HTTP Methods and Codes

What are the reasons for below errors:

→ **5xx**
500
502
503
504

→ **4xx**
400 →
401
403 → Forbidden
404 → Not found
405

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

What are the means of Below codes:

→ **3xx**
301
302
307
308

→ **2xx**
200
201
204

→ **1xx**
100
102

HTTP methods

GET → Read *

POST → Create *

PUT → Replace/Update/Change *

DELETE → Delete *

PULL → Download

PUSH → Upload

cURL → Request → Cross Origin for Domains

What is Azure Bastion?

https://learn.microsoft.com/en-gb/azure/bastion/bastion-overview

Azure CDN

https://azure.microsoft.com/en-us/products/cdn/
https://learn.microsoft.com/en-us/azure/cdn/cdn-overview?toc=%2Fazure%2Ffrontdoor%2FTOC.json

SCM (Source code Management) → GitHub, Bitbucket, GitLab

Tasks for Students:

1. Watch Azure AD section
2. Practice all Github exercise and done with all repo creation and other practicales
3. Create account in Bitbucker
4. Create account in GitLab

Bitbucket

Introduction

- Git repository management solution
- Collaborate on your source code
- It has 3 different deployment model
  - Bitbukcet cloud
  - Bitbucket Datacenter
  - Bitbucket Server
- What is Bitbucket: It is a cloud based service used to store source code and manage it.
- Bitbucket is used in:
  Access control
  Pull request
  Work flow control
  JIRA & Confluence
  Full rest API
  Bitbucket cloud and server.
- Bitbucker is a Sources code repository hosting service owned by Atlassian.
- Jesper Noehr in 2008

Feature & Applications

- Code review system to review pull request
- Use Git version control system
- Powerfull JIRA interaction
- In-line discussion - Comment suggest
- Bitbucket Cloud & Server
- Built-in Issue tracker to track the status of bug
- Bitbucket Interface - GUI
- Unlimited private repos
- 10 times cheaper Github
- Built-in JIRA & Trello
- It has a built-in CI/CD solution

What is CI?CD?

CI/CD is a process used to automate the building, testing, and finally deployment of applications on the cloud or wherever you want to deploy.

- Bridges the gap between development and Ops and no need to use 3rd party CD/CD like Jenkins.
- Authentication via Github
- Import repos
- Free for small teams with unlimited private repos

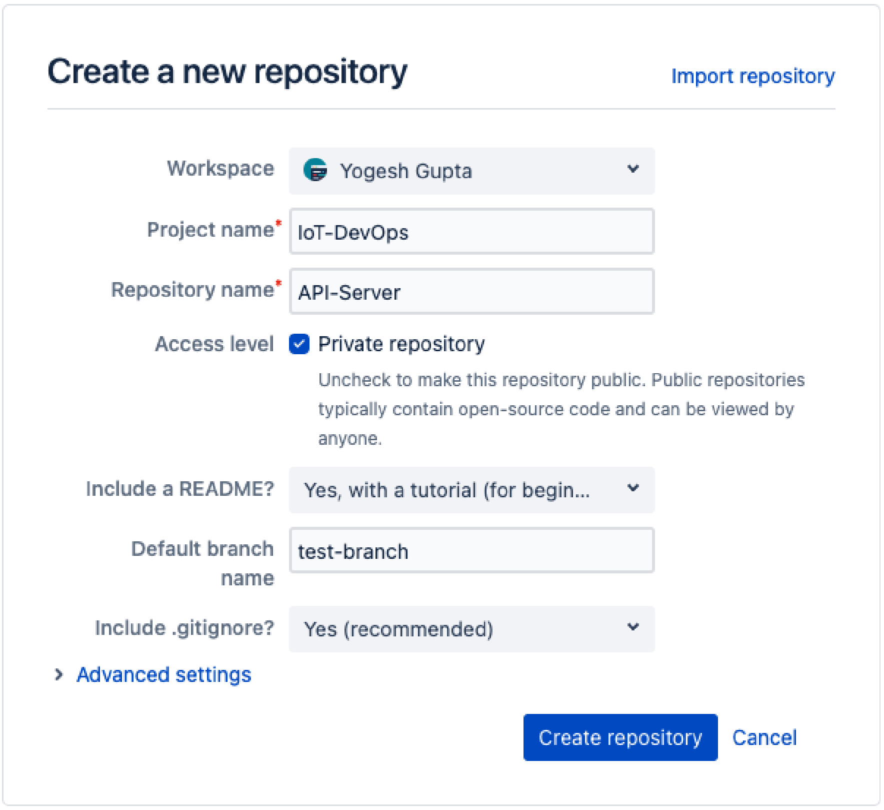
Terminologies

- Workspace
- Project
- Repository
- Access level
- Branch
- Gitignore
- clone

Distributed version control:

