1. Pull a remote branch locally
   Git fetch <remote-repository> <remote-branch>
   Eg:- git fetch origin user/t-aananth….
   Git checkout -b <local-branch> <remote-repository>/<remote-branch>
   https://www.loginradius.com/blog/async/git-fetch-remote-branch/patch

2. Cherry pick commits
   a. Git checkout <local-branch>
      Git cherry-pick <commit-id>
      Note: branch needs to be present locally before cherry picking, so do a git fetch beforehand
      https://www.atlassian.com/git/tutorials/cherry-pick

b. To cherry-pick all the commits from commit A to commit B (where A is older than B), run [Very useful while backporting]:
   git cherry-pick A^..B
   If you want to ignore A itself, run:
   git cherry-pick A..B

3. Deleting branches
   a. Locally:
      Git branch - d <local-branch>
   b. Remotely:
   git push origin --delete <remoteBranchName>

4. Going back a commit
   Git checkout <commit-id>

a. If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command.eg:- git switch -c <new-branch-name>
b. Or undo this operation with: git switch -

5. Stashing:
   Git stash - work gets stashed
   git stash save "add style to our site"
   Git stash pop OR git stash pop stash@{2}
   Stash a specific file: git stash push -m "describe changes to filename.ext" filename.ext
   Pop a specific file: git checkout stash@{0} -- <filename>
   Abort a stash pop: git reset --merge

6. Reverting:
   Eg:- (commits) a->b->c
   Git revert a
   (commits) a->b->c->d
   Where 'd' is a newly created commit, with 'a' undone, but b,c intact

7. Resetting, Checking Out & Reverting | Atlassian Git Tutorial

8. Restore a file(s) to its state in a commit:
   Assuming the hash of the commit you want is c5f567:
   git checkout c5f567 -- file1/to/restore file2/to/restore

9. Add a change to the previous commit (don't do this on public branches - causes BT when you need to sync local and remote)
   (some_branch) git add <file1> <file2> …
   (some_branch) git commit --amend

10. Add change to any commit - https://confluence.atlassian.com/stashkb/how-do-you-make-changes-on-a-specific-commit-747831891.html - do this only while cherry picking locally.

11. Git clean:
    To remove untracked files
    Untracked files are files that have been created within your repo's working directory but have not yet been added to the repository's tracking index using the git add command.
    git clean -n or -nd (tells what files/directories will be removed)
    -x to remove "ignored" files(out/debug) also
    Git clean [-d] (removes them)
    Git clean [-f or -fd] (force cleans)
    Git Clean | Atlassian Git Tutorial

12. Renaming branch:
    git branch -m <oldname> <newname>
    Current branch:
    git branch -m <newname>

    git push origin -u <newname>

13. Applying "patches"
    a. Git diff … > mypatch.patch
    b. Git apply mypatch.patch
    c. If that fails,
       git apply --reject --whitespace=fix mychanges.patch
       i. --reject option will instruct git to not fail if it cannot determine how to apply a patch, but instead to apply the individual hunks. If that fails, it creates .rej files, so you can check those and manually apply
       ii. --whitespace=fix will warn about whitespace errors and try to fix them
       iii. https://git-scm.com/docs/git-apply

15. Finding commit of a branch:
    git rev-parse branch-name

17. Miscallaneous
    a. Git fetch - download objects from the remote
    b. Git pull = fetch + merge
       but, git fetch remote branch && git rebase remote/branch is somehow usually faster & cleaner than git pull (why??)

c. Tracking = Tracking branches are local branches that have a direct relationship to a remote branch. If you're on a tracking branch and type git pull , Git automatically knows which server to fetch from and which branch to merge in.
d. Cannot setup tracking information; starting point 'origin/branch-name' is not a branch

Probably your origin remote is set up to fetch only certain branches
git remote set-branches --add origin branch-name
fixes it