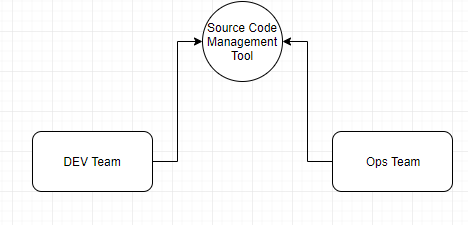
Bismilla Hirrahamaan hirrahim

**DEVOPS**

**Day 01: Date : 08th Oct 2018**

DevOps with cloud technologies.

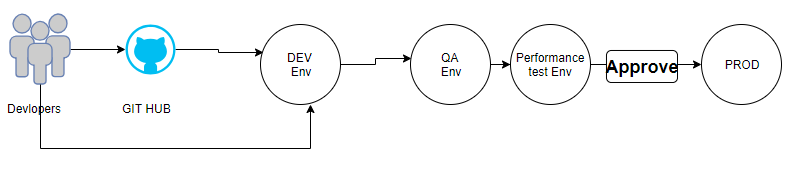
* Before DevOps, there no much communication b/w deva and ops team. Dev team used to commit the code in SCM(source code management) tool and ops team take code from tool and deploy.
* There was not short release of application before DevOps.
* 

1. Collaboration between all the team (Dev,Ops,QA).
2. Follow incremental development, this minimizes defects and improves productivity.
3. Automate the process, improves productivity.
4. Follow ***continues integration***, gives faster feedback about the code. It will also try to eliminate the issues. And developers can react to the issues faster and they can fix it with minimal efforts.
5. Implementation CD
   1. Continuous deployment
      1. No approval required to deploy pord.it deploys automatically
   2. Continuous Delivery.
      1. Before cod is deployed to prod, someone should approve.
6. Monitoring, Nagios (monitoring tools) for this Grafana(monitoring tools).
7. Dockerizing Apps, Kubernetes.
8. Configuration Management (ansible,chef,puppet,saltstack…etc).

Tools:

1. Git & Bitbucket
2. Maven for Build
3. Tomcat, a web server
4. Jenkins, automation server
5. Docker with kubernetes
6. Ansible for configuration management.
7. SonarQube (static code Analysis)
8. Nexus for Artifacts and Docker images
9. Nagios for monitoring
10. Linux, essentials and troubleshooting.
11. JIRA & Agile
12. Project

Flow:



**Day 02: Date : 09th Oct 2018**

DevOps is combination of tools.

1. **Git & Bitbucket :**

* GIT we call it as **source code manager.**
* It is also call it as **version control system**.

**Why we need SCM(source code management)?**

* It helps multiple developers to continue to same code base.
* It also help versioning the code. If something goes wrong with latest code version, you can roll back with previous working code.
* SCM tools also record all changes along with
  + Who can changed the files
  + Why it is changed
  + Timestamp, when it was changed
* Tracking defects is easy.
* You can secure your source using SCM tools, you can manage users and grant access to only authorized engineers.

**Why we should learn SCM as a DevOps**?

* We manage configuration files while working with

1. Ansible
2. Docker
3. Jenkins
4. Ec…

So we also must know how to use SCM tools.

Ex: YAML:

- Cat

- Dog

- Goldfish

-

- Python

- Lion

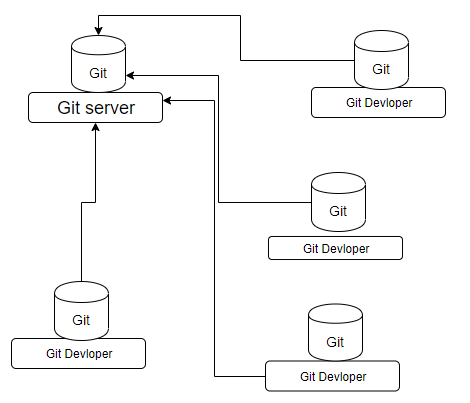
- Tiger

**Other SCM tools:**

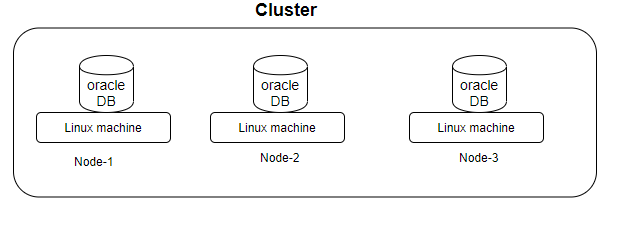
* **IBM Clearcase** : it is license product
* **Tortoise SVN** :open source
* **CVS** : it is similar to SVN :open source
* **Microsoft TFS** :open source
* **GIT**

**Why Git?**

1. Open source And free
2. Git is fast and distributed version control system.
3. Has more comprehensive features than any other tools.



