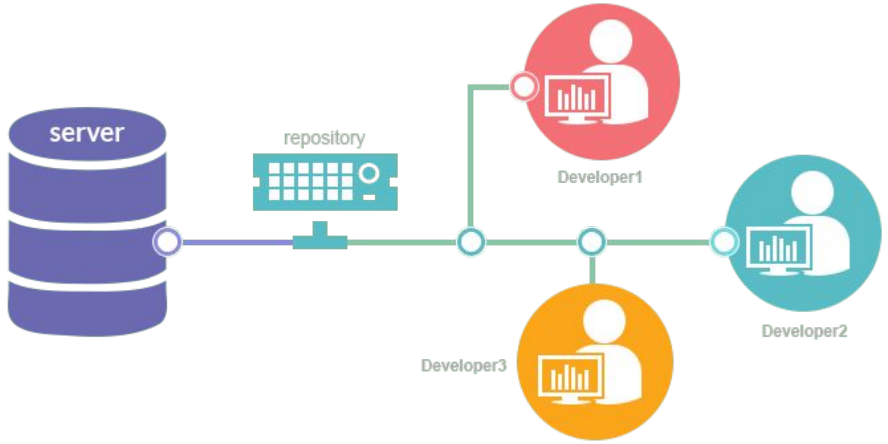
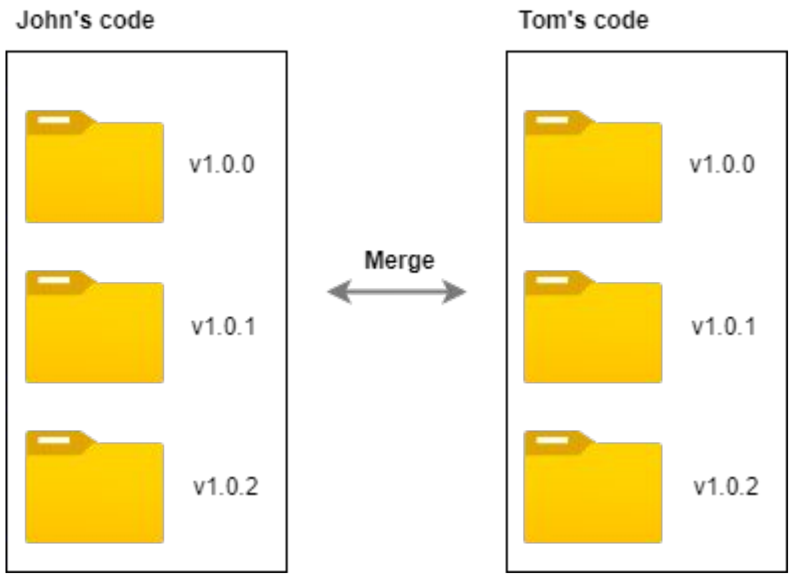
git for Dummies

What is git?

- Most popular **Version Control System(VCS)**
- Tracks changes in source code
- Distributed: Each user has full copy of the repository