- Git (This is the most usable control system in Orgs)
  Free, Open Source, Security SHA-1 encryption
- Mercurial
- Bazaar

Create a new Repo in BitBucket with new account.



Create a new repository

Workspace	Yogesh Gupta
Project name*	IoT-DevOps
Repository name*	API-Server
Access level	✓ Private repository Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.
Include a README?	Yes, with a tutorial (for begin...
Default branch name	test-branch
Include .gitignore?	Yes (recommended)

How to create a new Repository on Bitbucket and add it to Project?
How to change repo's project is in Bitbucket?
How to create a file in Bitbucket repo?

Git configuration

Install git in your local machine
https://git-scm.com/downloads

# git --version

Cloning a Public Repo

# git clone <public repo>
# git clone https://github.com/yogeshgupta0246/devops-study.git
# git clone https://yogeshgupta0246@bitbucket.org/yogeshgupta0246/test1.git

Cloning a Private Repo in GitHub over HTTPS

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
# git clone https://github.com/yogeshgupta0246/test-website.git

Note: You will give username and password then will face issue and need to generate personal access token on Github.
https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Note: If you are accessing an organization that uses SAML SSO and you are using a personal access token (classic), you must also authorize your personal access token to access the organization before you authenticate. For more information, see "About authentication with SAML single sign-on" and "Authorizing a personal access token for use with SAML single sign-on."

Note: If you'd rather use SSH but cannot connect over port 22, you might be able to use SSH over the HTTPS port. For more information, see "Using SSH over the HTTPS port."

Cloning a Private Repo in GitHub over SSH URLs

SSH URLs provide access to a Git repository via SSH, a secure protocol. To use these URLs, you must generate an SSH keypair on your computer and add the public key to your account on GitHub.com. For more information, see "Connecting to GitHub with SSH."

Generating a new SSH key

https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Adding a new SSH key to your GitHub account
https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

```
# git clone git@github.com:yogeshgupta0246/test-website.git
```

Cloning a Private Repo in Bitbucket over SSH

```
# git clone git@bitbucket.org:yogeshgupta0246/new-test-repo1.git
```