**Distributed version control:**



* Along with Git remote server, every copy on developer’s local machine behaves like a server.
* It allows you to work locally.

**Central Version Control**

* In this case only one server behave like SCM server.

**Setting Up Git Server:**

1. Hosted Git. i.e download git and configure and maintain it on your own server.
2. Cloud based Git, some third party will maintain server and git, you are just using it over internet.
3. Few companies provide git as a service. So many providers are there, ex: bitbucket,Github

**Creating Cloud account for Git:**

* BitBucket
* GitHub (Microsoft purchased GitHub)
* Gitlab
* CodeCommit in AWS.

**Creating account in BitBucket:**

* Upto 5 users its free
* You can create unlimited private/public repositories.
* Goto bitbucket.org
* Create your account.

**Day 03: Date : 10th Oct 2018**

Continue to last class

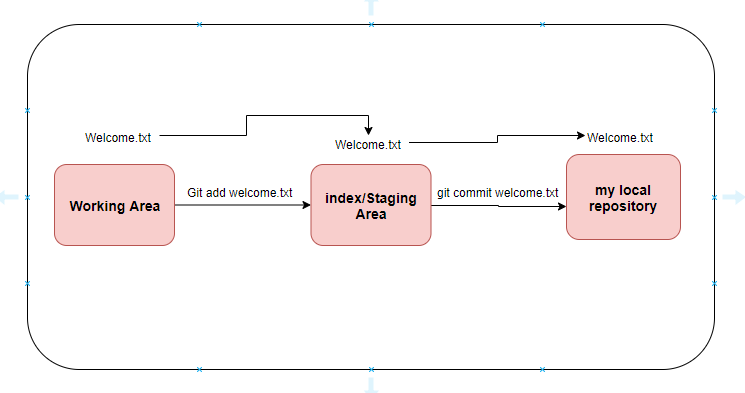
BitBucket :

Downloading git client. It comes with different flavors. CLI and GUI.

We need to more focus on CLI.

Installing Git Client:

* CLI (command line interface)
  + most of them use Gitbash(Git bash has both CLI and GUI).
* GUI (Graphical user interface)
  + Source tree
  + Git Bash
  + Smart Git
  + **Tortoise Git**
  + Etc…..
* Installing Git Bash.
  + Download and install.
* Creating repository in BitBucket.
* Git repository:
  + It represent a project.
  + For every project you have to create new repositories.
* Remote repository and Local Repository:
  + Remote : the repository that sits in remote server(ex: Bitbucket)
  + Local: the repository present on local machine.
* Learning how to use Git for managing Source Code:
  + Git Clone :
    - Git clone takes remote copy and save it on local repository. The copy present in local is called as local repository.
    - Git clone supports https / ssh protocols
    - SSH : is support only read access.it won’t give write access.
    - HTTPS: it gives read and write access.
    - Clone : is usually use it one time. Once you get local repository, next time onwards we will not use clone
* Doing modification to the local repository:
  + **Git status** : to check the my working file. It gives details about current working directory.
  + **Git add**: this add the file to index/staging Area.
  + **Add** : accepts wild card. Ex: git add file name , git add \* , git add \*.java , git add \*.yml
* Working area:
  + Git internally maintains this area and modification to the local repository are implicitly kept in “working area”.
* **Staging/Index Area :**
  + This area is used before committing your changes to local repository, your commit command commits on the files present in staging/index.
* GIT flow:



**Day 04: Date : 11th Oct 2018**

**Commit**: it picks only the files from the staging, ignore the files from working area.

If you have 03 files in in your working area and yo need to commit only two files. Please add only two files to staging area and commit.

**GIT config:**

* We need to configure username to access your repository.
* git config - -global user.name “ubedullasyed”
* - - : means all the repositories saved in my local repositories.
* git config - -global user.email [ubedulla.ss6@wipro.com](mailto:ubedulla.ss6@wipro.com)
* Note : above global config variable are stored under **# ~/.gitconfig**
* We can go to file location and make changes manually as well.

Commit :

**git commit –m ‘learn git’**

* **-m** is for message. And message will be type in **‘ ‘** .

Command flow:

Git add

Git commit

Git status

Git push origin master

Note: **origin** is path to the remote server. In the background this origin is mapped to url of remote location.

When we give clone command intial time it automatically mapped with origin(path).

