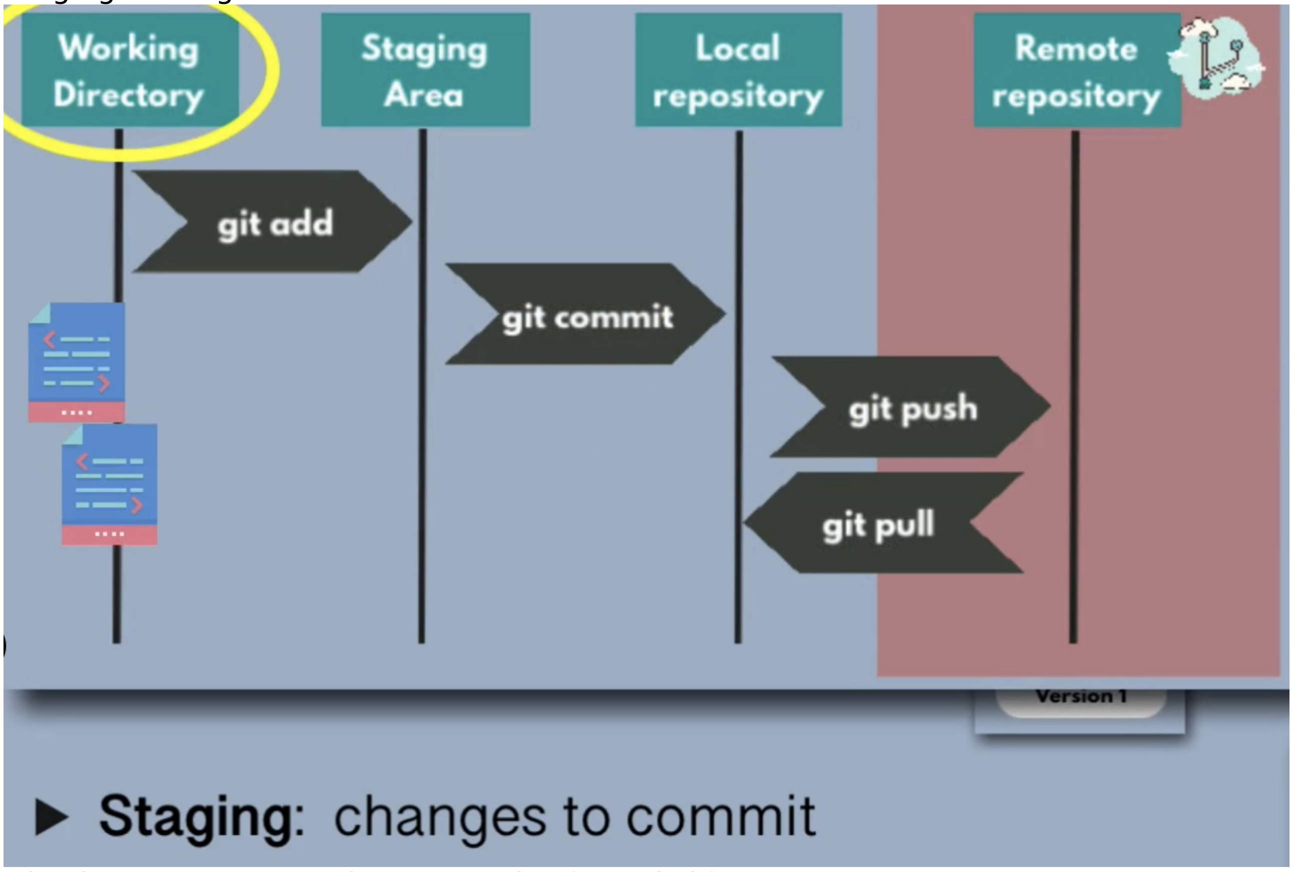
Version control ?
Developers working on same code
How do u share code among on the team
Code is hosted centrally on the internet
Every developer has a entire copy of the code locally
U fetch that code from that remote repository and push those changes in the repository .

Merge conflicts : when the same line is change git cat fix that alone ,
Best practice :push and pull ofter from remote repo.(**continuous integration)**
Integrating your code changes frequently.
Breaking changes don't effect u until u pull the changed code
Version control keep the history if the changes .
Every code change and file is tracked you can revert commits, and each change
Labelled with commit message.

