**mvn clean compile**

Maven command: Cleans the project (removes the target directory with compiled files) and compiles the source code into bytecode, placing the results in the target/classes directory.

**mvn clean package**

Maven command: Cleans the project, compiles the code, runs tests, and packages the compiled code into a distributable format (e.g., JAR or WAR file) in the target directory.

**mvn spring-boot:run**

Maven command: Runs a Spring Boot application directly from the source code, starting an embedded server (e.g., Tomcat) without needing to package it first.

**git pull origin master**

Git command: Fetches and merges the latest changes from the master branch of the remote repository (named origin) into the current local branch.

**git add --all**

Git command: Stages all changes (new, modified, and deleted files) in the working directory for the next commit.

**git commit -m "coderefactor"**

Git command: Creates a new commit with the staged changes, using the message "coderefactor" to describe the changes.

**git push origin master**

Git command: Pushes the local commits from the master branch to the master branch of the remote repository (named origin).

**git push origin master -f**

Git command: Force-pushes the local master branch to the remote master branch, overwriting any conflicting history on the remote. Use with caution as it can delete others' work.  
  
and after push if you have made some new change or added new class file then use below command  
**git add –all**

**git commit -m “New class added”**

**git push origin master -f OR git push origin main -f**

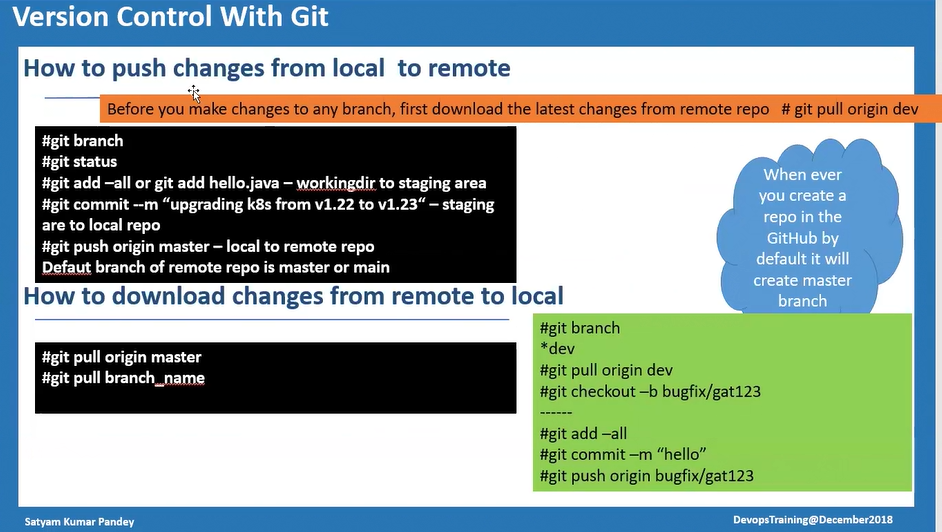
* To find How many branches in local repo and in which branch are we under local repo
* **git branch**
* Do we have the Dev, Test, Pre Prod, Prod, etc. all other branches present in local
* **git fetch origin dev (*if you want to fetch dev branch then dev else preprod or prod, etc.*)**
* **git checkout dev (*based on fetching which origin or branch we are fetching checkout that branch*)**
* **git pull origin dev (*to fetch all dev remote source code in our local system*)**
* **git branch (*to verify on which branch we are*)**
* To track the all the changes
* **git status (***it will track the change so before pull, commit and push we can just use it to verify whether what change is done in local system and* ***it will show the names of the file* *where changes are happened*)**
* To create a branch where developer will implement the code, test the code and then create an MR (Merge Request) and review and then push that code on the feature/bus branch and then push to dev branch.
* **git checkout -b feature/bus**
* **(*Use of git checkout: it is use to go from one branch to another branch, and also help to create a new branch from existing branch)***
* ***(Use of git checkout -b: (hyphen) -b it helps to create a new branch from existing branch*)**
* ***(Use of git checkout -b feature/bus: -b it helps to create a new branch from existing branch or which branch you open*)**
* To identify the changes between file changes on branch
* **git diff**
* To revert all the changes or to do undo all the changes in local we are using below command
* **git stash (*under local only after commit it will go ahead in another branch*)**
* Can we undo your changes from staging area?
* **Yes, we can do**
* Can we undo your changes from local repo?
* **No**
* We can add only one file also if we have to pass one particular file from multiple file change using full path of the file name
* **git add src/main/java/com/example/makemytrip/flights.java   
  (copy the full path of the file and paste it as above file path is added)**