* Origin entry is created when you clone.

**Git push** :

* Git push origin master
* Note: **origin** is path to the remote server. In the background this origin is mapped to url of remote location.
* When we give clone command initial time it automatically mapped with origin (path).

#**git remote -v** : it display the remote location files.

Git log :

It gives history of current branch.

**# git log “file name”** -> it gives history of this object.

# **git log - - oneline “file name”** -> it will show history in single line.

Note :: when we commit same name two file it use the “SHA” mechanism and give number. will have to check which algorithm sha is using?

**Find which files modified in a specific commit**:

* # **git log - - online**
* It will show how many files added.
* # **git show “commit id”** (which you got by “git log”)
* Help : **git show - - help** 🡪 it will help with git commands.
* Git show - - name - - pretty=”” **sha number**

**IMP\*\***

**Pulling changes from remote to local**:

* Whenever you work on new task. Always pull data from remote.
* **# git fetch origin master**
* It will fetch all remote commits and it keep in separate will not merge with local repositories.
* It will fetch all new commits from the remote location to local and it won’t merge with local branch.

**IMP\*\***

**Merging the changes in fetch with our local repository:**

* # git merge

IMP\*\*

# **git pull** :

it does #**git fetch** and # **git merge**

**Day 05: Date : 12th Oct 2018**

**Git branch**:

* Branch has collection of commits.
* Branch is light weight pointer. Which has sequence of commits.
* When we do commit on branch, branch pointer will take automatically moves to the latest commit.
* Light weight means: consume less memory and easy acess/fats.
* When we create another branch with same branch name. it will not keep another copy, just point to the master file ONLY.
* In git branch is lightweight, it will not create new setup of files. It just create pointer and points to the latest commit.
* Master is called default branch. It comes along with repository. It also called as main branch.

**When we use branch?**

* When you have to work in new task, bugfix, enhancement….etc. you have to do it in a branch.
* You should not directly push changes to ‘master’ and in real time, most of them will not have permissions to push to master branch.
* Master must always have well tested and bug free code.

**Creating a branch:**

* Branch always create from another branch.
* #git branch : is command to list all the branches in the local.
* #git branch –r : it will show all the branches in the remote.
* #git branch –a : it displays both local and remote.
* #git branch **task-1** 🡪 to create branch by name “task-1”
* #git checkout **task-1** 🡪 to switch the branch.

**Merging changes**:

* Option1:
  + Merging changes locally.
  + Switch to the branch where you want new changes.
  + First we need to switch to master #git checkout master.
  + #git merge task-1
  + It picks all latest commits from task-1 and merge with master branch.
  + **This option we don’t use in real time. This is not recommended option. We have better ways to do that.**
* **Merging via pull request**:
* **#git branch task-2**
* **You have checkout to be in task-2 branch while doing this.**
* Edit the file and do commits
* **#git push origin task-2**
* Then check and review the code and approve. Once approve developer can merge with master.
* We have to make changes in setting of bit bucket as once approve developer can merger the code after approval.
* We create pull request and merge the code.
* While working with git, we always delete source branch after merge.
* How to delete from local?
* #git branch –d task-2
* You cannot delete the branch being in the branch.
* #git pull
* Then delete.
* #git branch –d task-2

Force deleting a branch:

* #git branch –D task-2 (-d for normal delete –D for force delete)
* Branches gives you isolation.

**Day 06: Date : 15th Oct 2018**

Git pull request:

* It is a process. When you do pull request, it will show the difference b/w local and remote. It will not merge the code. After review and approve, developer will able to merge with master code.
* Once you got task
* Create a new branch.
* switch to branch
* edit the file in gui for testing
* git status
* git add \* ( we can use “.” also)
* git commit –m ‘implemented in new branch’
* git push origin task-4(task-4 is new branch name)
* then login to bitbucket
* open your repositpry
* got to barnchess
* select your branch
* got pull request
* create pull request.
* Select source(task-4) and destination (master)
* If you need you can choose reviewer
* Then create pull request.
* Reviewer will log in his account and goto pull request.
* Check and approve.
* Approve option we need to make settings in bit bucket settings.
* And merge the code with master. Then you can delete you task-4 branch.