Git is the most popular version control tools , older one is SVN.:
Remote git repository: where the code lives.
Local git repository: local copy of the code.
History : git log
Staging : changes to commit



▶ **Staging**: changes to commit

Git client : to execute git commands . (CLI of git)
Git hub , git lab are the platform that host your repositories
Companies have there own git servers

Git add <filename>
For git add . (Add all file in staging area)

To make a folder a git repository in local system : git init

Push code form local to remote repo ==> Git remote add origin url(of remote git repo)
:
Now we need to connect the branches of both local and remote repo ==> git push —set-upstream origin master
All this information is added to .git folder
By removing .git folder // connection is lost with the remote repository .

Concept of Branches :
Concept of branches exist in order to cleanly divide the work of different developers .
Best practice is to create a separate branch for each feature and each bug fix.
Developers can now commit without worrying to break main branch and once fully implemented and tested locally now I can merge it to master branch .
Also new feature can be deployed with new release version .
With branches u can have a stable main or master branch

Git branch : let u see all the branches
Git pull to see the created branch on CLI
To change to new_branch : git checkout bug fix/user_auth_error
Now u will have the local copy of that branch which is connected to that remote branch .
Making changes in this new branch will only effect this branch only .

How to create new branch locally in CLI ?
Go to the master :
Git checkout -b feature/database-connection
Now remote repo does not know about this feature branch =>
Git push —set-upstream origin feature/database-connection (this will push the branch to the remote repository)

Common practice :
Master and a dev branch :
Dev branch : intermediary master (they create there feature or bug fix branch and merge in dev branch) and from dev to master .
Master : ready for production

Master vs master +dev branch :
Only master branch for continuous integration / delivery .
Because every this a pipe line is triggered when ever feature / bug fix code is merged into master .
This pipe line will do : test build deploy .

Build and deliver the change after every merge to the master branch (modern way).

MERGE REEQUESTs:
When developed are done with feature implementation or bug fix
Best practice : review code changes before merging
Master should be stable and ready for production and has no half implemented or broken things inside as much as possible .

Making a Pull requests : review your feature branch if u have fully worked on it .
: this will ensure if is is ready to merged to master .

0,2 : first no.says these are the change my master has that my feature branch does not have :

Second no. says these are the changes here 2 changes I have made in feature branch that are not in master branch .
So do merge request : asking the master : I can merge to u or not .

```
git checkout feature/branch
git merge master
git push origin feature/branch:master
```

Deleting branch :
Deleting branch after merging :
:create new branch when needed :
Advantage of that is u don't end up having hundred of useless branches.
Which one is merged ? Which one is active ? Which one has bugs? No one knows.

To to the main branch , git pull , git branch -d name_Of_The_Branch_u_want_to_delete .

///
CASE 1: when there are two uses working on the same branch one change from the local and one do changes from the remote repo .
And they both don't know about each changes.
And when from local repo u do git add . , git commit . And then git push (it will show error) , because from remote repo already something is committed . So first u need to git pull to get the change to local repo . Then only u can push new changes to remote repo .

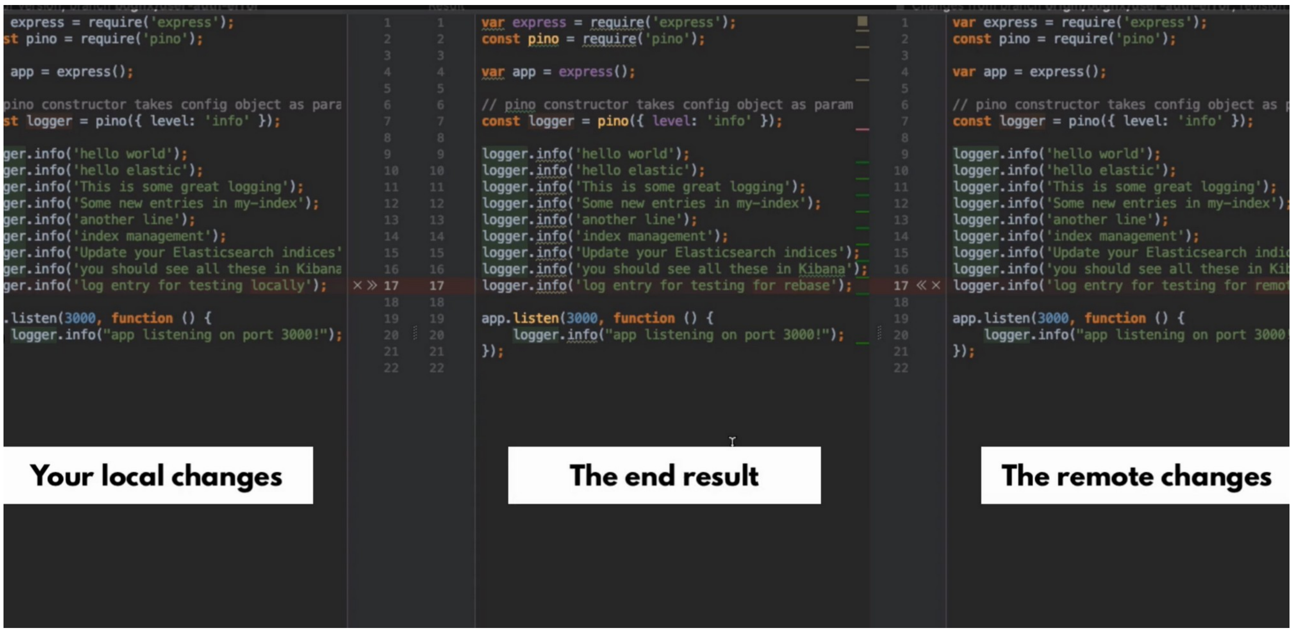
Now if u git push : there will be two commits one my commit and one merge commit .