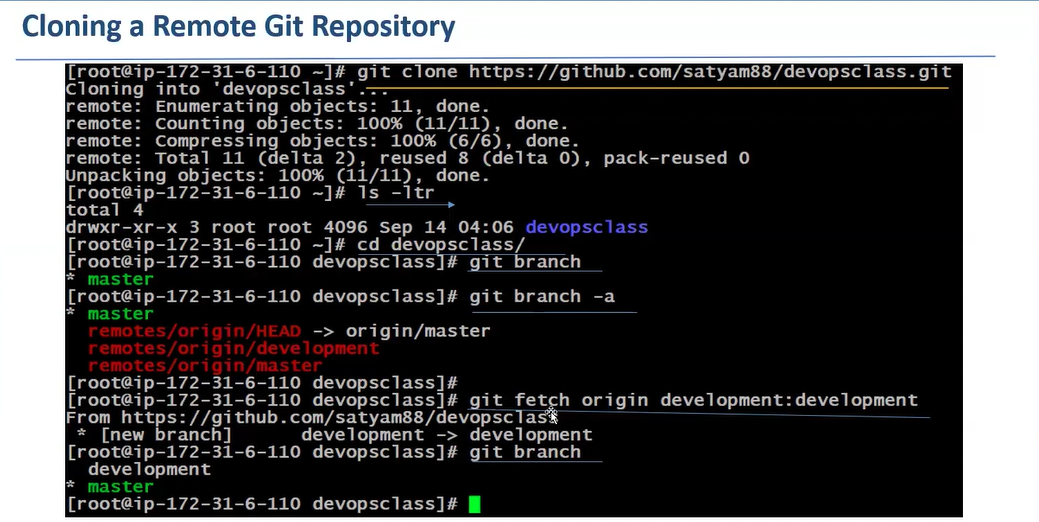
**In short**

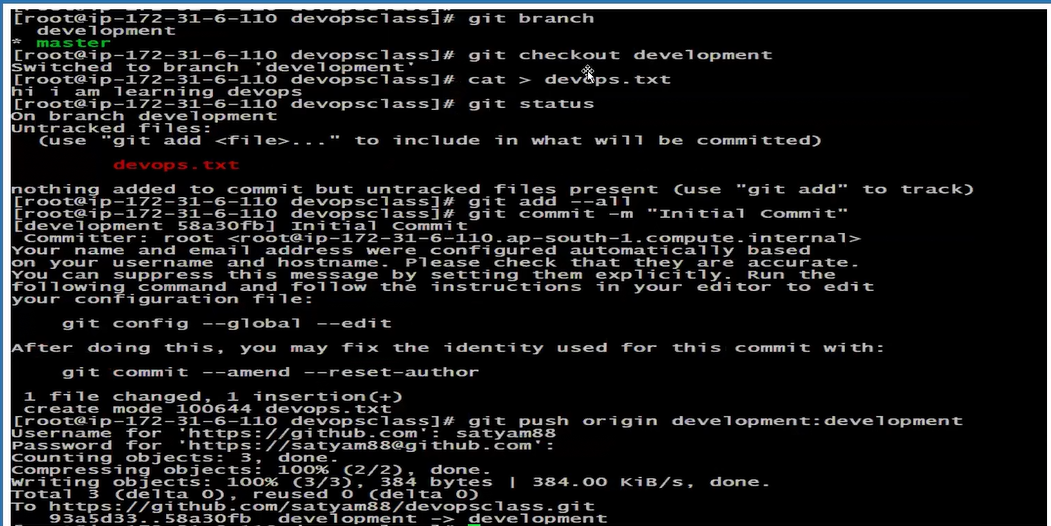
|  |  |
| --- | --- |
| **Git Command** | **Use** |
| **git status** | **List all the Modified Files** |
| **git branch** | **Show the current branch** |
| **git checkout** | **To change the branch from current branch** |
| **git checkout b** | **To create a new local branch** |
| **git add --all** | **Park the changes to working directory to staging area** |
| **git commit -m** | **Park the changes from staging area to local repo** |
| **git push origin master** | **Push the changes from local repo to remote repo** |
| **git push origin master -f** | **Force full push** |
| **git fetch origin dev** | **Download the branch from remote repo to local repo** |
| **git pull origin dev** | **Synch/download the changes from dev branch of remote repo to local dev branch** |
| **git diff** | **Shows the difference** |
| **git stash** | **To do undo or revert all your changes** |
| **Git branch -D branchname** | **Delete the local branch only not dev, prod (not to delete from remote do it only local system so that we can fetch it from remote by using git checkout dev OR preprod)** |
| **git cherrypick(interview prespective)** |  |
| **git clone** | **Used to download repo from remote to local** |