IMP\*\*

**Managing git conflicts**:

* How do you manage conflicts?
* Conflicts happens, when two developers are working on same file and same line. We get conflicts.
* When we get conflicts first we need to pull the code from master to local and merge in local code and resolve the conflict’s and push back to master again.
* **How to resolve the conflicts?**
  + Manually: we can use manual approach to resolve the conflicts.
  + We also can use tools to resolve the conflicts. There are many tools for this ….Like “DiffMerge”
* Testing of merge:
* Delete your old branches if you have any
* Git branch bigfix (crate new branch)
* git checkout bugfix
* edit text file write something on particular line.(line-9)
* git add \*
* git commit –m ‘fixed bug’
* lets edit the same text file from the remote repository. GUI
* Lets write on same line(line-9)
* Commit
* Goto cli
* Git push origin bugfix
* Got pull request in GUI
* Source and destination to be select.
* Create pull request.
* We get conflict here
* Goto gitbash and pull the code from master.
* Git checkout master
* **Git pull origin master**
* **Git checkout bugfix**
* **Git merge master**
* We get conflict here
* We need to resolve.
* Open editor
* HEAD Is our changes.
* Master is remote changes.
* Do update manually changes in text file of your conflicts.
* Git add \*
* Git commit –m ‘resolved conflicts’
* Git push origin bigfix
* Got GUI and check in pull request warning will go off.
* Check on merge. You changes will update will update with master code.
* **Normally we delete branches after task completed. No one maintain branches separately.**
* We were having very small conflict last time

**Using a tool to resolving conflicts:**

* Google : **diffmerge**
* It is GUI.
* One is to merge and second can see the differences with this tool.
* Using this tool we can resolve conflicts and merge.
* We also can use this for gitdiff.
* Git also comes with some default merge tool.
* They are not very helpful.so developer integrate external merge tools with git.
* In our cse we are using diffmerge.
* #git mergetool - -tool-help (it will show the list of tools which we can use for merging)
* Download diffmerge
* Download and install

**We need to integrate diff merge with git :**

* From the same diffmerge page
* Goto documentation.
* Integration with third party software.
* Select “git”
* Select your OS
* Windows
* Open setting for windows.
* Choose “git bash shell”{github for windows or git bash shell}
* Copy the code in C:users/git cinfig file:
* [merge]
* tool = diffmerge
* [mergetool "diffmerge"]
* trustExitCode = true
* cmd = C:/Program\\ Files/SourceGear/Common/DiffMerge/sgdm.exe
* -merge -result=\"$MERGED\" \"$LOCAL\" \"$BASE\" \"$REMOTE\"

**Day 07: Date : 16th Oct 2018**

Git flow with commands:

* Created repository in bit bucket
* Cloned repository
* Git status
* Git add \*
* Git commit –m ‘message’
* Git push origin master
* Git fetch
* Git merge
* Git pull (git fetch & git merge)
* Git branch JAS-123456
* Jira is web tool to assign the task to developers.
* Git checkout JAS-123456
* Git push origin JAS-123456
* Git pull request
* Git branch –d JAS-123456
* Git branch –D JAS-123456
* Git log
* Git log ‘filename’
* While using diff merge for resolving conflicts, it might lean few unwanted files. We can configure difffmerge to automatically remove those unwanted files.
* Hwo to remove unwanted files:
* Go to git merge tool
* Got temporary files
* **keepbackup** to be set as false
* Got to .gitconfig file and paste under [merge tool]
* **keepbackup =false**
* first create branch do changes in file
* **move to master do changes in file.**
* Both the place commit the code
* checkout to branch and git merge master
* git mergetool
* after merge
* git status
* git commit –m ‘message’
* git push origin branch name
* trustExitcode = true (need tocheck)

**git Diff**:

* by default it will check the difference b/w local repository and working area
* it is for multi purpose command. It find difference b/w two parts.
* For testing:
* Maek changes in your file in working area.
* Run git diff
* It will show the difference bcz we have not commited to local reposiruty.
* Git difftool : it is built in tool
* Copy the below code in to .gitconfig file
* [diff]
* tool = diffmerge
* [difftool "diffmerge"]
* cmd = C:/Program\\ Files/SourceGear/Common/DiffMerge/sgdm.exe
* \"$LOCAL\" \"$REMOTE\"
* Git diff - -oneline ‘filename’
* Git diff tool “sha no current” “sha no old”

IMP \*\*

**Git HEAD** :

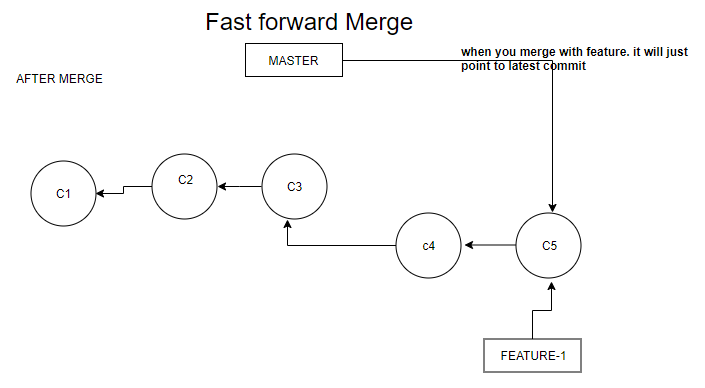
* It is pointer it offen points to latest commit in the current branch.

