



Developer must **NOT** make changes directly in master branch and rather should use feature branch

Only verified code should enter the master branch

© LearnKartS. All rights reserved.

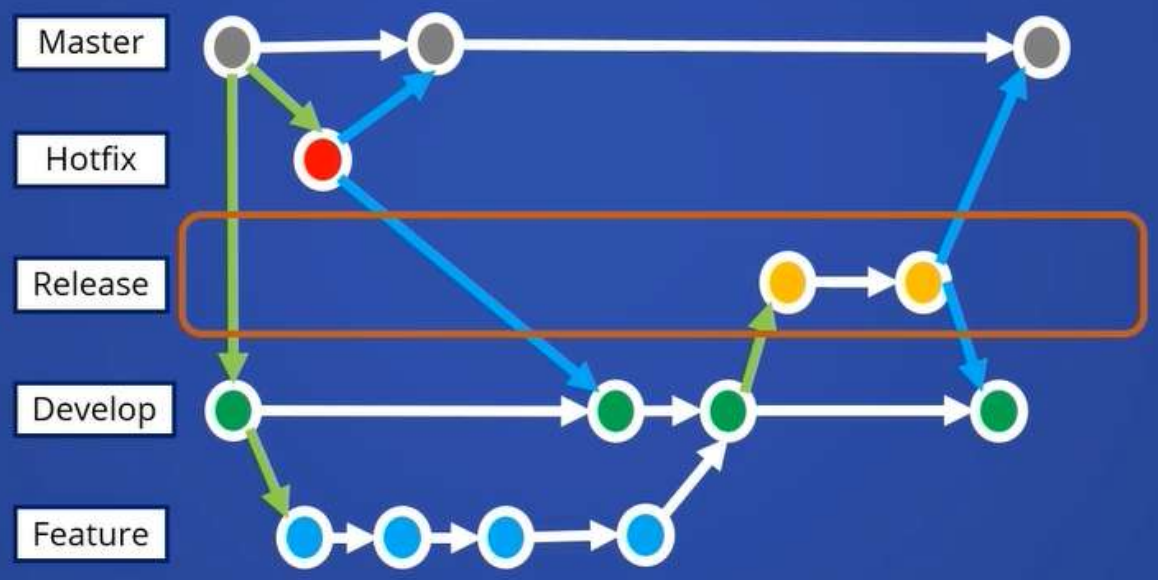
LearnkartS
Certification Prep and Projects

Release Branch

Create a release branch once changes in a develop branch are done

Merged with the master branch after final check

Close to production release, the release branch is branched out from the develop branch



Release branch helps to make sure that no further commits are done

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Release Branch

Release branch name usually begins with the '**release-**' prefix

Release branch shall address

- Bug fixes
- Release-related issues

Develop

Release

Only when final code is ready, release branch is created

Developers continue to work on develop/integration branch with new features without interrupting the release process