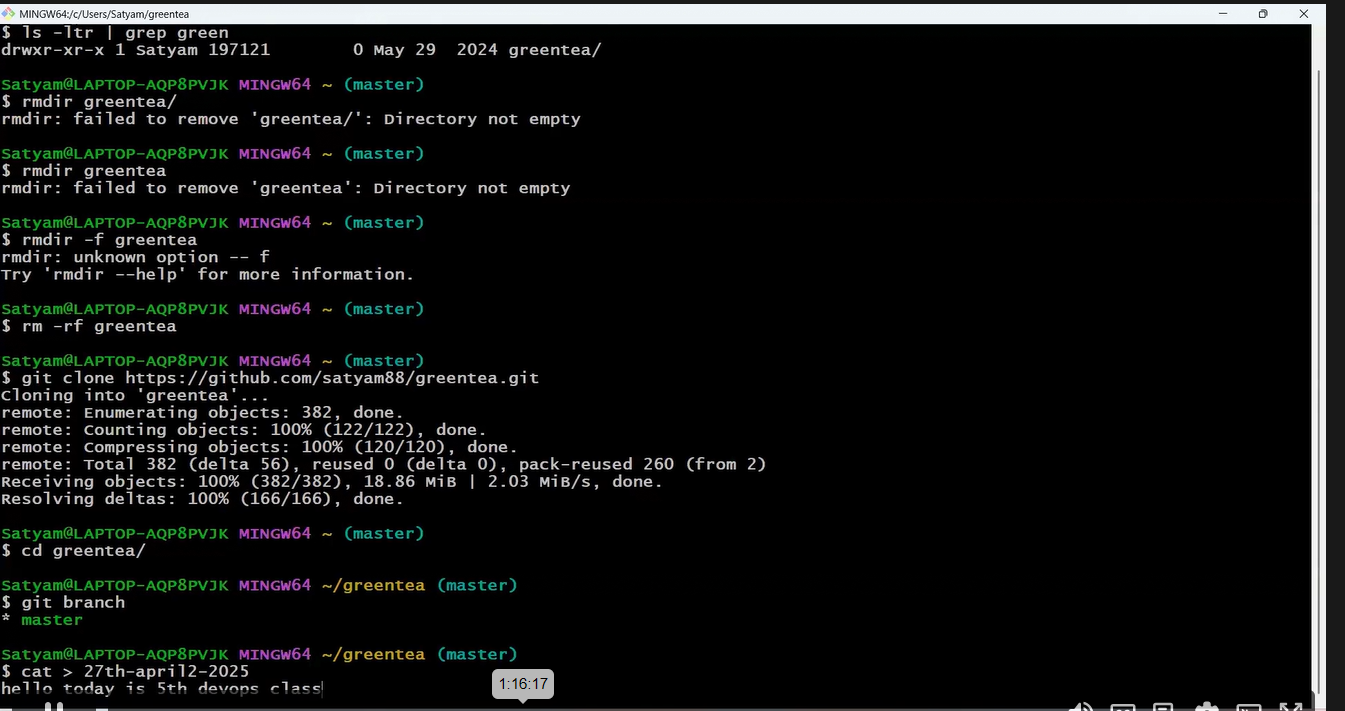
|  |  |  |
| --- | --- | --- |
| **Microservices Architecture** | **Maven** | **Git** |
| Java Spring Framework | Maven is a build tool | Git belongs to family Source Version Controle 1.Git  2.Mercurial  3.Subversion  4.Perforce |
| Microservices means small programs | used to compile and package Java Code | Git is a command line tool which can be install in windows and mac |
|  | If we have Hello.java and we have to make Hello.class then we have to use mvn clean compile command | Git is used to track all the changes done by a developer |
|  | If we have Hello.class and we have to make Hello.jar then we have to use mvn clean package command | Git is used to push the changes from local repo to remote repo, Changes like Adding, Delete, Modified |
|  | Based on programming language build/compile tool will be different   |  |  | | --- | --- | | Programming | Build tools | | Java | Maven, Gradle,Ant | | .NET | MS Build | | Node.js | npm | | **Git is Distributed Source Version Control** Under Distributed source version control we have two copy one is in local and one is one in remote  After Installing Git it will divide in three part   |  | | --- | | Working Directory | | Staging Area | | Local Reps |   Git is implementation of Distributed source version control |