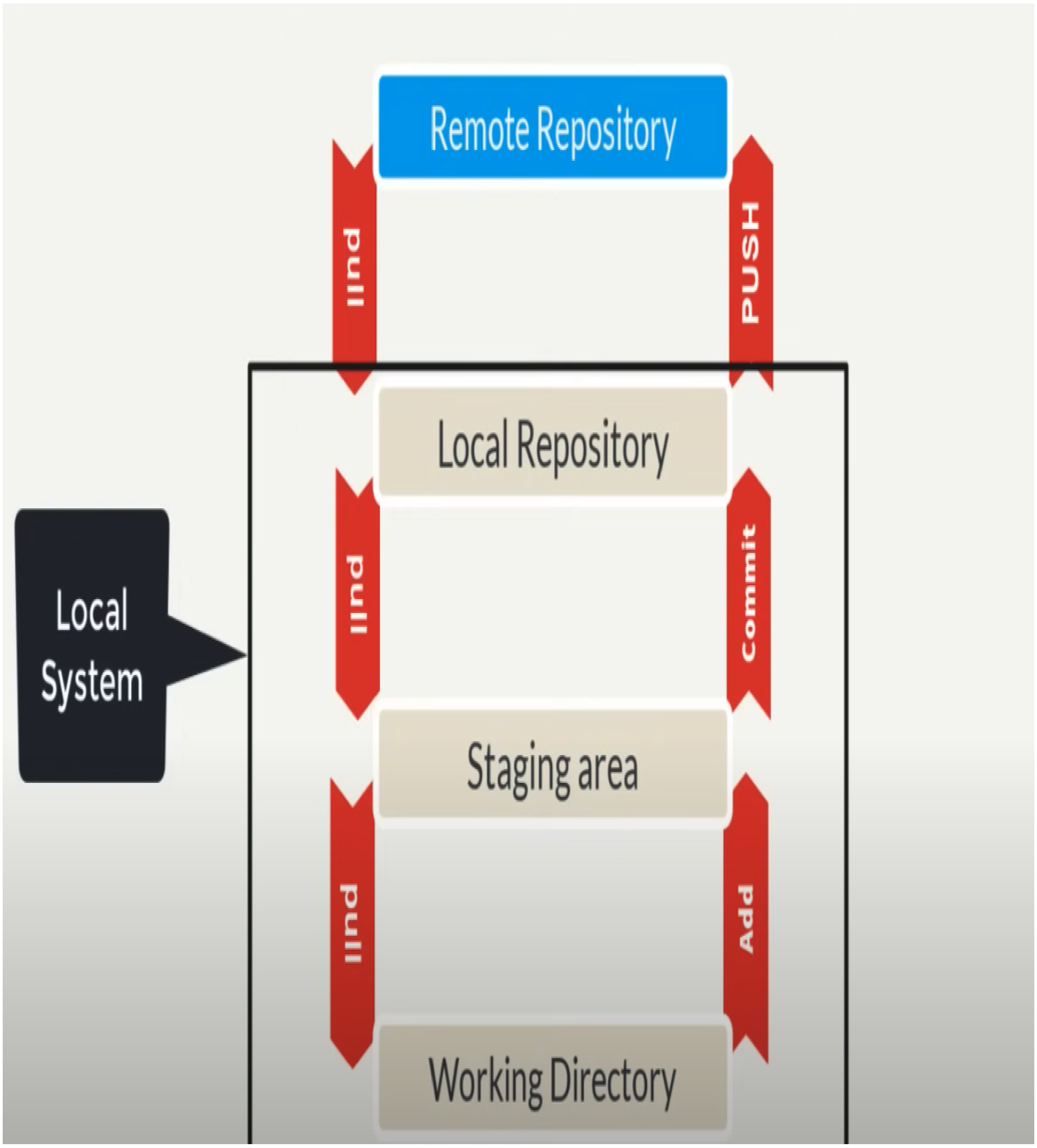
```
# ssh -T git@bitbucket.org
```

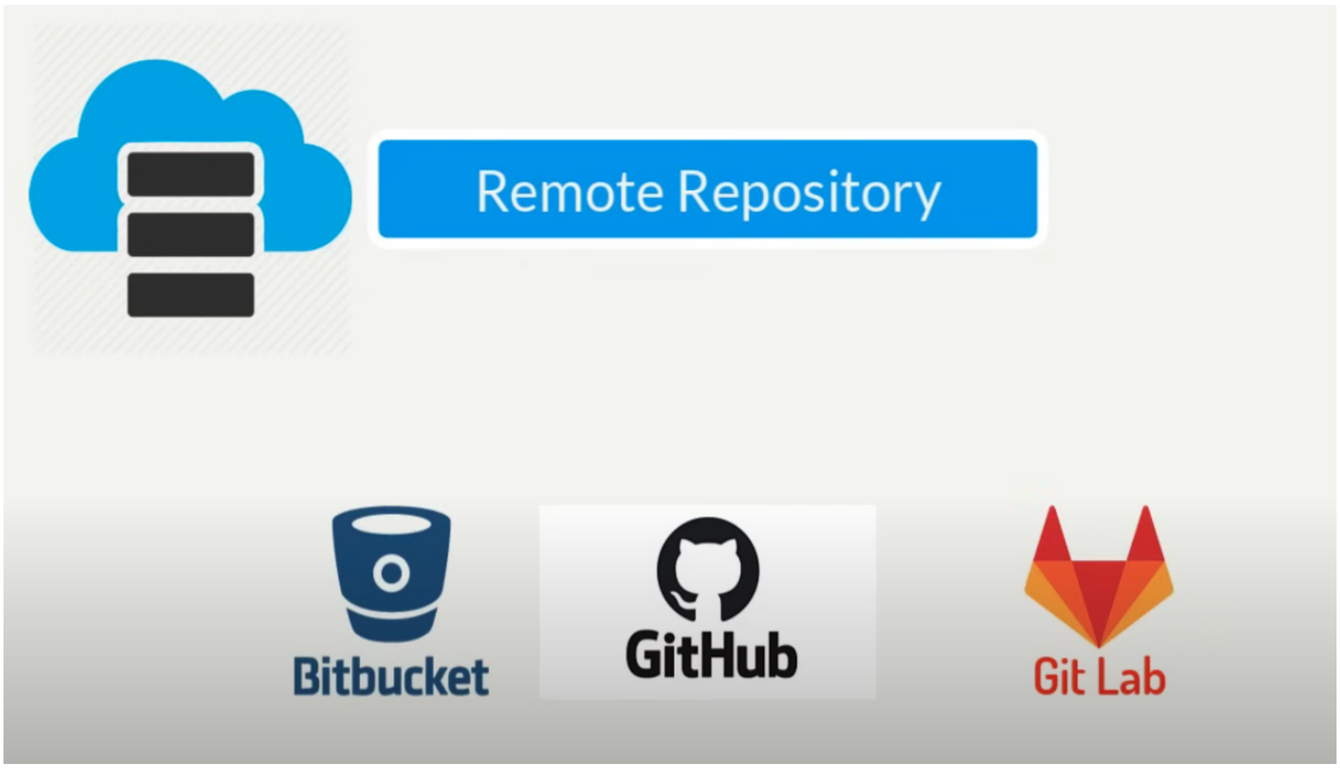
Follow below doc for SSH configuration:
https://support.atlassian.com/bitbucket-cloud/docs/log-into-or-connect-to-bitbucket-cloud/#Personal-SSH-keys

Cloning a Private Repo in Bitbucket over HTTPs

```
# git clone https://yogeshgupta0246@bitbucket.org/yogeshgupta0246/new-test-repo1.git
```

For this we need to create an App Password:
https://support.atlassian.com/bitbucket-cloud/docs/create-an-app-password/


Remote Repository
pull
PUSH
Local Repository
pull
Commit
Local System
Staging area
pull
Add
Working Directory



GUI Tools for git

- GitKraken → https://www.gitkraken.com/
- Sourcetree → https://www.sourcetreeapp.com/
- TortoiseGit → https://tortoisegit.org/
- Github Desktop → https://desktop.github.com/

Note: Why CLI?

Note: System < Global < Local

https://github.com/joshnh/Git-Commands

# git status (To check the git status)

→ Add some files in repo directory
Red - working directory

# git add <file name> (To add single file in staging area)
# git status
→ We can also use # git stage command instead of git add

```
# git commit -m “any comment” (To commit the files in local Repo)
# git status
```

```
# git log (To check the commit and commit-hash)
```

```
# git add . (To add all files in staging area)
# git status
```

## git restore

Restore means when you bring down the **Staging** area to **Working Directory** or we can say unstage the files.

→ Now edit any previous file

```
# git status
# git add <file name> (To stage the file)
# git restore --staged <file name> (To unstaged the files)
# git status
# git branch (To check the current branch and all as well)
# git checkout -- <file name> (To restore the file from Local Repo)
# git status
```

## git reset

Update any file and add it to staging and then to Local repo. Now if we want to move the Head to lover SHA Hash then we can use **git reset**

```
# git reset --soft <SHA Hash>
```

After reset, the file changes will remain to staging area. Let’s get rid of these changes by using **git restore** command

```
# git restore --staged . (To unstage the files)
# git checkout -- . (To restore files from Local Repo)
```

## List the files in Stage area

Move any file using mv command

```
# git ls-files (To list the files in Stage area)
# git status
# git add .
# git status (it will display renamed)
# git ls-files
```

Now delete any file and git status and git add it

## git stash

```
# vim om1
# vim om2
# git status
# git add .
# git status
# git ls-files
# git stash save <any name> (To save files in stash)
# git stash list (To list stashes)
# git stash apply <stash no> (To restore the files from stash)
# git stash drop <stash no> (To delete the stashes)
```