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Hotfix Branch

Used to add a critical fix to the production codebase quickly

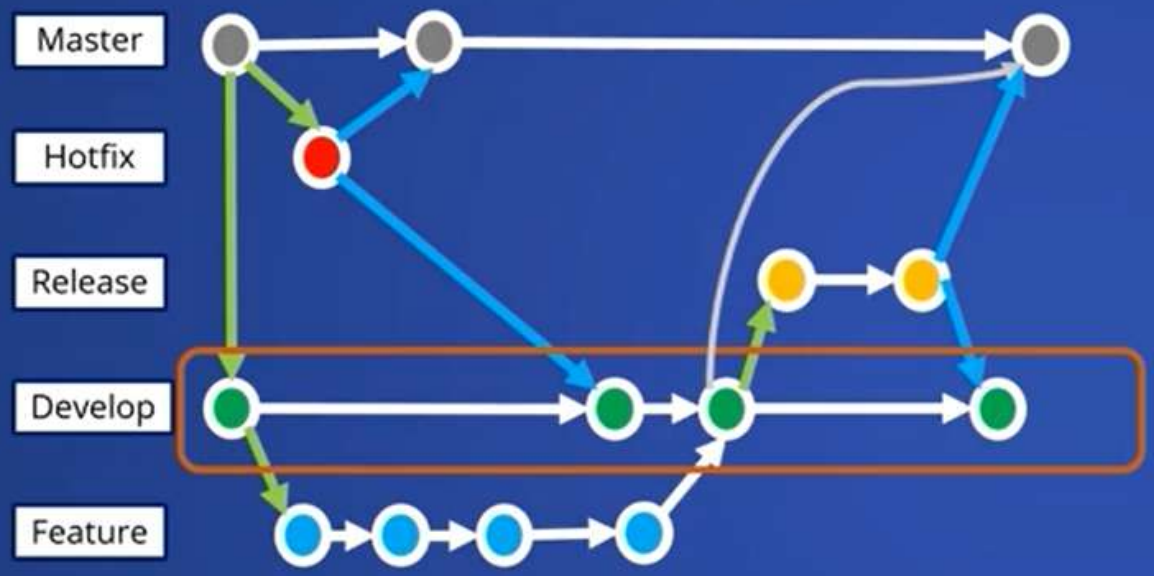
Used if some issues happen in the production environment and want to avoid the full cycle of development

Hotfix branch is merged with master and back merged with development branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Develop/Integration Branch



Used for development

All feature branches will be merged to the integration branch

A develop/integration branch should always be kept stable

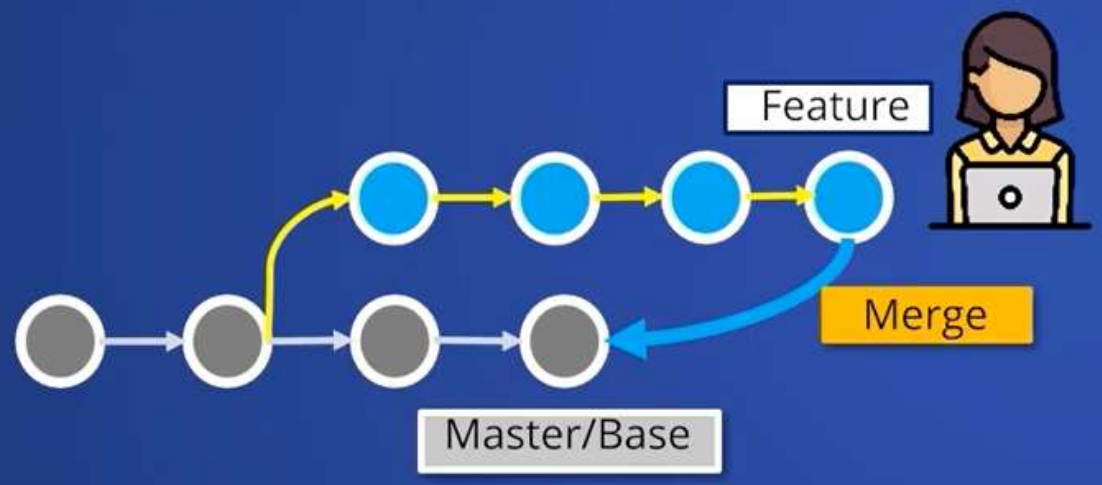
Integration branch can be directly merged to master or can have a release branch in between