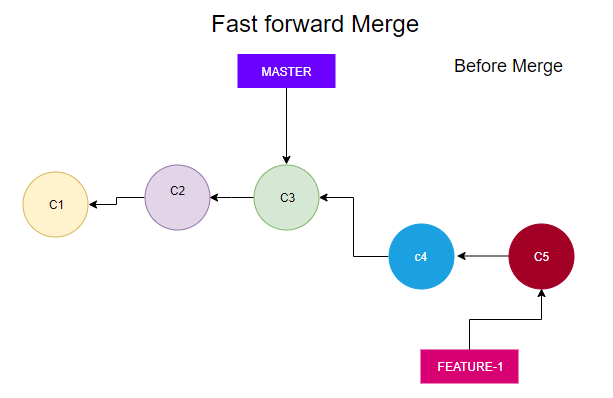
**Day 08: Date : 17th Oct 2018**

Git merging strategies:

1. Fast forward merge:

**Paste the diagram here**





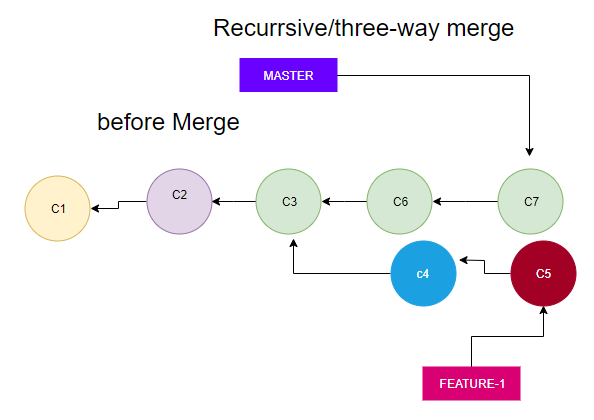
* 1. Feature branch is created from master based out of C3
  2. After feature branch is created there are no commits in master.
  3. If we merge now git will do fast forward merge.
  4. It will just moving the pointer to latest commit of feature branch.
  5. It simply moves the pointer to feature branch C5.

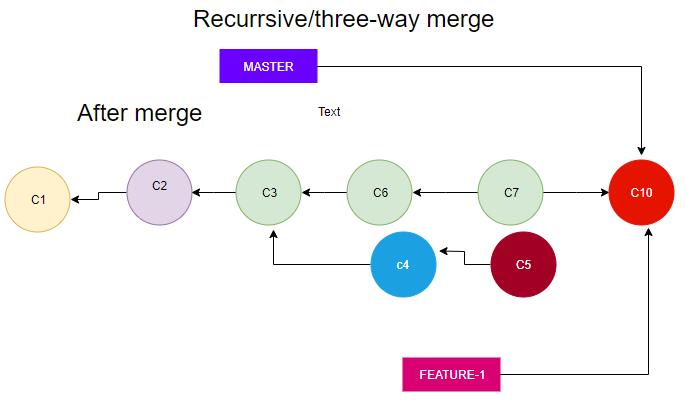
1. Recursive/three-way Merge:

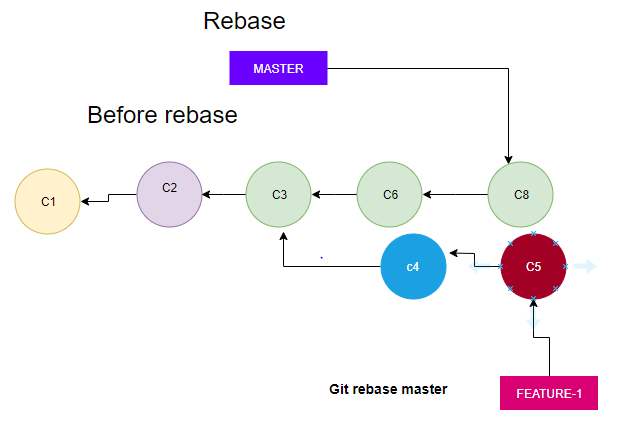
Paster diagram here.