## git branch
Branch is the part of everyday development process.

```
$ git branch (To check the branches)
$ git branch <branch name> (To create a new branch)
$ git checkout <branch name> (To switch branch)
# git branch -d <branch name> (To delete a branch)
```

## git push

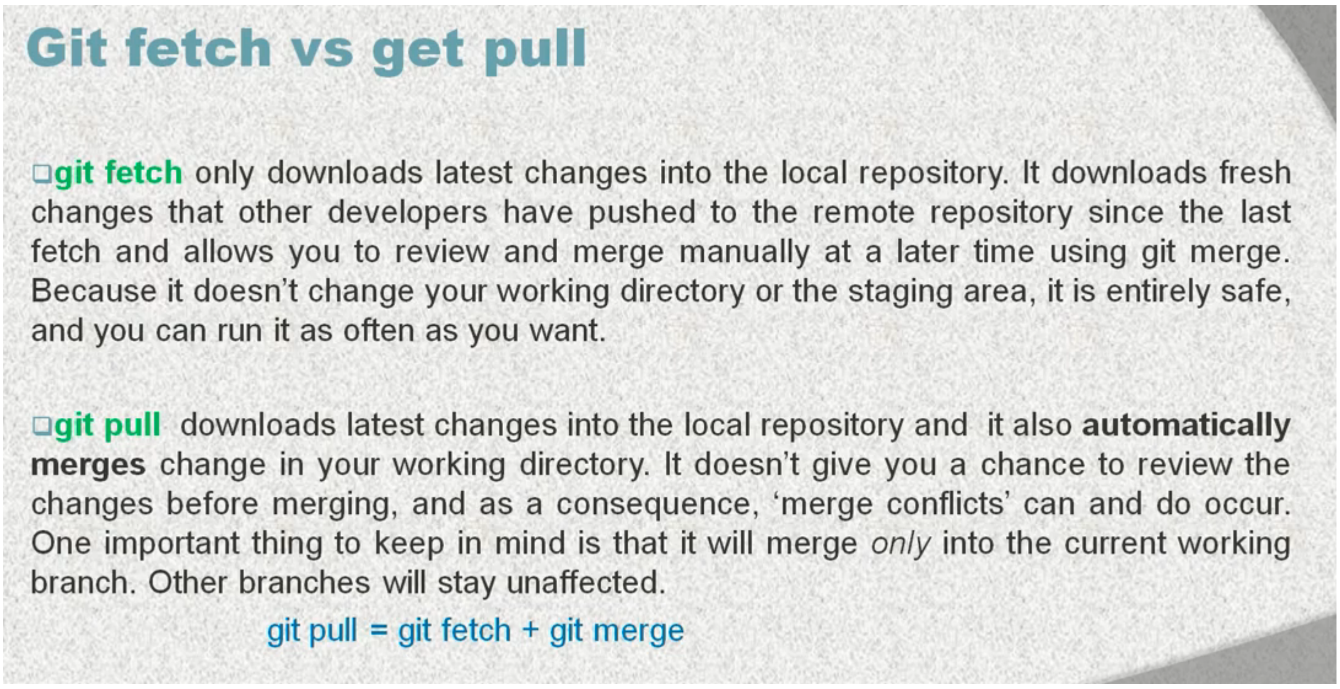
```
# git push (To push the changes in remote repo)
# git push --set-upstream origin <branch name> (To upstream a new branch)
```

## git merge
→ Go on main branch on which we have to merge the changes (master)

```
# git checkout master
# git merge stage
# git push
```

## git fetch & git pull
# git fetch (git fetch is **a primary command used to download contents from a remote repository**)

```
# git merge
```