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Merge

Process of bringing the changes from one branch to another



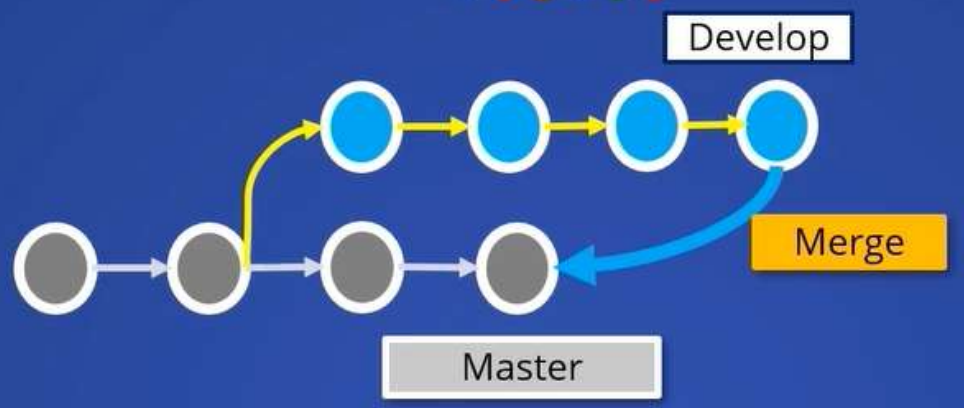
Multiple developers work on local repository on specific feature branch

After making the changes merge feature branches to the master branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Merge Scenario – Merge Feature Branch to Master Branch




```
$git log
```

Check the available logs

```
$ git checkout master
```

Make master branch as current branch

```
$ git merge develop
```

Merge the branch into master branch

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Merge Types

Two-way Merge

Three-way Merge

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Two-Way Merge

Where the merge involves just two snapshots

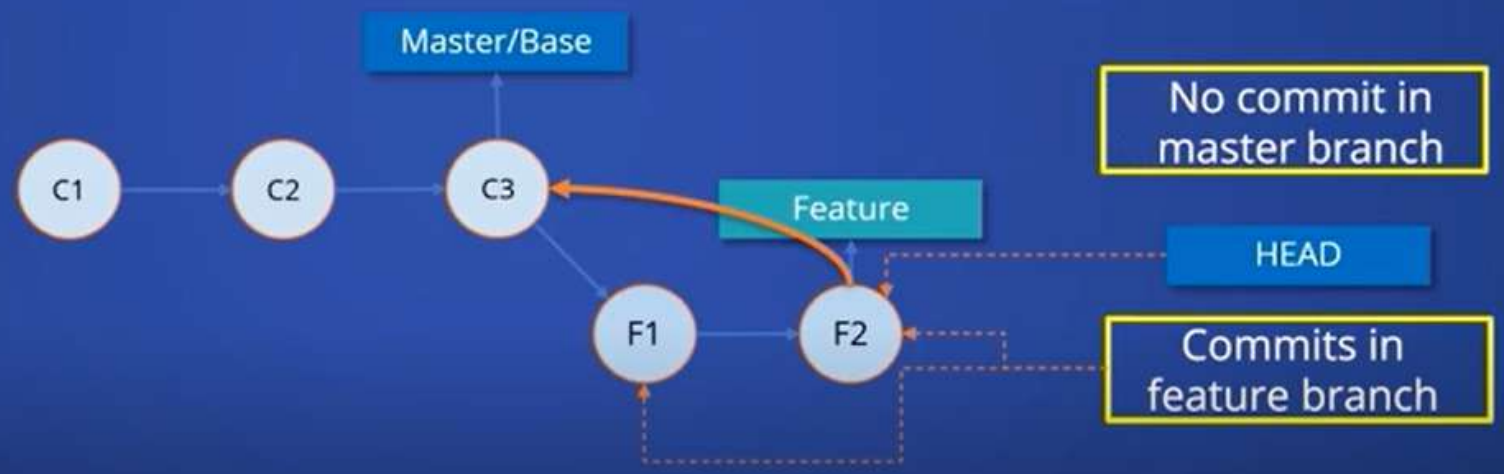
To merge feature branch to master branch

Since feature branch is new branch, the master branch is no longer committed

2 snapshots – F2, C3

F2 and C3 snapshots will be compared

Fast-forward merge



No commit in master branch

Commits in feature branch

© LearnkartS. All rights reserved.