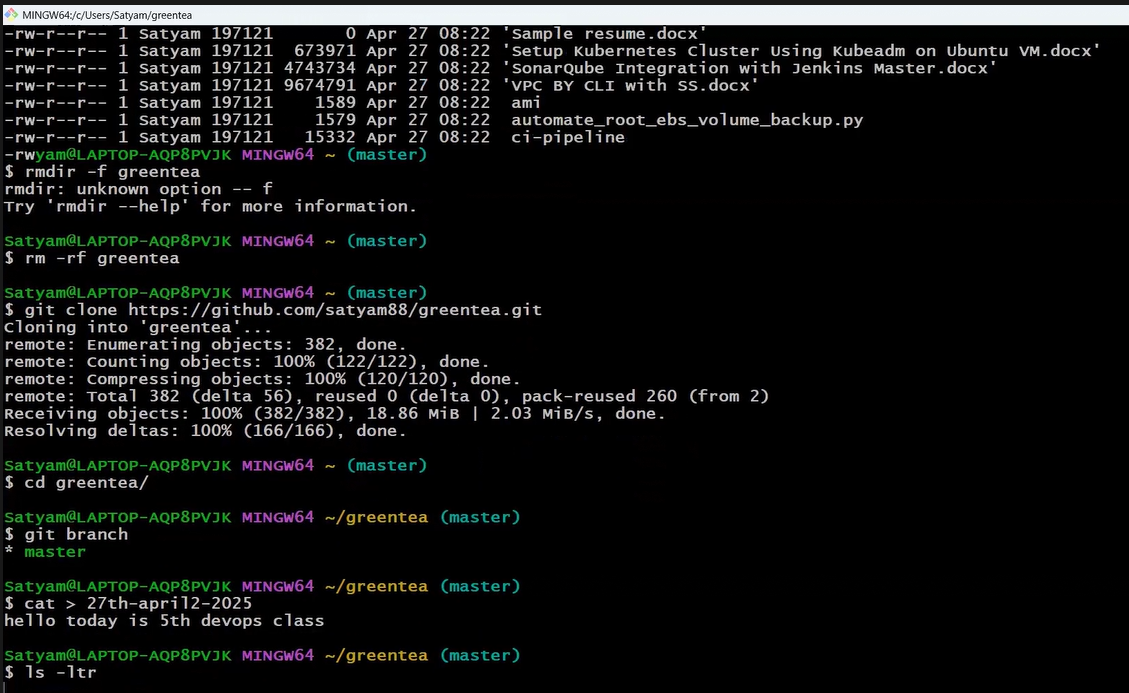
****

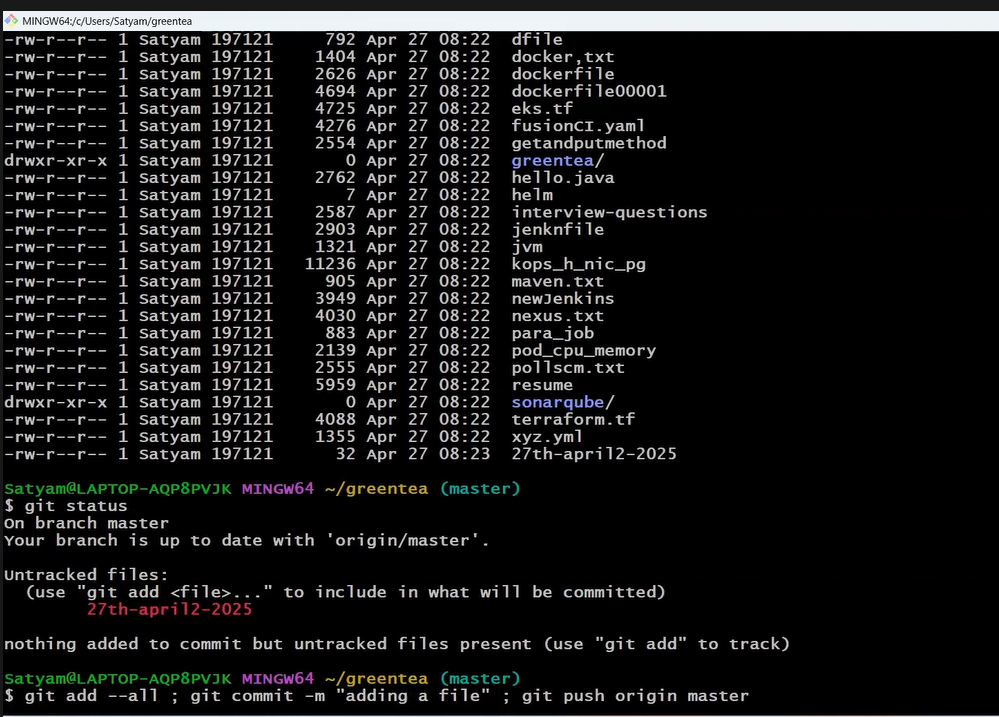
****

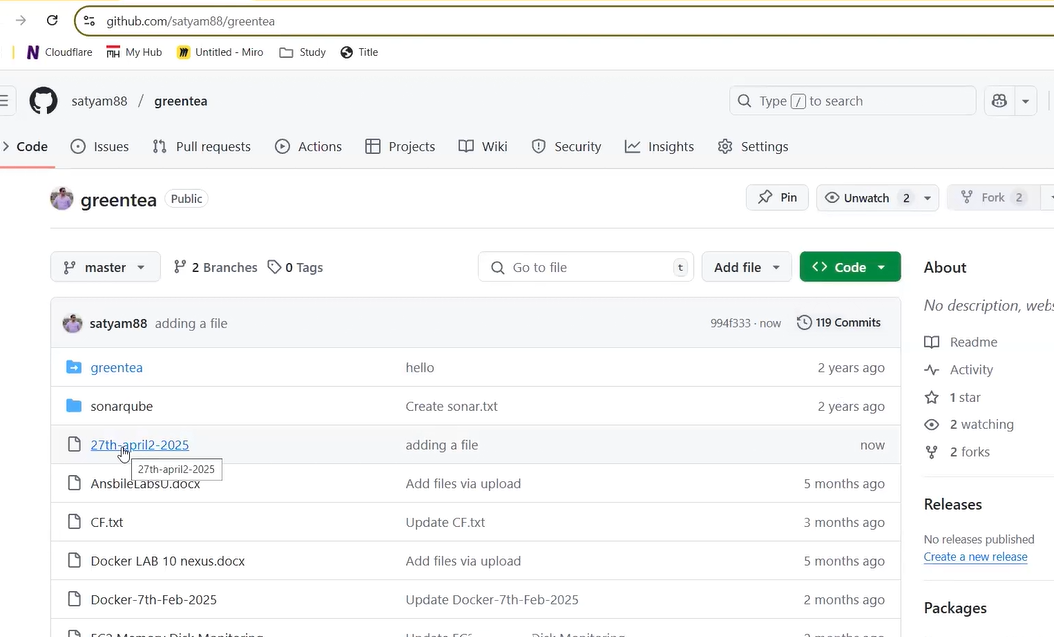


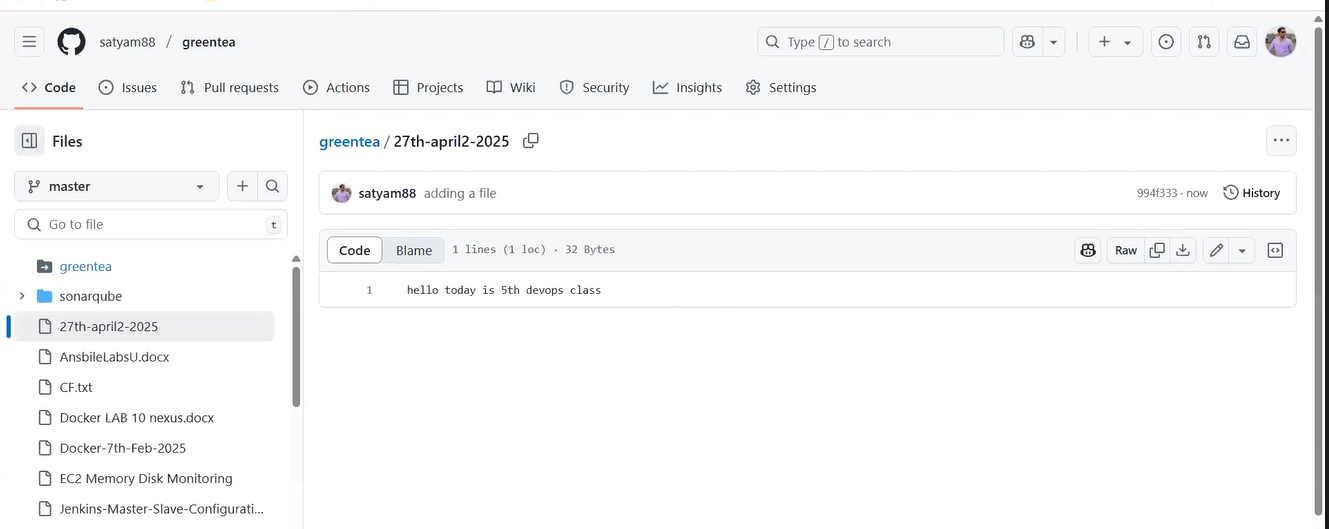
**git bash**

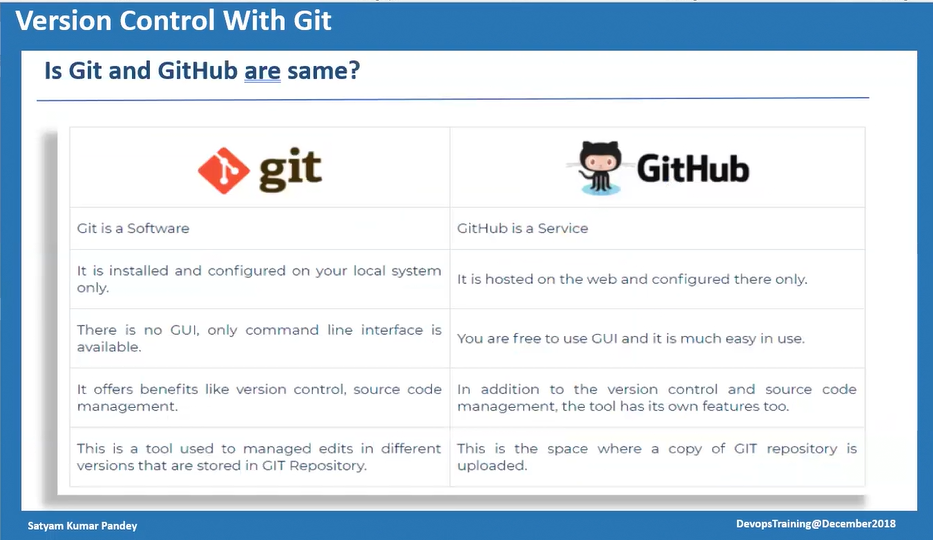
****

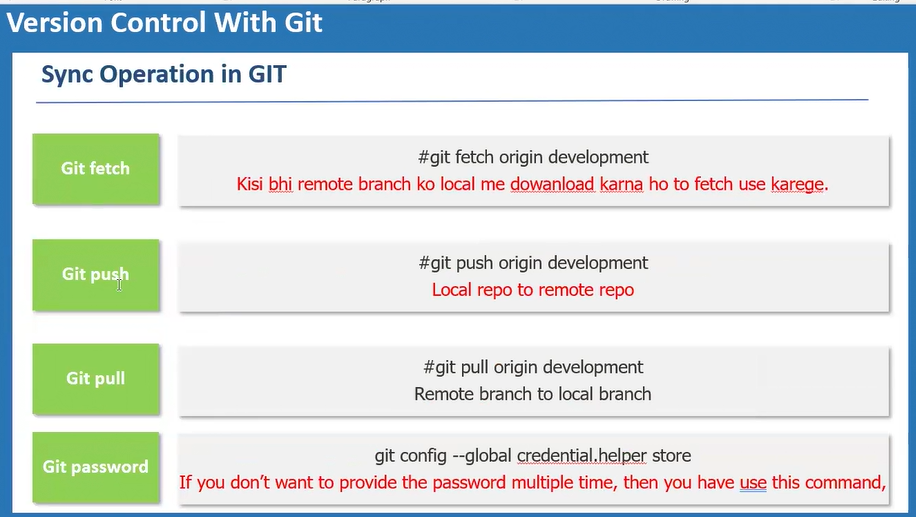
****

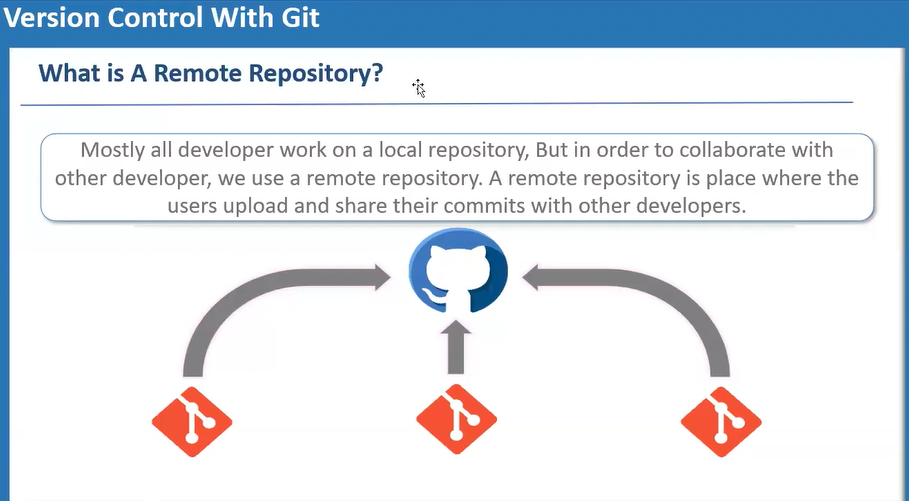
**ddf**

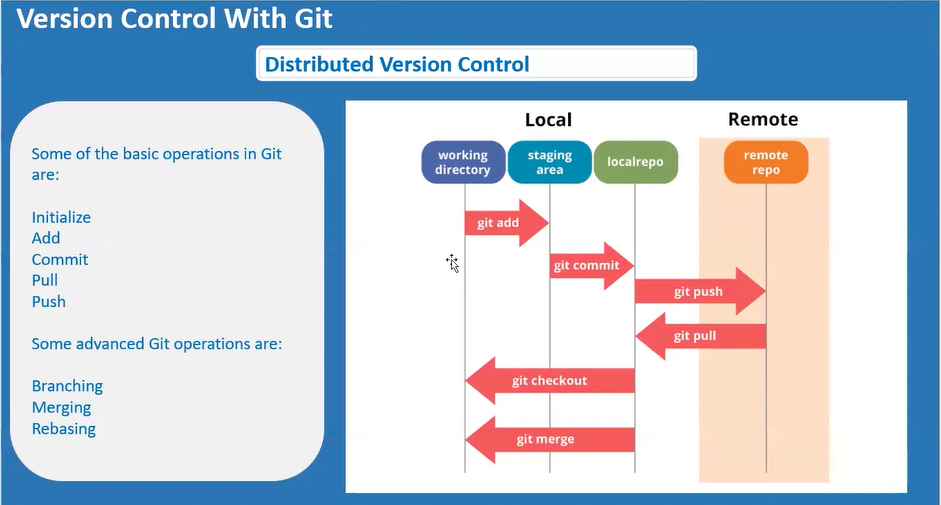