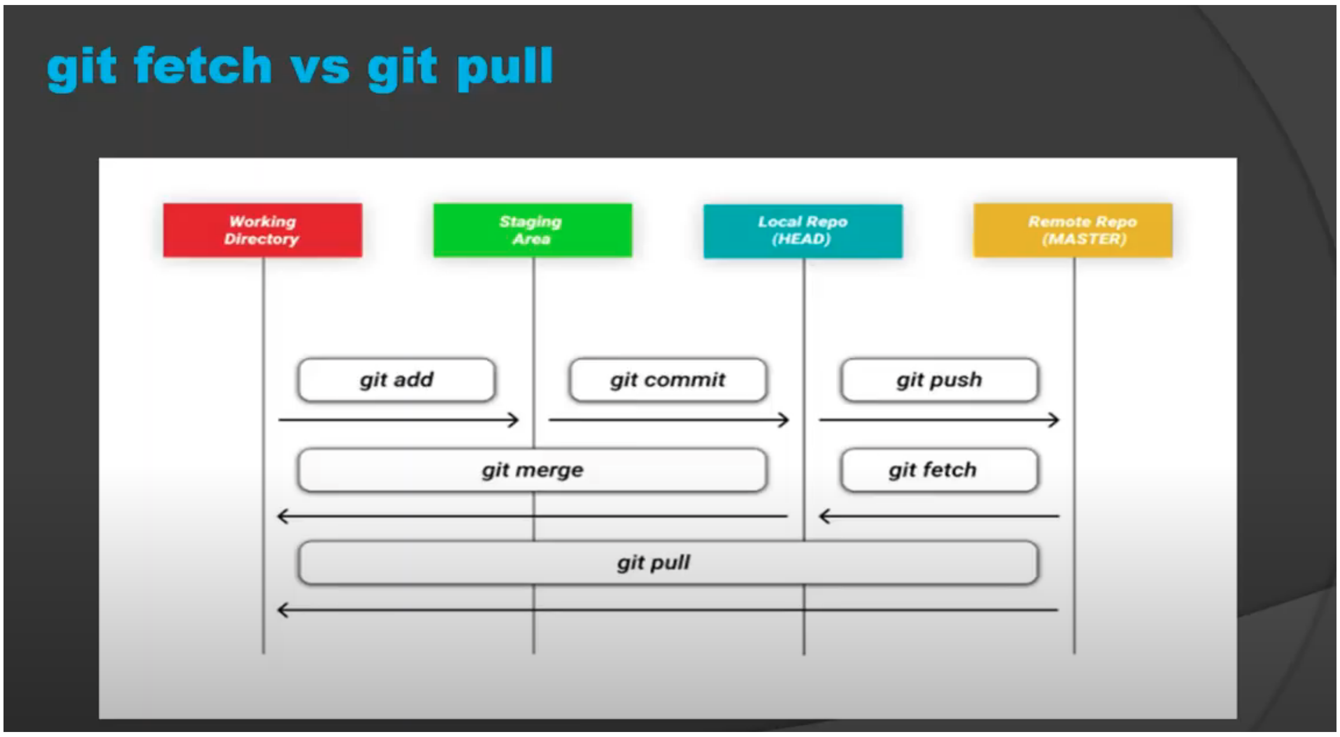


Git fetch vs get pull

❑**git fetch** only downloads latest changes into the local repository. It downloads fresh changes that other developers have pushed to the remote repository since the last fetch and allows you to review and merge manually at a later time using git merge. Because it doesn't change your working directory or the staging area, it is entirely safe, and you can run it as often as you want.

❑**git pull** downloads latest changes into the local repository and it also **automatically merges** change in your working directory. It doesn't give you a chance to review the changes before merging, and as a consequence, 'merge conflicts' can and do occur. One important thing to keep in mind is that it will merge *only* into the current working branch. Other branches will stay unaffected.