LearnkartS
Certification Prep and Projects

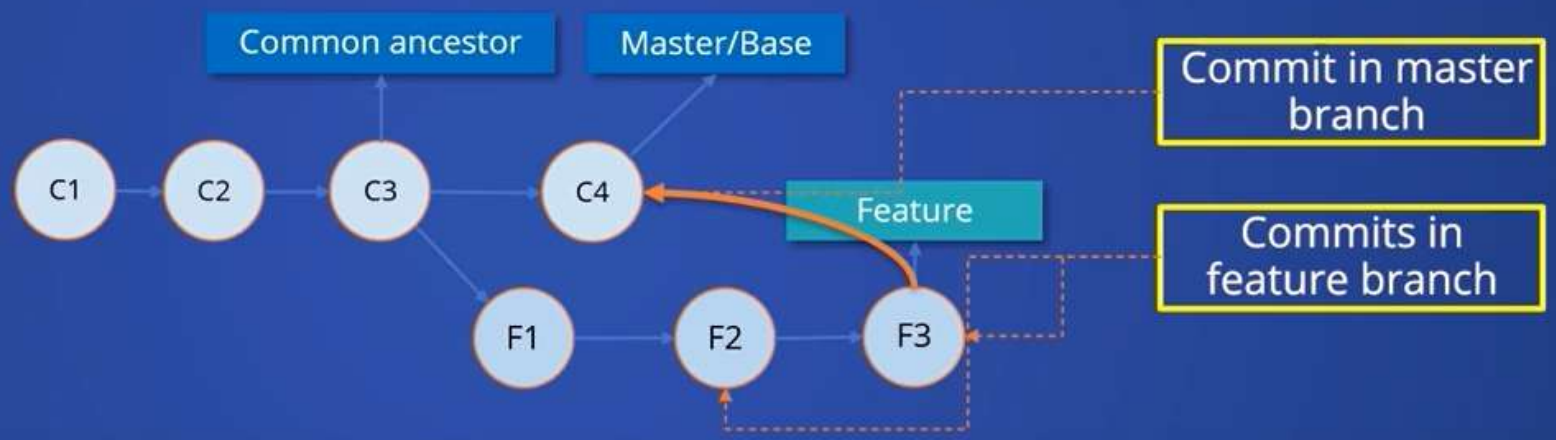
01:17 / 02:11

Three-Way Merge

Where the merge involves three snapshots

3 snapshots – C4, F2, F3

C4 and F3 snapshots will be compared against C3



More chances of conflict

© LearnKartS. All rights reserved.

Conflict Scenario

If two or more team members work in the same file simultaneously

Overlap of work may result in a conflict at the time of merging

During the integration of different feature branches into the develop branch may cause conflicts

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Resolve Merge Conflict

A developer will see the following error during a git merge in the case of merge conflict

MERGE CONFLICT

```
$ git merge develop
Auto-merging sample.java
CONFLICT (content): Merge conflict in sample.java
Automatic merge failed; fix conflicts and then commit the result.

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   sample.java

no changes added to commit (use "git add" and/or "git commit -a")

Lenovo@AnujSharma MINGW64 ~/Desktop/sample_repo (master|MERGING)
```

Conflicts resolution time

A minute to many days

[Depending on number of files in which conflict is observed]

Recommendation

Synchronize local repository with remote (PUSH/PULL) frequently

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Resolve Merge Conflict

Step 1

Open the conflicted file and make the necessary changes

Step 2

Use **git add** command to stage the newly merged content

Step 3

Perform a new commit using **git commit** command

Git creates a new merge commit to finally complete the merge

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Commands to Resolve Conflicts

Command	Description
git log --merge	To generate a list of conflicting commitments
git diff	To identify differences between repositories or files
git checkout	To reverse the file changes or to change branches
git reset --mixed	To undo changes in the working directory and screen
git merge --abort	To exit the merge and return to the state before the merging starts
git reset	To reset the disputed files

