**Learning Udemy course part-1**

**Git and GitHub in depth: Integrating Jenkins with Git**

* **What is Devops:**
* DevOps=Development + Operations.
* DevOps is a culture or process.
* To automate the application, build, and deployment process.
* By using DevOps culture, we can establish collaboration between development and operational teamwork.
* By using DevOps culture, we can speed up application delivery.
* DevOps is a set of Tools.
* Devops is the combination of best practices and tools which is designed to increase an organization company's ability to deliver applications/ projects and services faster than traditional software development processes.
* This speed enables organizations to better serve their customers and compete more effectively in the market.
* **Benefits of Devops:**

1. Free and open source
2. Faster development
3. Rapid delivery
4. Improved collaboration
5. Automated processes

* **Phases of the Devops lifecycle:**
* Plan
* Code
* Build
* Test
* Release
* Deploy
* Monitor
* Feedback
* **Devops workflow:**
* **Git:**
* Git is a distributed version control system that enables multiple developers to collaborate on software projects efficiently.
* It is free and open source.
* It was created by Linux Torvalds in 2005.
* Git helps you keep track of code changes.
* It is version control system means it is also known as source control is the practice of tracking and managing changes to software code.
* designed to handle everything from small to very large projects with speed and efficiency.
* It is also called source control system.
* Tracking and managing changes to source code overtime.
* **Goal of git:**
* Speed
* Data integrity
* Support for distributed
* Nonlinear workflows.
* **Two types of version control systems:**

1. Centralized version control system
2. Distributed version control system.

* **How to create GitHub account, create repository and clone a repository:**
* Go to browser
* Enter GitHub sign in
* Give email, username and password
* Account will be created.
* **Create repository in GitHub:**
* Click on new
* Enter repository name
* Click on public or private
* Click on README file
* Click on create repository
* Repository created.
* **Clone a repository:**
* Git clone <Repository URL>
* Clone means copy the files from the remote repository to our local machine.
* **Stages in git:**
* **There are three stages in git:**

1. Working area/untracked area (git add <filename>)
2. Staging area/ tracked area (git commit -m “message”)
3. Local repository. (files will see in local repository).

* **Config:**
* Git config --global user.name “enter name”
* Git config --global user.email “enter email”
* To see the config details git config --global --list.
* **Git pull:**
* In git pull is a command that fetches changes from a remote repo and merges them into the local repository
* **Git push:**
* Push means that sends committed changes from a local repository to remote repository.
* **Git log:**
* To see the commit history.

Git log --oneline(to see all commits)

Git show <commitid> (to see single commit in detailed view)

* Head: it means lastest commit happened on your system.
* **Branches:**
* Branching in git is the process of creating a new, independent line of development in a repository. This allows developers to work on new features, bug fixes, or other changes without affecting the main codebase.
* You can merge a branch into another branch using a pull request.
* **Different types of branches:**
* **Master:**

The master branch which will create contains production code all development code is merge into master after release.

* **Develop:**

This branch contains the pre-production code inside develop branch we have create number of branches.

* **Feature:**

a branch created for a specific feature or set of features.

* **Release:**

a branch created for a specific release or version of the software.

* **Bug fix:**

it fixes the bugs.

* **Create branches:**
* **Git branch <branch name> -** it creates new branch
* **Git branch-** it shows all the branches under the repository.
* **Git checkout <branch name>-** switch from one branch to another.
* **Git push --set-upstream origin <branch name>-** push a new branch to a remote repository and upstream tracks the information for branch.
* **Git branch -d <branch name> -** it deletes the branch
* **Git diff branch name..branch name-** it shows difference between two branches**.**
* **How to create pull request and merge one branch into another:**
* Merge data from feature branch to develop branch
* Select feature branch
* Select base branch is develop and compare branch is feature.
* Click on pull request
* Click on create pull request
* Add new description
* Click on create pull request
* Click on merge pull request
* Click on confirm merge.
* **Conflicts and how to resolve it:**
* For example, two branches created:
* Branch one is testforconflict
* Branch two is develop
* When you tried to merge testforconflict into develop branch conflict will come because it will try to merge all the files from testforconflict. The file name would be same so no conflict. Content is different than conflict will come.
* **How to resolve this conflict:**
* We can download one software that is **"tortoisegit".**
* By using tortoisegit we can resolve the conflict
* Open tortoisegit
* Right click on the conflict
* One popup window open
* Click on use this whole file.
* **Git clean:**
* The git clean command is used to removed untracked files and directories from a repository.
* Create file touch <file name>
* Create directory mkdir <directory name>
* To remove the created file and directory
* Git clean -f (it removes the file)
* Git clean -fd (it removes the directory and file also)
* **Undoing changes on existing file**
* This command will discard any changes made to the specified file and revert it to its previous state.

Git checkout -- <filename>

* **Git revert:**
* The git revert command is used to create a new commit that undoes the changes made in a previous commit.

Git revert <commit id>

Note: it works like undo.

* **Git reset**:
* It is used to reset the current branch head to a specified state and optionally reset the index and working tree to match.

Git reset --soft <commit id> (files not removed) remove the commit id from the logs from the history.