These merge commit refers to that u have pull the change before u push . To avoid those merged branch every time we get changes from the remote branch .

So solution is using git pull rebase : git pull -r : this pulls the change from the remote branch and it stacks out commits on top of that so there is no merge branch commit in btw.

When remote branch has something that we don't have locally and we try to push from local to remote // sol: u have to pull that first then push and to git pull -r to avoid extra merge commit .

Case 2:: make change locally do git add . , git commit -m , make change the same line do commit changes .
Two developers can change the exactly same file same line in parallel this will cause merge conflicts.
If I do git pull -r // this will give merge conflict error . Because git does not know which version u want to take right . .
You must tell git which change to make and which to remove .



```
 express = require('express');
st pino = require('pino');

 app = express();

pino constructor takes config object as para
st logger = pino({ level: 'info' });

ger.info('hello world');
ger.info('hello elastic');
ger.info('This is some great logging');
ger.info('Some new entries in my-index');
ger.info('another line');
ger.info('index management');
ger.info('Update your Elasticsearch indices'
ger.info('you should see all these in Kibana
ger.info('log entry for testing locally');

.listen(3000, function () {
 logger.info("app listening on port 3000!");
```

```
var express = require('express');
const pino = require('pino');

var app = express();

// pino constructor takes config object as param
const logger = pino({ level: 'info' });

logger.info('hello world');
logger.info('hello elastic');
logger.info('This is some great logging');
logger.info('Some new entries in my-index');
logger.info('another line');
logger.info('index management');
logger.info('Update your Elasticsearch indices');
logger.info('you should see all these in Kibana');
logger.info('log entry for testing for rebase');

app.listen(3000, function () {
    logger.info("app listening on port 3000!");
});
```

```
var express = require('express');
const pino = require('pino');

var app = express();

// pino constructor takes config object as p
const logger = pino({ level: 'info' });

logger.info('hello world');
logger.info('hello elastic');
logger.info('This is some great logging');
logger.info('Some new entries in my-index');
logger.info('another line');
logger.info('index management');
logger.info('Update your Elasticsearch indic
logger.info('you should see all these in Kib
logger.info('log entry for testing for remot

app.listen(3000, function () {
    logger.info("app listening on port 3000
});
```

How to solve conflict : ??
(IDE INTELLIJ)Go to git then go to resolve conflict ., and both developed talk to each other and decide what to do .

And run : git rebase —continue
Now u can push charges to remote : git push

.gitignore file :
Add .gitignore file to the project root directory.
To exclude certain folders or files from git to be tracked.
Something that other developers does not need to down load in there project
Eg .idea folder (specific to intillij ide) . Build folder where compiled files are located
Add .idea/* , build/* in .gitignore folder .
Also node modules // dependencies that every one should install locally . We don't need them as a part of the code.

We need to tell git forget about the folder we are not tracking it any more :
We want to ignore that from any git activity :
For that we need to remove that from git cache .