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Merge Strategies

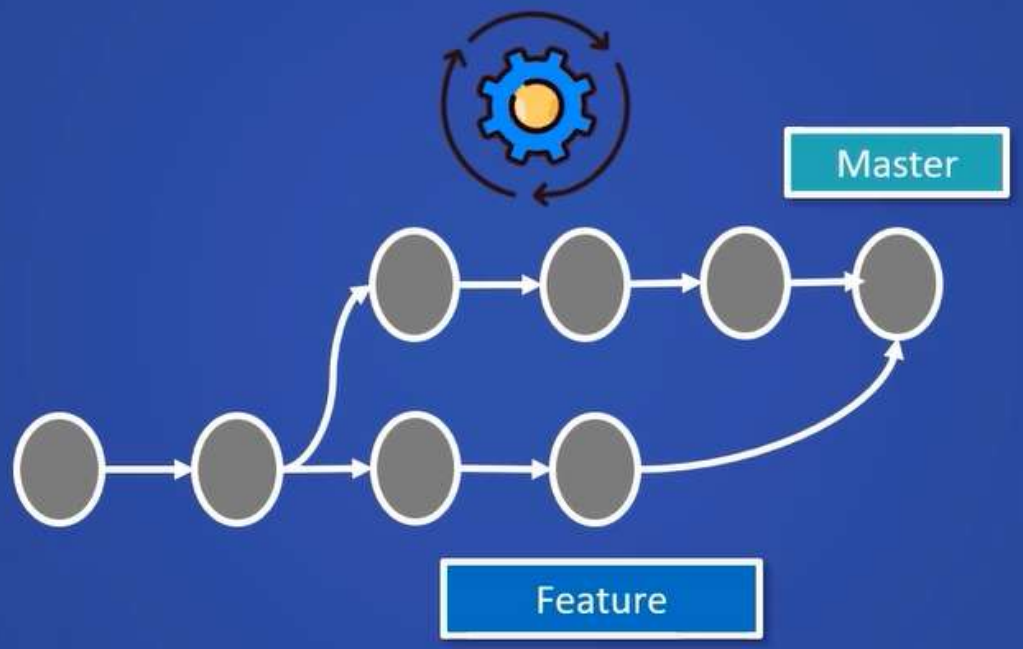
© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Recursive

The recursive strategy can detect and manage mergers

Default strategy




```
$ git merge -s recursive branch1 branch2
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

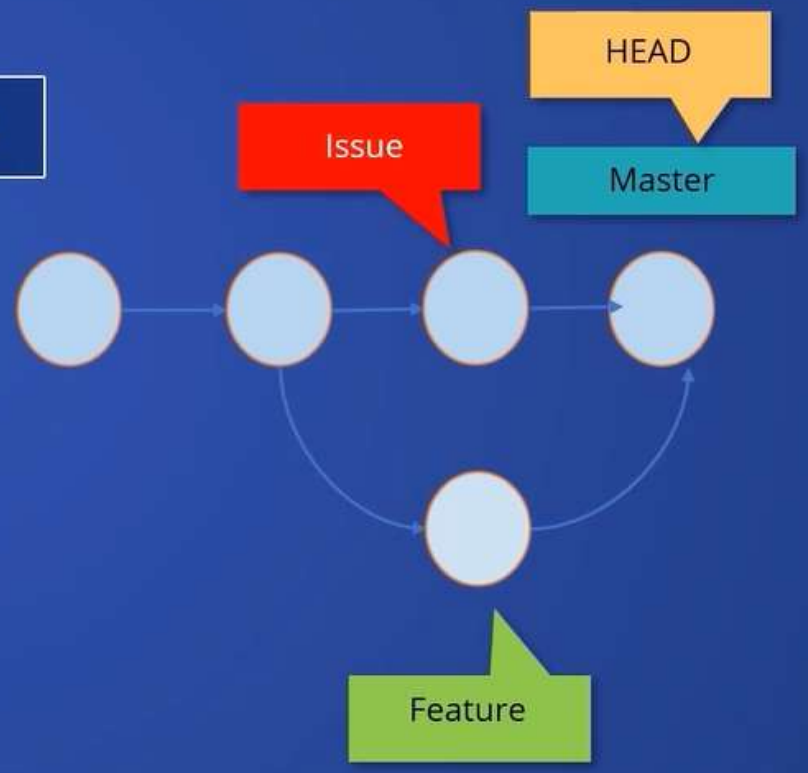
Resolve

A 3-way merge to resolve branches