* 1. After feature branch is created from C3, master branch progress with couple of commits.
  2. In this case if we merge, git will choose “recursive merge”

Diagram :







IMP \*\*

1. **GIT REBASE:**
   1. Feature branch is basedout of C3.
   2. Master is progress with two commits, after we creating feature branch.
   3. If we run **git rebase master** command from feature branch, our feature will be rebased to C8 in the master.
   4. While rebasing git will also merge few commits in master with feature.
   5. In the merge process, there might be conflicts, if there are conflicts, we have resolve. Using merge tool.
   6. Note: **maximum avoid rebasing because it tampers (modifies) git history.**
   7. We should have history as it is. Normally we should not make changes in scm history.

Demo for rebase:

* Create a new branch
* Git branch feature-1
* Git checkout feature-1
* Add some data in text file of feature-1 branch.
* Git add\*
* Git commit –m ‘’
* Git checkout master
* Make changes in text file, do not make conflicts add the data in last or top
* Git add \*
* Git commit –m ‘’
* Git checkout feature-1
* Git rebase master
* Git status
* Git checkout master
* Master file was not having changes of feature-1 now we need to merge with feature-1
* Git merge feature-1

**Git reverse Operations (reverting changes):**

* Demo
* Clean your working area
* Git push if you have any
* If we get error,
* Git pull
* Git mergetool
* Resolve the conflicts.
* Save and close
* Git status
* Git add \*
* Git commit –m ‘’
* Now
* Git push

**Now reverse :**

* Got master branch
* Make changes in text file
* Make lots of changes
* We want to undo this
* **Git checkout filename** -> it will discard all the changes in working area(Undoing changes in working area).
* Make some changes in text file\
* Git add \*
* Now we are in staging area
* Now I want to undo from **staging area.**
* Git reset HEAD filename -> it will undo changes from staging area.

**Day 09: Date : 18th Oct 2018**

Continue to last class.

**Removing a local commit**:

* We can do this with help of one of the two commands.
* **Git reset** commitid
* It removes the commit from local.
* If your commit is already push to the remote, we can’t use **reset**.
* You can’t remove specific commit, using reset. we can only remove recent end commits.
* In this case
* **Git revert** :
* It will not remove the changes, it will undo the changes only.
* Undoing also new commit kind of thing
* Note : you can’t remove merge commits.
* Git revert commitid -> it will undo all changes happens in this commit by making a new commit.
* Git revert is safe to use even if your commits are in remote.
* revert will not remove the commit, instead it will undo the changes in commit by making a new commit.
* You can do orbitry commits using revert. It means you can revert any of the commits. It will remove particular commit data and commit with new commit.
* Always revert is safe to use.

**Git soft reset**:

* **Git reset –mixed commitid**
* It removes the commit and changes are kept **in working area**.
* Git reset –soft commitid
* It removes the commit and keep changes to the commit in **staging area**.
* Git reset –hard commitid
* It removes the commit and permanently **discards everting** from your machine.
* DEMO
* Git checkout file name : remove from working area
* Git add \*
* Git commit –m ‘ ‘
* Git reset –soft commitid

IMP \*\*

**Git cherry-pick :**

* We can pick a specific a commit from different branch and merge with current branch.
* Make sure you have to be in right branch. Ex: master

**Git ignore files**:

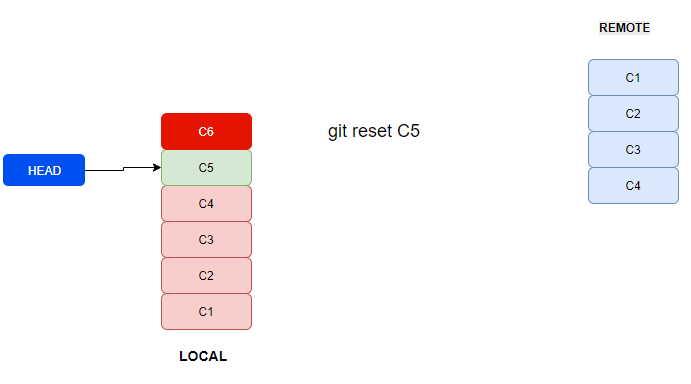
* Google :
* Using .gitignore we can ignore specific files and folders, when you stage your changes.
* Create a.class and b.class files
* #Touch A.class B.class
* #touch .gitignore
* Vi .gitignore