****

****

****

****

****

****

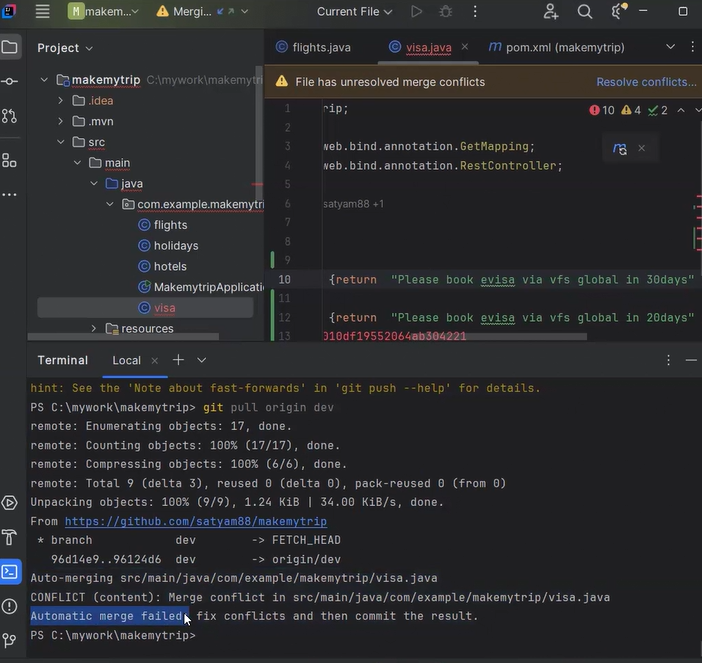
**Based on above SS**

When we make many changes on working directory and after that give command **git add –all,** after adding this command our working directory changes will go to staging area, after that adding the **git commit -m "CodeRefactor"** our changes will get commit on our local repo, and after that adding this command **git push origin master -f** our local repo changes will go to remote repo

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GitHub**   |  | | --- | | GitHub is used to Store the Source code | | Simple Word: -It’s an online service where anyone can store their source code on github | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Online Service to store code | |  | | --- | | Pull Request | | Branching strategy | | Code Promotion | | Branch Permission | | WebHook | |  | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Source Code Management | |  | | --- | | GitHub | | BitBucket | | GitLab | | AWS Code Commit | | Azure Repo | | |