```
$ git merge -s resolve branch1 branch2
```

Two HEADs can be resolved through a 3-way merge algorithm

Provides very fast resolution



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

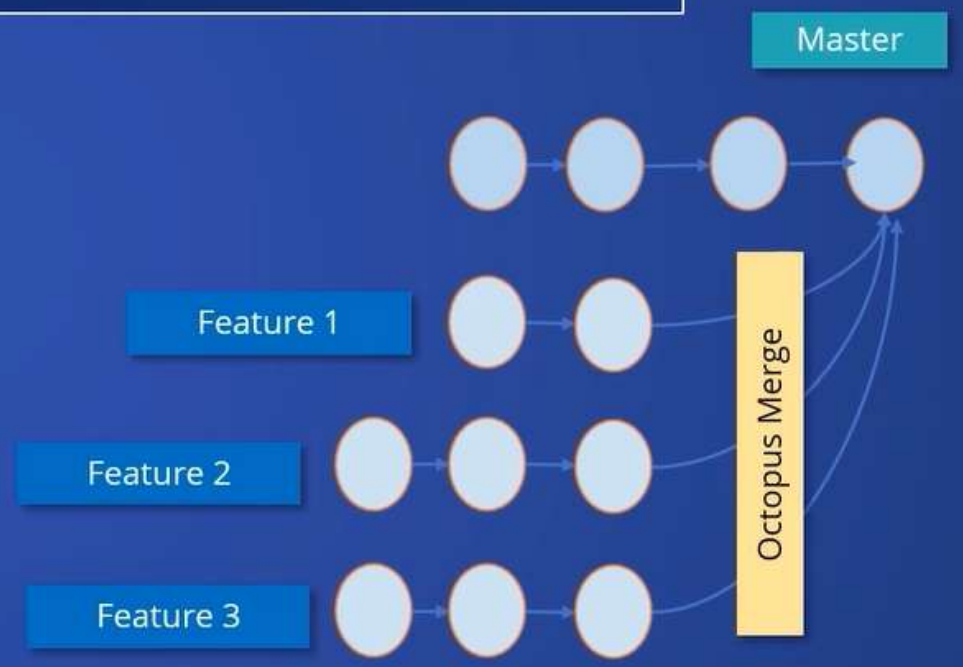
Octopus

To deal when merge involves two or more branches

```
$ git merge -s octopus branch1 branch2 branch3 branchN
```

Does NOT allow manual conflict resolution

Benefit
More than one branches can be merged



© LearnKartS. All rights reserved.

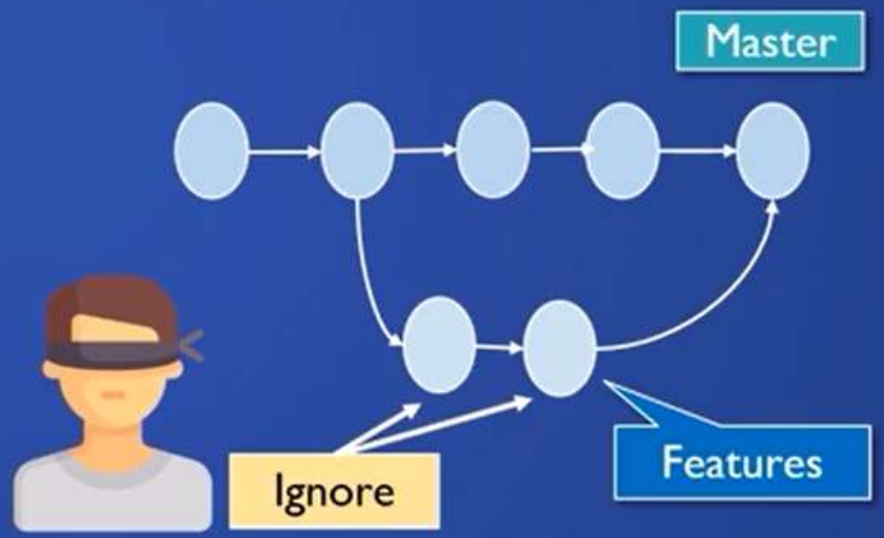
LearnkartS
Certification Prep and Projects

Ours

Does NOT consider the changes from feature branch in case of conflict and uses our changes