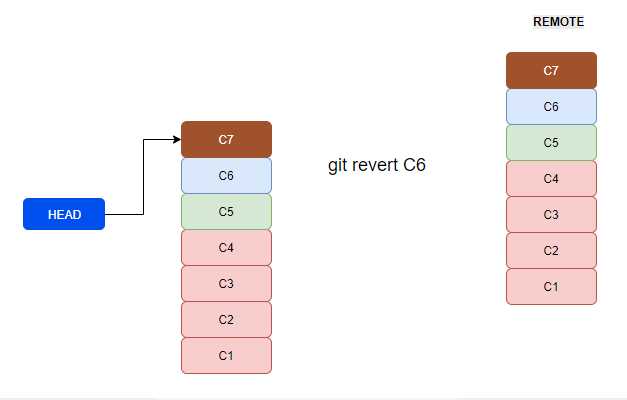
\*.class

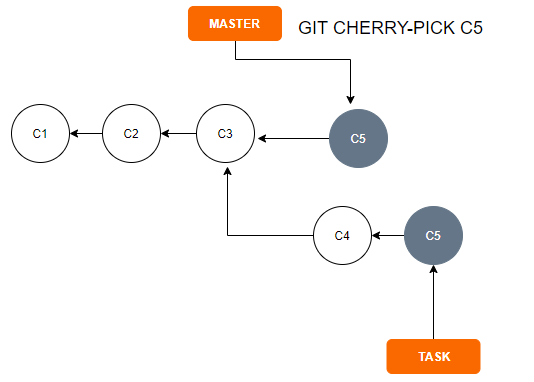
\*.sh

:wq

* Git status
* It will ignore .class files.







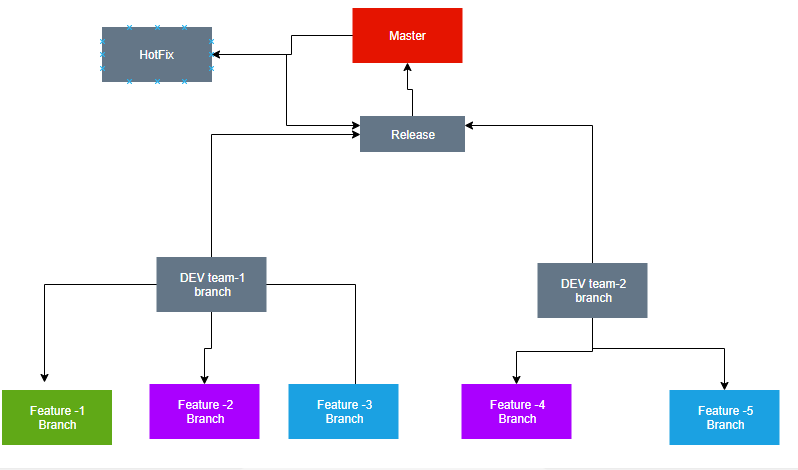
**Day 10: Date : 25th Oct 2018**

VV IMP\*\*

**Git branching strategy:**

* **Feature branch**:
  + Is the branch created by the developer to implement new feature and from which branch the branch is created.
  + Feature branch is created from development branch.
* **Development branch** :
  + Every team will have separate development branch. This branch is used to integrate all features implemented by this team.
  + Note: if we have multiple teams, it is good to have multiple development branches.
* **Release branch** :
  + Is used to integrate changes done by multiple teams.
  + It may not require if we have only one developer.
* **MASTER branch**:
  + This is the main branch for all. It should contain well tested bug fix code. The code is released to production from master.
* **Hotfix:**
  + This branch is directly created from master. It is used to fix production defects.

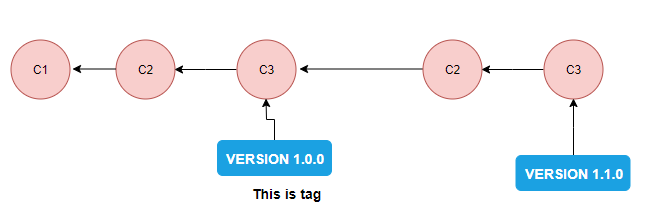
**Diagram for git branching stagey**:



**Git Tag** :

* Definition: tag is light weight pointer. Pointing to a specific commit.
* Tags are used for versioning the code and releasing to the customer.
* Head is always move to latest commit. But versioning will be light weight pointer. Just for our reference we versioning the code.
* We can do commits on branch, we can’t commit on tags.

Diagram :



**Diff b/w branch and Tag**:

**Branch**:

* It is used to implement new features or to fix the defects.
* We can do commits in a branch.

**Tag** :

* Tags are used to version the code and release it to the customer.
* We can’t commit changes to a tag.