Git rm -r —cached .idea (-r mean recursive)
Now git add .
Git commit -m"added git ignore file"
Git push

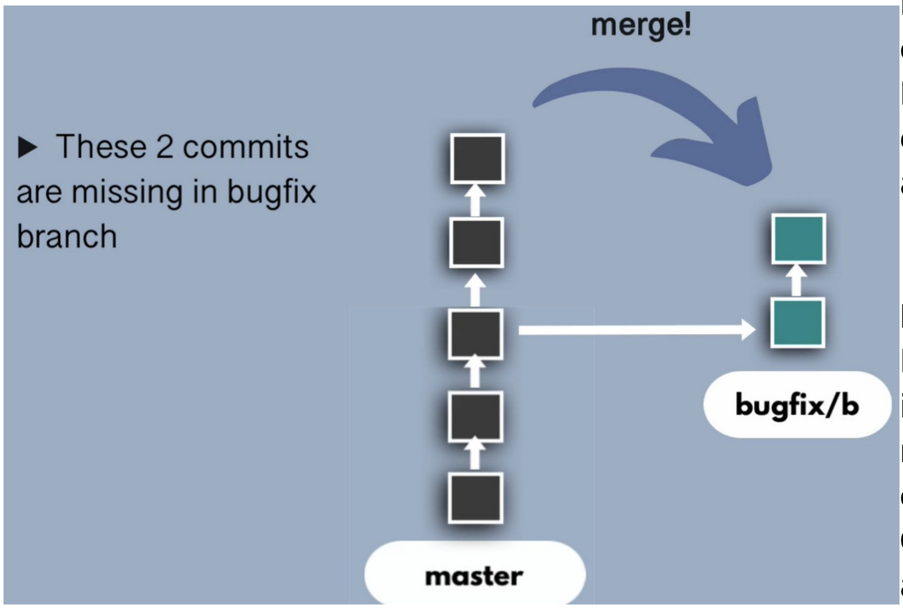
Save work in progress changes (stash):
Eg : do local change in a brach I don't want to commit those change
Git checkout master // it will not allow u to switch as u have some local changes
// u just cant leave the branch If u have local changes solution for this is —->
Git stash // this will save the change for later .
Git status now // I have no change that are active .
Now u can : git checkout master .
Now how do I get my changes back :? Git stash pop .

Use case hide changes temporary to check it that code works without my code change .
Bring back the code in the working directory git stash pop .

Going back in the history :
Git log : each commit has unique hash
We can go back to the specific project version .
Git checkout commit_hash
This is detached head
Go back to up to date state : git checkout bugfix/user-fix-error
HEAD is the latest commit stage in branch

Undoing commits (git revert, git reset):
Make changes in local repo :
Git add .
Git commit -m
Git status : shows commit Is only locally .
Before I pushed it I release the change I made are wrong .
Git log : **HEAD** last commit :
Git reset —hard HEAD~1(this no represent how many last comments I need to reset)
Now the last commit is gone and
Current last commit is the previous one .
hard(this reverts the comments and also discard the changes that where in commit)

Now I want to correct the change I made .
I want to still do more changes or correction on the commit . ?
Git reset —soft HEAD~1(to keep the changes in the working directory)
Change is still there but commit is gone . Commit got reverted .



Now u can correct the code .
Now re add and re commit the change again .

Revert:
I forget to edit some line in local repo and I remember it after committing .
Get commit —amedn(ammend)// this

instead of creating new commit this will merge this new changes I remember now to the last commit I made .
Now push that to remote repos .

Now I realise that commit is extremely wrong ,or it broke something now I have already pushed it to remote repository .
Now I want to undo the commit in remote repository as well .
Solution is : git reset —hard HEAD~1(hard because I want to discard all the changes I want to start from scratch) now my local has the change but remote still has the commit.
Git push —force (we have reverted something that remote repo already has of just git push will not allow us to push our changes)

Dont do this in master or dev branch only when working alone in a branch : because other developers are pulling form that repo which has your wrong code previously

Go to the master branch :
Git revert commithash: create a new commit to revert to old commit's changes . (The line that we deleted has been added back).
Git reset commit hash: removed old commit

Above pic : merging :
U want to pull change from your master branch to bug fix branch .

Go to local master branch
Local master branch should be up to date : so git pull in local master branch.
Git checkout bugfix/aur-auth-error
Git merge master(source from where u want to take the changes from)

Git for devops

1.when working with infrastructure as a code concept .
There are many configuration file of Kubernetes , terraform ,ansible
Bash , python script to automate some task

These files should be tracked _ history of changes .
Securely stored in one place .
So u need collaboration on those configuration file .
2. Ci cd pipeline :
Check out the application code , u test it u build the application etc.
U need to setup integration with build automation tools and git repository .
Eg : getting a commit hash for a specific commit , check if the changes have happened on front end or backend code .