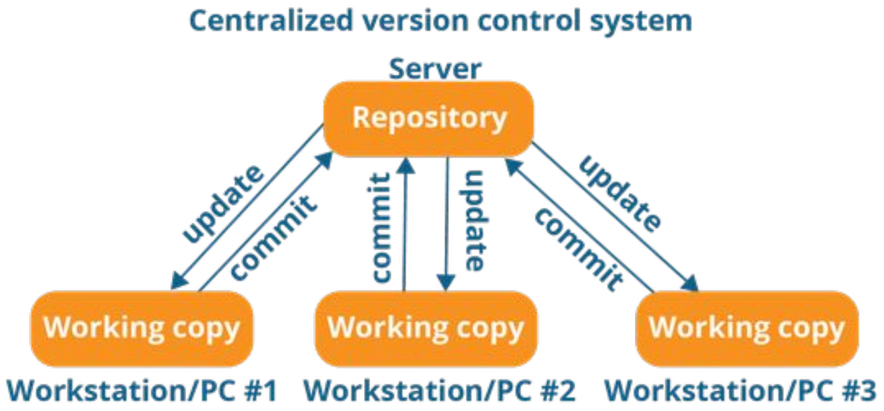


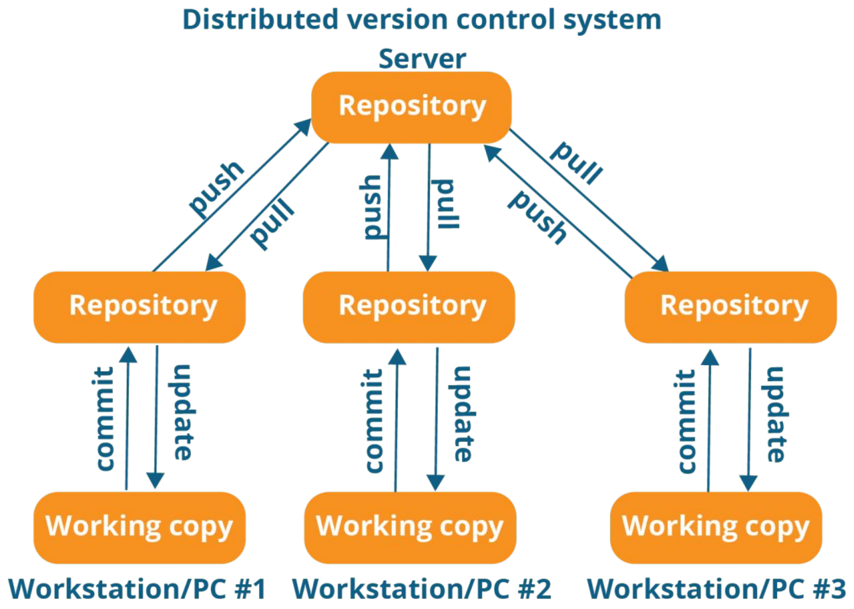
Why we need VCS?



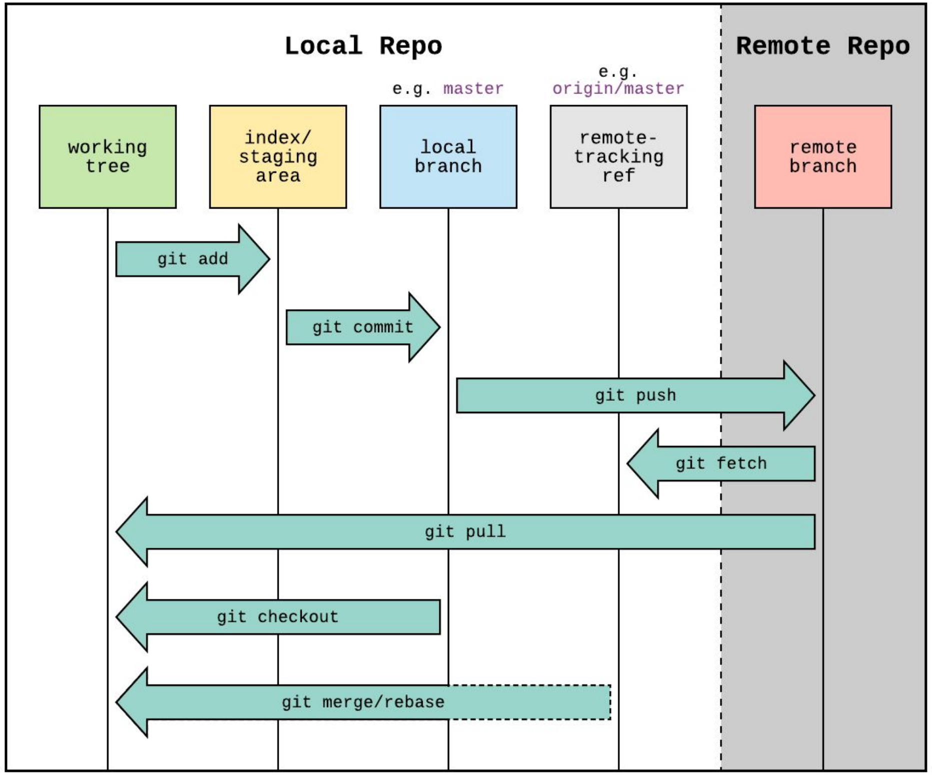
Types of VCS

1. Centralized VCS Architecture
2. Distributed VCS Architecture





Git Workflow



› git add

- Stage specific file

```
> git add <file_path_1> <file_path_2> <file_path_3>
```

- Stage by extension

```
> git add *.txt
```

- Stage all changes

```
> git add .
```

- Can remove staged changes using

```
> git rm --cached <file_path>
```

› git status

- Displays the state of the working directory and the staging area

```
> git status
```

Previously staged changes can be seen. Let’s do a new change change and see what happens.

› git commit

- Create a snapshot of the staged changes along a timeline of a Git projects history.

```
› git commit -m <commit_message>
```

Flags

- *-a and -am need tracked files with modifications*
- *git commit -a* :Stage and commit all the changes
- this will open the editor to enter message
- *git commit -am* :Stage and commit with commit message

Tips

- **Commit size**: don’t make it big nor small
- **When**: Create milestones/sub-tasks and commit upon completion
- **How**: Commit often, but commit when needed
- Use meaningful commit messages

.gitignore

- The purpose of gitignore files is to ensure that certain files not tracked.