```
$ git merge -s ours branch1 branch2 branchN
```

All the changes from feature branch is ignored and changes from master branch will be considered



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Subtree

Used for merging trees which have common ancestor

```
$ git merge -s subtree branchA branchB
```

Common ancestor

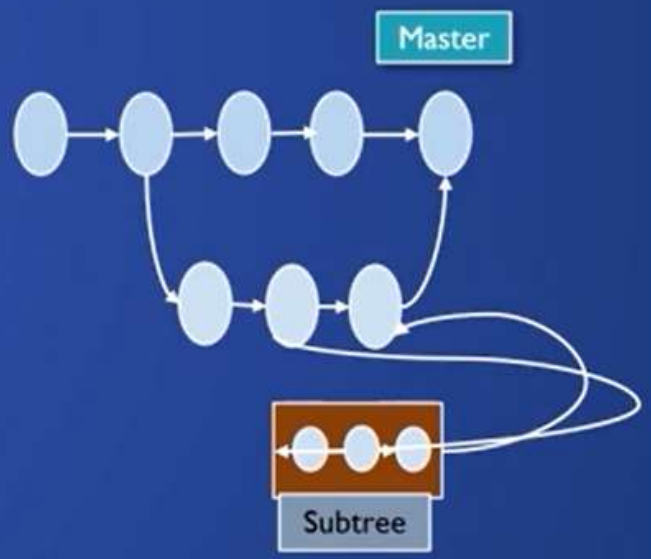
Tree A

Tree B

Merge A and B

B is modified to reflect the tree structure of A if a sub-tree of A is corresponding

Sub-tree B : Incremental changes are done on top



Since both are from common ancestor subtree will handle the merge

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Rebase

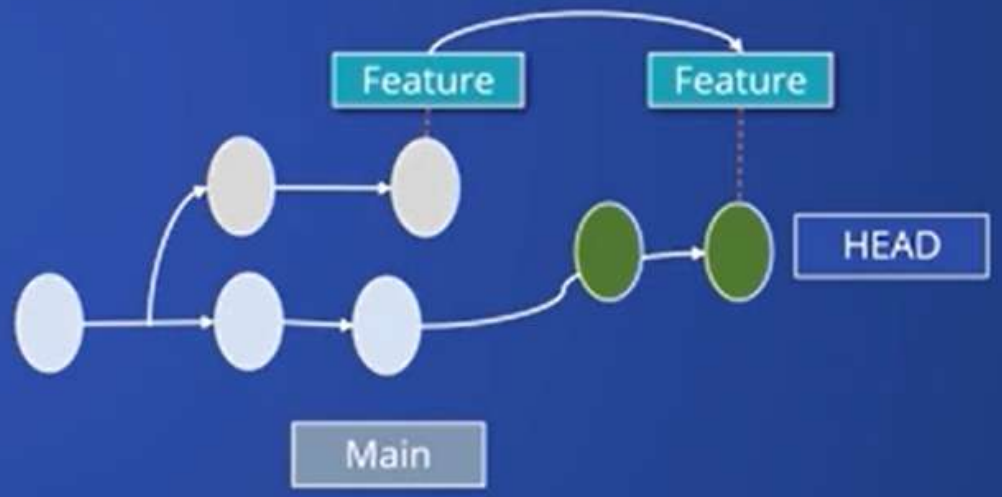
Rebase is a single "patch" for all changes

Rebase is another way to integrate changes between branches

Copy the change and implement in a branch as a patch

No merge commit will be created in the destination branch

Seems like a local commit



© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Why Rebase?

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Git Rebase vs Git Merge

Choose Merge

- To add branch changes back to the base branch
- To keep the same history rather than rewrite it
- To revert the changes quickly

Choose Rebase

- To squash multiple commits
- Can change the order of commits
- Helps clean the history

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Lesson 6:

Git with IDE

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

IntelliJ IDE

Smart Code Completion

Chain Code Completion

Context based coding
Auto-completion feature

Advanced code completion feature → Provides all context-based options