git pull = git fetch + git merge

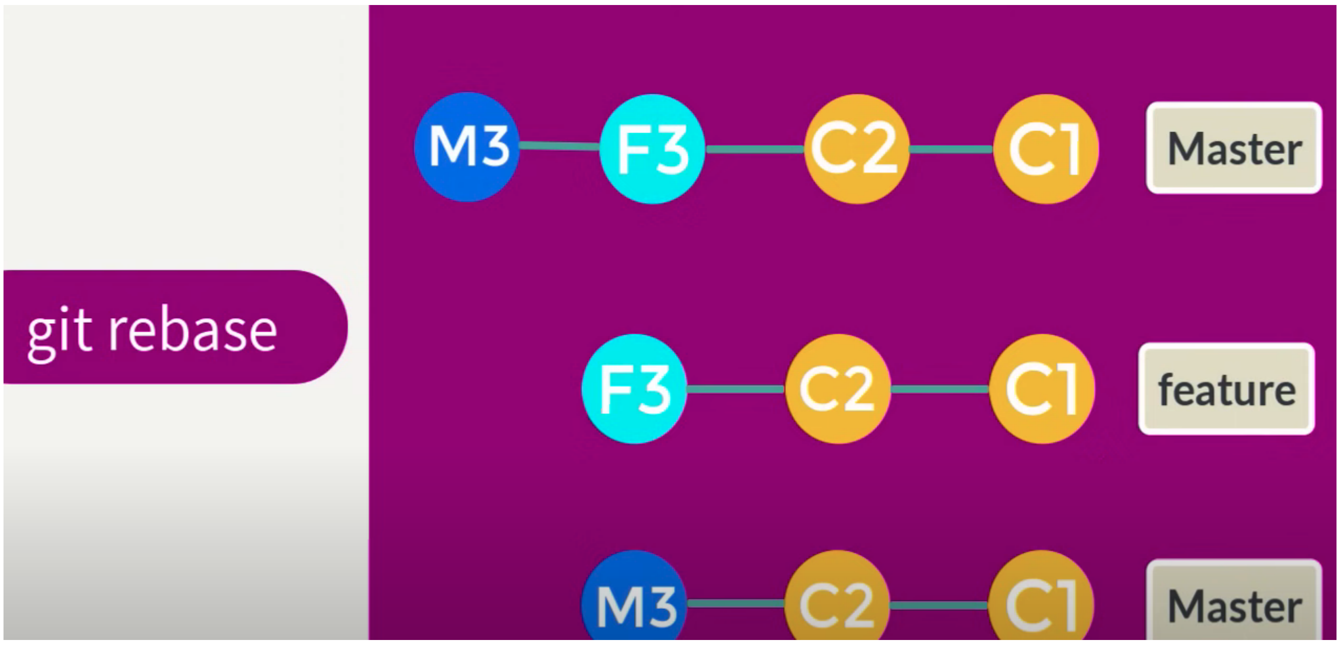


git fetch vs git pull

# git pull (The git pull command is **used to fetch and download content from a remote repository and immediately update the local repository to match that content**)

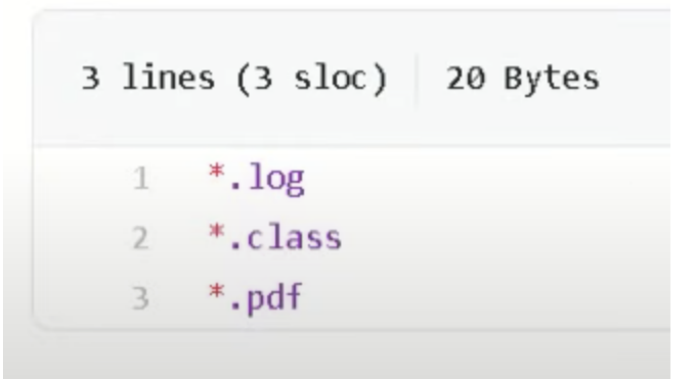
# git reset –hard (**resets the current branch tip, and also deletes any changes in the working directory and staging area**)

# git rebase (Rebasing is **the process of moving or combining a sequence of commits to a new base commit**. Rebasing is most useful and easily visualized in the context of a feature branching workflow)



## .git ignore

gitignore file is **a text file that tells Git which files or folders to ignore**, . gitignore file is usually placed in the root directory of a project.



```
3 lines (3 sloc) | 20 Bytes
*.log
*.class
*.pdf
```

## Create a branch using git and push it into remote repository and then delete it

# git branch
# git branch <new branch name>
# git branch
# git checkout <new branch name>
# git branch
# git push
# git push --set-upstream origin <branch name>
# git checkout master
# git branch -d <new branch name> (To delete branch locally)