Includes:

- Sensitive data containing files
- Log files
- Files generated by code quality control tools
- Third party libraries
- Media files etc.

Git branches

- A Git branch is like a separate workspace for your changes
- Helps you work on different tasks without messing up your main project
- Perfect for adding new features or fixing bugs without affecting the main code

To create a branch

```
> git branch <branch_name>
```

To see branch list

```
  > git branch
```

Git branches cont.

To switch between branches

```
> git checkout <branch_name>
```

To create a branch and switch to it in one go

```
> git checkout -b <branch_name>
```

To remove branch

```
  > git checkout -d <branch_name>
```

Tips

- **Clear Names**: Use simple names that describe the task (e.g., login-feature, bugfix-123)
- **Frequent Saves**: Save/commit your work often to avoid problems
- **Merge Often**: Keep your branches up to date
- **Clean Up**: Delete branches when you’re done with them

› git push

- The git push command is used to upload local repository content to a remote repository

```
> git push -u origin <branch_name>
```

`*origin*` (upstream) is the default name for the remote repository you cloned from

- `*-u*` flag: Sets the upstream (tracking) branch for the current local branch
- If you didn't use `*-u*` flag, You must specify the remote and branch every time

```
  > git push origin <branch_name>
```

Pull Requests(PR)

- **Code Review**: Facilitates peer review of changes
- **Discussion**: Enables conversations around code changes
- **Approval Process**: Requires review and approval before merging
- **Integration Testing**: Runs tests to ensure code quality
- **Conflict Resolution**: Helps manage and resolve merge conflicts.

› git pull

- **Update Local Repository**: Fetches and merges changes from a remote repository
- **Synchronization**: Keeps your local branch up-to-date with the remote branch
- `***git pull***` Fetches and merges changes from the default remote branch (usually origin/main or origin/master).

```
> git pull
```

- `***git pull origin <branch_name>***` Fetches and merges changes from a specified branch on the remote repository

```
> git pull origin <branch_name>
```

Conflicts

- **Conflict Occurrence**: Happens when changes in the local and remote branches cannot be automatically merged
- **Conflict Resolution**:
  - **Identify Conflicts**: Git will mark conflict areas in the files.
  - **Manual Resolution**: Edit the conflicting files to resolve issues.

***Note**:*

***Automatic Merging**: Git tries to merge changes automatically.*

***Manual Intervention**: Sometimes manual editing is required for conflicts.*

***Best Practices**: Regularly pull changes to minimize conflicts.*

› git fetch

- **Download Changes**: Retrieves updates from a remote repository
- **No Merge**: Unlike git pull, it does not merge changes automatically
- `git fetch`: Fetches updates from the default remote repository
- `git fetch origin <branch_name>`: Fetches updates from the specified remote repository
- **View Changes**: `git log origin/<branch_name>`
- **Compare Changes**: `git diff origin/<branch_name>`
- **Merge changes**: `git merge origin/<branch_name>`

Notes:

- **Safety**: Ideal for cautious updates, as it doesn't alter your working directory
- **Combine with Pull**: Typically used before git pull to see changes

Todo

Explore Other Git Commands

Advanced Branching and Merging

- `git rebase`: Reapply commits on top of another base tip.
- `git cherry-pick`: Apply changes from specific commits.

History and Inspection

- `git blame`: Show what revision and author last modified each line of a file.
- `git show`: Display various types of objects (commits, tags, etc.).
- `git reflog`: Show the history of all changes to the tip of branches.

Stashing and Cleaning

- `git stash`: Temporarily save changes that are not ready to be committed.
- `git stash pop`: Reapply stashed changes and remove them from the stash list.
- `git clean`: Remove untracked files from the working directory.

Tagging

- `git tag`: Create, list, delete, or verify tags.
- `git tag -a <tag_name> -m "message"`: Create an annotated tag.

Undoing Changes

- `git reset --hard <commit>`: Reset the index and working directory to a specific commit.
- `git revert <commit>`: Create a new commit that undoes changes from a previous commit.

Reference

https://www.youtube.com/watch?v=0chZFIZLR_0

https://www.youtube.com/watch?v=8JJ101D3knE&t=3722s&pp=ygUDZ2l0

https://www.youtube.com/watch?v=T13gDBXarj0&list=PLfU9XN7w4tFzW200TaCP1W9RTE8jRSHU5&pp=iAQB

https://www.youtube.com/watch?v=T13gDBXarj0&list=PLfU9XN7w4tFwKwh_xPSQ_X1-hROQEpHnM&pp=iAQB

https://www.youtube.com/watch?v=Uszj_k0DGsg&pp=ygUSZ2l0IGdvb2QgcHJhY3RpY2Vz