```
@GetMapping("/")
@ResponseBody
@Transactional(readOnly = true)
public List<City> helloWorld() {
    return cs.find;
}
    cityService.findCities(CitySearchCriteria criteria, Pageable pagea
^↓ and ^↑ will move caret down and up in the editor >>
```

```
atom.coffee
24  module.exp
    exports
    exportsPath
    exception
    Examples
    executeJavaScriptInDevTools
```

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

IntellJ IDE

Static Member's Completion

Allows to use static methods or constants

Automatically adds the required import statements to avoid compilation error

Detecting Duplicates

Finds the duplicate code or variable

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

IntelliJ IDE

Inspections and Quick-fixes

Whenever IntelliJ detects that one is about to make a mistake, a little light bulb notification pops up on same line

Gives suggestions to developers specifically to improve quality

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Lesson 7:

GitLab

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Features of GitLab

User-friendly Interface

Branches – Private or public

Ability to manage multiple repositories

Code review

Built-in bug and issue tracking

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Features of GitLab

Built-in, free Continuous Integration/Continuous Delivery (CI/CD)

Project Wikis

Easy creation of code snippets for sharing parts of the code

GitLab licensing and delivery models

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Lesson 8:

Bitbucket

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Pipelines

An integrated CI/CD service

Separate CI server (like Jenkins, TeamCity, Bamboo) is NOT required

Usage
Build automation
Deployment automation

Provides end to end pipeline

Usage quota – in minutes

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Pipes

Pre-defined workflows to automate CI/CD pipeline

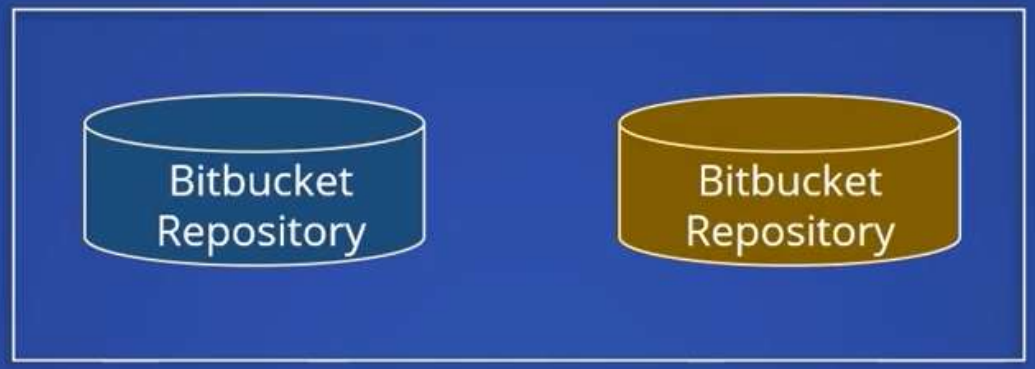
Configure as per the specific need

Automatically generates pre-defined pipelines which can be integrated easily

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects

Project



Composition of repositories

Makes it easier for teams to focus on a goal product

Project can have multiple repositories

© LearnKartS. All rights reserved.

LearnkartS
Certification Prep and Projects