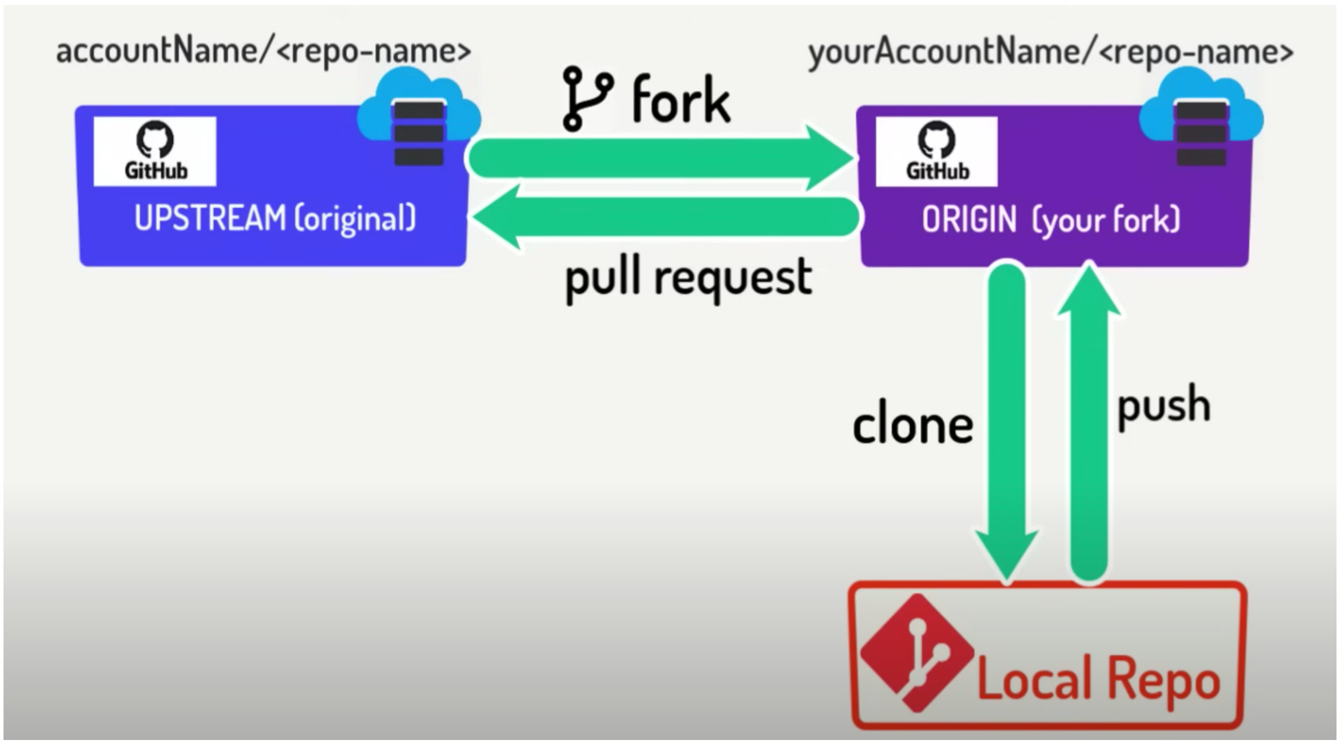
# git push origin -d <new branch name> (To delete branch remotely)
If you get the error below, it may mean that someone else has already deleted the branch.

```
error: unable to push to unqualified destination: remoteBranchName The destination
```

Try to synchronize your branch list using:
# git fetch -p

The -p flag means "prune". After fetching, branches which no longer exist on the remote will be deleted.

##git fork
git fork is **a rough copy of a repository**. Forking a repository allows you to freely test and debug with changes without affecting the original project.



## git diff
The git diff command **helps you see, compare, and understand changes in your project**.
→ clone a repo and edit any file and run git diff command, you will see the changes
# git diff <file name>

## git rm
git rm can be used to remove files from both the staging index and the working directory.

```
# git rm -f <file name> (To remove any file from working directory & staging area)
# git rm -- cached <file name> (To remove file from staging area only)
```

Read few GUI settings for Bitbucket & Github

What is Pull request?
A pull request – also referred to as a merge request – is an event that takes place in software development when a contributor/developer is ready to begin the process of merging new code changes with the main project repository.

How to generate a PR, approve & merge it?

## git cherry-pick
git cherry-pick is a powerful command that enables arbitrary Git commits to be picked by reference and appended to the current working HEAD.

```
# git cherry-pick <commit no>
# git push
```

How to resolve conflicts in PR in Bitbucket?