GIT BRANCHING

WEBRTC VIRTUAL CLASSROOM PLATFORM

TABLE OF CONTENTS

GIT COMMANDS

PREREQUISITE

Git is installed on user local machine and user has a local copy of a remote branch.

VIEW LOCAL BRANCHES

```
git branch
```

VIEW REMOTE BRANCHES

```
git branch -r
```

VIEW ALL BRANCHES

```
git branch -a
```

CREATE LOCAL BRANCH

```
git branch 'branch-name'
git checkout 'branch-name'
```

The above command creates a local branch forked off from the current branch you are currently in.

Alternately, a branch can be created and checked out via single command

```
git checkout -b 'branch-name' master|develop
```

DELETE LOCAL BRANCH

```
git branch -d 'branch-name'
```

CREATE REMOTE BRANCH

```
git branch 'branch-name'
git push origin 'branch-name'
```

DELETE REMOTE BRANCH

```
git push origin -delete 'branch-name'
```

SYNC DELETED REMOTE BRANCH

After someone deletes a branch from a remote repository, git doesn't automatically delete the local repository branches on **git pull** or **git fetch**. However, if the user would like to have all tracking branches removed from their local repository that have been deleted in a remote repository, they can type:

```
git remote prune origin
```

CHECK OUT A BRANCH LOCALLY

```
git checkout 'branch-name'
```

UPDATING CODE FROM REMOTE BRANCH

```
git pull
```

git pull is equivalent to

```
git fetch
git merge
```

REBASE LOCAL BRANCH

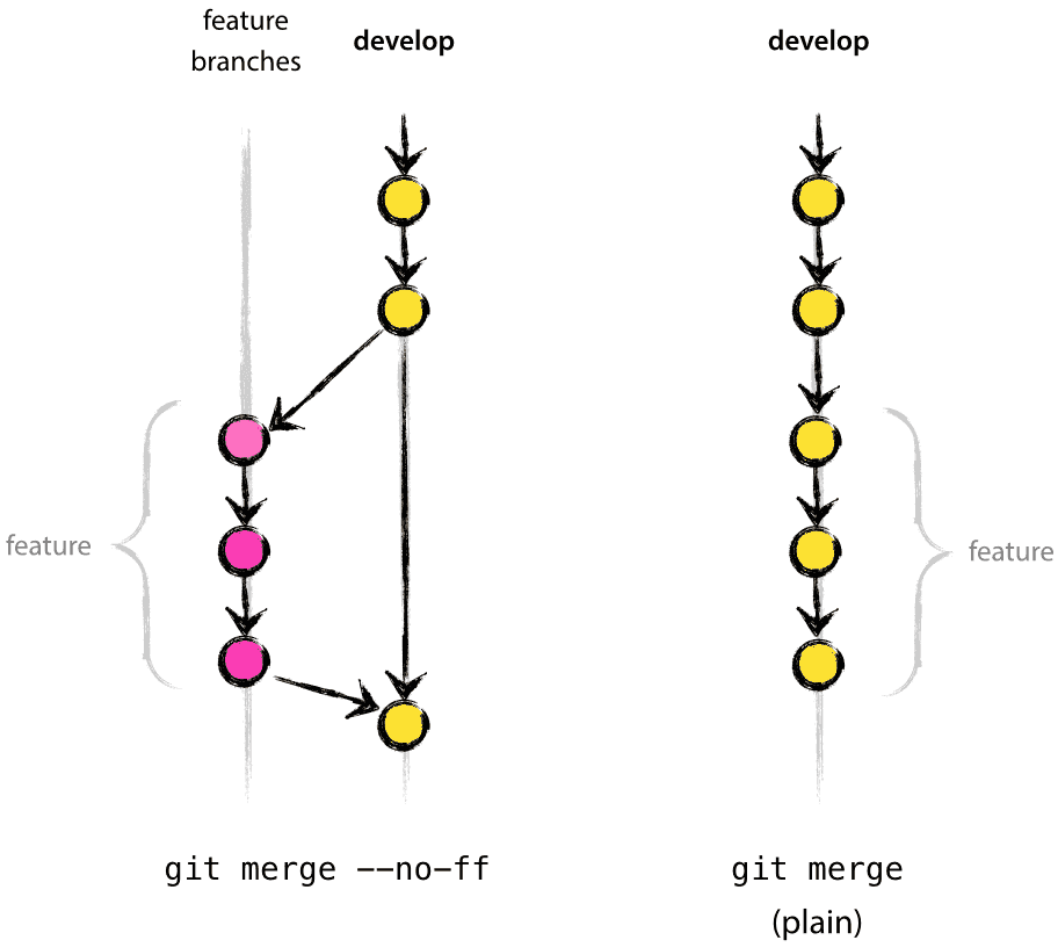
```
git rebase, OR
git pull --rebase
```

