

| | | |
|-----------------------------|---|-------------|
| EX. No: 5 DATE:20.2.2024 | WORKING WITH COLLABORATIVE REPOSITORY MANAGEMENT USING GIT | URK23CS7021 |
|-----------------------------|---|-------------|

Working with collaborative repository management using Git involves collaborating with other developers and effectively managing changes to a shared codebase. Here are some key practices and concepts to consider:

Forking a Repository

- When you want to contribute to a project hosted on a remote repository, it's common to start by forking the repository.
- Forking creates a personal copy of the repository under your GitHub account or another hosting platform, where you can freely make changes without affecting the original repository.
- To fork a repository on GitHub, navigate to the repository's page and click the "Fork" button.

Cloning a Forked Repository

- After forking a repository, you need to clone the forked repository to your local machine to start making changes.
- Use the 'git clone' command, as explained in a previous answer, to clone the forked repository.

Adding an Upstream Remote

- The original repository that you forked is known as the "upstream" repository. It represents the source of truth for the project.
- To synchronize your forked repository with the latest changes from the upstream repository, you can add an "upstream" remote.
- Use the "git remote add" command to add the upstream remote URL. For example:

git remote add upstream <upstream-url>

Keeping Your Forked Repository Up to Date

- To incorporate the latest changes from the upstream repository into your forked repository, follow these steps:
 - Fetch the latest changes from the upstream remote: **"git fetch upstream"**
 - Checkout your local main branch: **"git checkout main"**
 - Merge the changes from the upstream/main branch into your local main branch: **"git merge upstream/main"**
 - Push the updated main branch to your forked repository: **"git push origin main"**

Creating Branches for Collaborative Work

- When working on a collaborative project, it's common to create branches for new features, bug fixes, or other changes.
- Use the 'git branch' command to create a new branch: **"git branch <branch-name>"**
- Switch to the newly created branch using 'git checkout': **"git checkout <branch-name>"**
- Alternatively, you can create and switch to a new branch in one command: **"git checkout -b <branch-name>"**

Pushing Changes and Creating Pull Requests

- Once you've made changes on a branch, you can push the branch to your forked repository using **"git push origin <branch-name>"**.
- After pushing the branch, you can create a pull request on the upstream repository to propose your changes for merging into the main codebase.
- On the upstream repository's page, find the "Pull requests" section and click the "New pull request" button.
- Select the appropriate branches for the base (usually main) and compare (your branch) branches.
- Provide a title and description for your pull request, then click "Create pull request" to submit it.

Reviewing and Merging Pull Requests

- Collaborators or maintainers of the upstream repository can review and comment on your pull request.
- They may request changes or provide feedback before merging the changes.
- Once the pull request is approved, it can be merged into the main branch of the upstream repository.