Contents

1 Definition

Git is a distributed version control system. In contrast to distributed version control system(CVCS)

2 commands

- `git --version`
- `git config --list`
- `git status`

- `git log`

3 git ignore file

- `.DS_STORE`
- `.project`
- `*.pyc`

4 add files in staging are

- `git add -A`
- `git reset` *remove all files from staging area*

5 commit

- `git commit -m "message"` *commits staged files*

6 clonong a remote repo in git

- ~git clone <url> <where to clone>
- *ex* `git clone ../local-repo.git .` *cloning a local repo inside a directory*

6.1 view infornation about the remote repo

- `git remote -V`
- `git branch -a`

7 Pushing changes

7.1 First commit like done before

- `git diff`
-

7.2 Then Push

- `git pull origin master`
- `git push origin master`

8 Common workflow while using git

8.1 Dont commit at the master branch, create a branch for a desired feature.

- ~git branch <branch-name>
- ~git checkout <branch-name>
-

8.2 After commit push branch to remote

- ~git push -u origin calc-divide *-u tells that we wanna associate our local and remote branch*

8.3 Merge a branch with master

- `git checkout master`
- `git pull origin master`
- `git branch --merged`
- `git merge <branch-name>`
- `git puch origin master`

9 Errors and trouble shooting

9.1 Bare and Non bare files problem

```
Centros@Centros-PC MINGW64 /d/code/git/remote-repo (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
remote: error: refusing to update checked out branch: refs/heads/master
remote: error: By default, updating the current branch in a non-bare repository
```

```
remote: is denied, because it will make the index and work tree inconsistent
remote: with what you pushed, and will require 'git reset --hard' to match
remote: the work tree to HEAD.
remote:
remote: You can set the 'receive.denyCurrentBranch' configuration variable
remote: to 'ignore' or 'warn' in the remote repository to allow pushing into
remote: its current branch; however, this is not recommended unless you
remote: arranged to update its work tree to match what you pushed in some
remote: other way.
remote:
remote: To squelch this message and still keep the default behaviour, set
remote: 'receive.denyCurrentBranch' configuration variable to 'refuse'.
To D:/code/git/remote-repo/../local-repo/.git
 ! [remote rejected] master -> master (branch is currently checked out)
error: failed to push some refs to 'D:/code/git/remote-repo/../local-repo/.git
```

9.2 Solution Summary

You cannot push to the one checked out branch of a repository because it would mess with the user of that repository in a way that will most probably end with **loss of data and history**. But you can push to any other branch of the same repository.

As bare repositories never have any branch checked out, you can always push to any branch of a bare repository.

There are multiple solutions, depending on your needs.

- Solution 1: Use a Bare Repostiory

As suggested, if on one machine, you don't need the working directory, you can move to a bare repository. To avoid messing with the repository, you can just clone it:

```
machine1$ cd ..
machine1$ mv repo repo.old
machine1$ git clone --bare repo.old repo
```

Now you can push all you want to the same address as before.

- Solution 2: Push to a Non-Checked-Out Branch

But if you need to check out the code on your remote '<remote>', then you can use a special branch to push. Let's say that in your local repository you have called your remote 'origin' and you're on branch master. Then you could do

~ machine2$ git push origin master:master+machine2~

Then you need to merge it when you're in the 'origin' remote repo:

~ machine1$ git merge master+machine2~

- Autopsy of the Problem

When a branch is checked out, committing will add a new commit with the current branch's head as its parent and move the branch's head to be that new commit.

So

A B [HEAD,branch1]

becomes

A B C [HEAD,branch1]

But if someone could push to that branch inbetween, the user would get itself in what git calls **detached head** mode:

A B X [HEAD] [branch1]

Now the user is not in branch1 anymore, without having explicitly asked to check out another branch. Worse, the user is now **outside any branch**, and any new commit will just be **dangling**:

[HEAD] C A B X [branch1]

Hypothetically, if at this point, the user checks out another branch, then this dangling commit becomes fair game for Git's **garbage collector**.