With the rebase command, you can take all the changes that were committed on one branch and replay them on another one. This results in merged code and is slightly different from a three-way merge done via git merge command.

Let's say we want to merge a local 'efk' branch to 'develop' branch. Instead of doing a git pull on develop and then merging the 'efk', an alternate way to achieve the same thing is

```
git rebase develop efk // checkout efk and replay it on develop
git checkout develop
git merge --no-ff efk
```

The --no-ff flag causes the merge to always create a new commit object, even if the merge could be performed with a fast-forward. This avoids losing information about the historical existence of a feature branch and groups together all commits that together added the feature. Compare:



In the latter case, it is impossible to see from the Git history which of the commit objects together have implemented a feature—you would have to manually read all the log messages. Reverting a whole feature (i.e. a group of commits), is a true headache in the latter situation, whereas it is easily done if the --no-ff flag was used.

Further read on understanding rebase command:

https://git-scm.com/book/en/v2/Git-Branching-Rebasing

MERGE LOCAL BRANCH

```
git checkout develop
git merge --no-ff efk
```

The above commands merge the local 'efk' branch to the develop branch. Please remember to stage and commit your changes in the local branch before merging to develop branch.

VIEW MERGED BRANCHES

```
git branch --merged
efk
*develop
```

Because you already merged in 'efk' branch earlier, you see it in your list. Branches on this list without the * in front of them are generally fine to delete with git branch -d; you've already incorporated their work into another branch, so you're not going to lose anything.

VIEW UNMERGED BRANCHES

```
git branch --no-merged
```

VIEW COMMIT HISTORY OF A BRANCH

```
git log
```

TAG A BRANCH

```
git tag -a 'tag-name'
git show 'tag-name'
git push origin 'tag-name'
git tag // lists all tags
git push orign --tags // pushes all tags
```

TAG A COMMIT SHA

```
git tag -a 'tag-name' 'specific-commit-SHA-goes-here'
git push origin 'tag-name'
```

CHECKING OUT TAGS

```
git checkout -b 'branchname' 'tagname'
```

BRANCH STRATEGY

Permanent running branches:

master – goes to production, gets tagged, code gets merged from release branch via pull requests, push is restricted or disabled, branch deletion is forbidden

release – branch off master, QA branch, branch deletion is forbidden, push restricted or forbidden, get updates from develop branch via pull request

develop – branch off master, development branch, delete forbidden. Merge can be done either via pull requests from developer remote feature branches or from a direct merge from developer feature/topic branches. Needs to finalize on either of the strategy

Temporary branches

topic or feature branches – feature specific. Deleted after feature is merged into master branch

hotfix branches – fix production issues. gets merged into release, develop and master branch.

BRANCH PERMISSIONS AND CONTROLS

Bitbucket has options for fine grained controls settings on a branch like disable branch deletion, restriction of git push restriction by users, pull reviewers etc. They can be viewed in the UI under settings option.

PULL REQUESTS

When a developer has finished developing a feature, or has done a hotfix, a pull request can be generated to merge changes in a branch where git push is restricted or forbidden.

Pull requests let you tell others about changes you've pushed to a repository on git. Once a pull request is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary.

Pull requests can be initiated, managed, discussed, reviewed, merged and closed from Bitbucket UI. For further reading please visit https://help.github.com/articles/using-pull-requests/

REFERENCES

http://nvie.com/posts/a-successful-git-branching-model/

https://git-scm.com/book/en/v2/Git-Branching-Rebasing

https://help.github.com/articles/using-pull-requests/