Git reset --hard <commit id> (files removed)

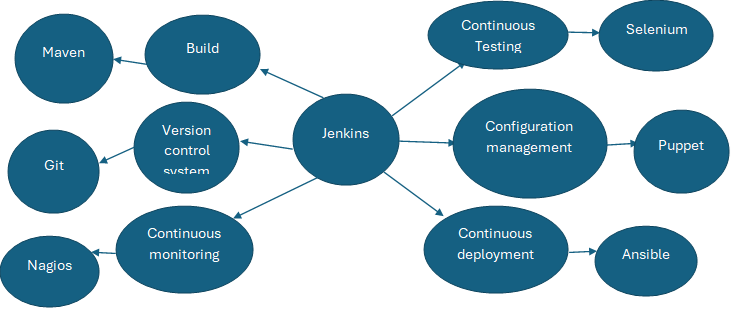
* **Git fetch:**
* Git fetch only downloads new data from a remote repository.

Git pull = git fetch +git merge

* **Git merge**:
* The git merge command is used to integrate changes from one branch into another
* **Git pull:**
* The git pull command is used to fetch changes from remote repository and merge into local Repository.
* **Meaning of cherry pick:**
* Cherry pick is a command that allows you to apply a specific commit from one branch to another.

Git cherrypick <commit id>

* **Git tags:**
* Git tags are used to mark specific points in a repository’s history, making it easier to reference and track changes.
* Tags are used to mark a commit stage as relevant. we can tag a commit for future reference.
* Tags are much like branches, and they do not change once initiated. We can have any number of tags on a branch or different branches.
* Primarily it is used to mark a projects initial point like V1.1.
* **To see all tags**
* Git tag -l
* **Developer is done with the project, and they want to release it. How can they release it.**
* Git tag -a “Release\_V1.0” -m “Tagged Project A Version1.0”
* **To check the version**
* Git tag Release\_V1.0 (this is done in our local machine we want to push into remote repository)
* Git push origin tag Release\_V1.0
* **How to delete the release**
* Git tag -d <version> (it deletes only in local machine)
* **How to delete in remote repository**
* Git push --delete origin <version> (it deletes the version in remotely)
* **What is rebase and what is the difference between rebase and merge.**
* Rebase is a git command that replays your commits on top of another branch.
* Merge is a git command that combines the changes from two branches by creating a new merge commit.
* The purpose of both the commands is same. The purpose is to merge one branch content into another branch.
* **What is Jenkins?**
* Jenkins a is a free and open-source automation tool.
* It helps automate the parts of software deployment related to building, testing, and deployment which supports continuous integration and continuous delivery.
* With Jenkins, organizations can accelerate the software development process through automation.
* Jenkins integrate development life cycle processes of all kinds, including build, document package, stage, deploy, static analysis and much more.
* Jenkins achieve CI with the help of plugins. Plugins allow the integration of various devops stages. If you want to integrate a particular tool, you need to install the plugins for that tool.
* For example: git, maven2 project, amazon ec2, HTML publisher etc.



* **Integrate jenkins with GitHub repository:**
* **How to get the code from github to jenkins**
* Username and password approach
* Ssh keys approach
* Personal access tocken approach
* Then it will build the project (using maven for java)
* **Prerequisites:**
* Git
* Java
* Apache maven
* Create one ec2 instance
* Execute Sudo su(to switch root user)
* Yum install git (installed git)
* Install java
* Install maven (sudo wget <url of maven>)
* Install jenkins(go to google- install jenkins-install on linux-select long term support release-copy all urls one by one).
* **Configure jenkins on ec2**
* Enter administrator password
* Click on installed jenkins plugins
* Create username, password, confirm password, full name, and email.
* Click on save and continue
* Click on save and finish
* Jenkins dashboard displayed
* **Configure GitHub credentials and maven location on jenkins:**
* Click on manage jenkins
* Under security click on manage credentials
* click on jenkins
* click on global credentials
* Click on add credentials
* Click on down arrow select username and password
* Enter username and password (GitHub username and password)
* Enter id (GitHub credentials)
* Description (added GitHub username and password)
* Click on create
* Go to dashboard (need to provide the location of maven)
* Click on manage jenkins
* Click on global tool configuration
* Click on add maven
* Give name(maven-3.2.5)
* Enter home directory of maven
* Click on apply and save.
* **I have to create job**
* Click on new item
* Enter job name
* Click on freestyle project
* Click on ok
* Give description (about project)
* Go to source code management
* Select git
* Go to github select repository and copy url and paste it in jenkins
* Click on down arrow add credentials
* Enter branch \*/main
* Click on add timestamps to the console output under build environments
* Click on build steps
* Select invoke top level maven targets
* Select maven version
* Enter package name(clean package)
* Click on apply and save.
* Clcik on build now.(success)
* **Add maven plugins on jenkins**
* Click on manage jenkins
* Click on plugins
* Click on available plugins
* Search maven integration plugin
* Click on install without restart
* Go to new item
* Enter job name(mavenproject)
* Select maven project
* Click on ok
* Give description (about project)
* Go to source code management
* Select git
* Go to github select repository and copy URL and paste it in jenkins
* Click on down arrow add credentials
* Enter branch \*/main
* Enter clean install
* Click on apply and save
* Click on build now(success).