**RESOLVING MERGE CONFLICT INTO GIT**

**If it showing like Merge conflict then some other developer may have already push the code on remote that is the reason while git push your local repo doesn’t have the latest updated code first we have to pull the latest updated code then commit and push every time for any single change also**

****

**If we use git pull origin dev then  
we have two part   
1. Automatic Merge Success: when changes are in 2 file then it will automatice merge successfully**

**2. Automatic Merge Failed: if changes are in same file and same line then automatic merge will failes**

**10/05/2025**

**Port number**

**http and https**

**dhcp**

**http error**

**http status code**

**3 way tcp hand shaking**

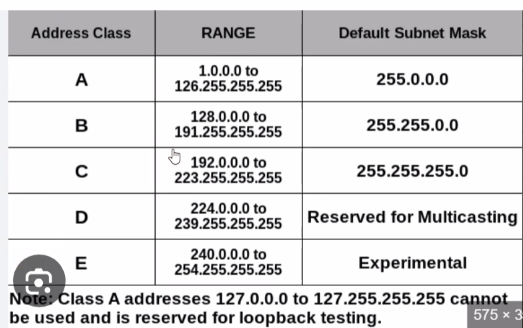
**OSI layer**

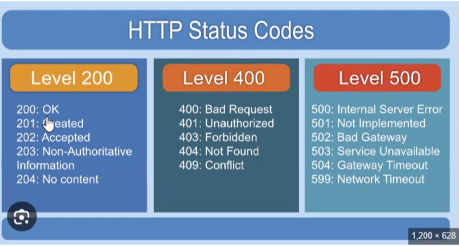
**Ipv4 and ipv5**

**Class and class b ip address**

**Cidr**

**Latency in project**

****

****

**Cloud is a abstraction of data center**

**Hardware just for understanding below  
Emc symmetrix**

**Netapp**

**Vmware**

**Juniper switches  
  
CI**

**\*jenkins**

**-Jenkins basic**

**-Jenkins installation**

**-jenkins tool configuration(maven, git and java)**

**-jenkins plugin installation**

**Types of jobs in Jenkins**

1. **Freestyle => basic level job limited use**
2. **Maven project =>**
3. **Pipeline => most used**
4. **Multibranch PipeLine**
5. **Parametrised Pipeline Job**

**43151 port number**

**PipeLine Jenkins Job**

1. **Scripted method – old method**
2. **Declarative methods – new method**

**To create Pipeline which method is used?  
=>Declarative method**

**Jenkins declarative Pipeline uses grovvy syntax**

**Pipeline block is on this {} curley braces**

**As example**

**Pipeline**

**{**

**Stages**

**{**

**Stage(a)**

**{**

**}**

**}**

**}**

**pipeline {**

**agent any**

**stages {**

**stage(check java version) {**

**steps {**

**“java –version”**

**}**

**}**

**stage(check mvn version) {**

**steps {**

**"mvn -- version"**

**}**

**}**

**stage(check git version) {**

**steps {**

**"git -- version"**

**}**

**}**

**}**

**}**

**\*entire pipeline will be written in pipeline block.**

**\*In jenkins we create the pipeline using Jenkins declartive pipeline**

**\*Jenkins declartive pipeline will use grovvy syntax**

**\*the concept of writting the pipeline via code is called as pipeline-as-code**

**\*there will be multiple stage in pipeline, the number of stage depend on project requirer**

**equiremnt, there is no fixed no of stages.**

**\*if first stage will fail then the pipeline will get abort and it will not run the second stage.**

**\*steps block inside the stage contains the command to be executed.**

**# A pipeline is a linear sequnce of stages, where every stage will do some work and pass the control to another stage.**

**Pipeline => stages => stage => steps**

**pipeline {**

**agent any**

**tools{**

**maven ‘mvn\_3.9.9’**

**}**

**stages {**

**stage(‘Code Compilation’) {**

**steps {**

**sh 'mvn clean compile'**

**// Example: sh 'dotnet test'**

**}**

**}**

**stage(‘’) {**

**steps {**

**sh 'mvn clean package'**

**// Example: sh 'dotnet publish'**

**}**

**}**

**stage(Code Packaging) {**

**steps {**

**sh 'mvn clean package'**

**// Example: sh 'dotnet publish'**

**}**

**}**

**}**

**}**

**Plugins need to install on jenkins**