Demo :

Got bitbucket

* You can create branches in bit bucket also
* Got branches
* Create branch
* Give branch name “feature-1”
* Type : you can select as bugfix,feature,hotfix…
* Goto git bash
* #git tag gst-1.0.0(versioning name)
* Git checkout commitid
* Git tag gst-0.0.1 (because I have tagged to previous commit)
* Git log - -oneline (it will show the version of your code)
* Git tag (will display all the tags)
* Git push origin gst-1.0.0 ( we are pushing tags to remote)
* We never delete tags.
* After pushing you can tags in gui
* If you want delete the tag
* Git tag –d gst-0.0.1

**Day 11: Date : 26th Oct 2018**

Git Init :

* This command which converts local folder into git repository.
* Create a folder and run git bash from there.
* Git status
* Git init
* Git status
* Git commit -m ‘message’
* Git push origin master
* Here you will get an error, bcz your repository is not authenticated.
* Now you have to create a repository in the bitbucket. We can’t create repository from local and push.
* Got to bitbucket,repository-> create new repository-> give repository name-> then create.
* Copy the url of Https……..
* Goto gitbash
* Git remote add origin <https://..........(enter> the url)
* Now you can push the data to remote
* Git push -u origin master
* Here -u is telling that “up stream”. Next time onwards we can push directly. No need to mention origin path.
* Git will update default up stream as origin master.

IMP\*\*

Git stash:

* It saves all the changes in working index into stash area and makes working tree clean.
* Ex:- if one developer is working on task, he has created a branch for this task and writing code, all of suddn his manager call and ask to fix one production issue.
* Now he has to create a branch and pull the data from master and work, But he is already having some file in his working area of his current task. In this scenario, he has to enter below commands and save his task data in stash place and continue to work on production issue.
* Create a branch for production issue
* Git status
* Git stash save
* Git stash list
* Git stash pop
* Git stash pop (to get the stash changes back. It takes recent entry in the stash and places I working tree and it remove the entry in the stash)
* Git stash list (it will show empty)
* Git status
* Your changes are back
* Git stash save –m ‘enter message here’ (need to check)
* Git stash apply (this will not delete the entry from stash list)
* Pop will delete entry and apply will not delete the entry from stash list.
* Git stash list
* Chose the old stash entry not recent one
* Git stash pop “give stash id” (here you can enter pop or apply)

=========**GIT closed**================

**Build tool – (Maven)**

* We can compile source code.
* Manage dependencies.
* Creates directory structure.
* Copying appropriate files into appropriate directories.
* Create software package.
* **What is maven**?

It is popularly known as build and dependency management tool, however maven can do more than build and dependency management.

Note : Most of the developers say maven is extension to “Ant”.

* **What kind if application we can build using Maven?**

Maven is mostly used for building java based applications.

Note : If your code is .**net** use **MSP build tool**.

* **What other build tolls**?

MSBuild for .Net

**Ant** for java

**Gradle** for java,android etc…..(we use for mobile application also)

C/C++ will use **Make** build tool

**Day 12: Date : 29th Oct 2018**

**Installing and configuration Maven on Linux:**

* + We need Linux machine to practice.
    - Install virtualization tools like(VMware,virtualbox,etc)
    - Else create an account in cloud(AWS,azure,gcp,Digitalocean) and utilize.

**Launching linux machine in AWS**:

* Launch ec2
* Select the OS image is “**Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type**”
* Tag as Maven server.

Connect to remote linux machine which is in AWS:

* Use git bash/putty to connect to remote servers.
* ssh - i keyname.pem ec2-user@publicip
* “ **- i** “is stands for identity of your password.

**Installing maven on EC2**:

* Prerequisite for maven is JDK7.
* Google : maven download-> download-> check the system requirements(it is for windos).
* EC2 by default comes with JRE 7.
* JDK – java development Kit.
* JRE- Java Runtime Environment.
* **Let’s install JDK:**
  + Yum – it is **package manager** to install/uninstall/upgrade the softwares in Linux.
  + #sudo yum list | grep java-1.7 (check the package name from yum repo)
  + Java –version
  + Javac –version (it is for JDK)
  + Choose and Install the package of JDK is : “ java-1.7.0-openjdk-devel.x86\_64”
  + #**sudo yum install java1.7.0-openjdk-devel**
* **Let’s Install Maven**:
  + Cd /opt
  + Google for maven download. Click on download link.
  + Choose binary tar.gz archive.
  + #wget http://mirrors.estointernet.in/apache/maven/maven-3/3.5.4/binaries/apache-maven-3.5.4-bin.tar.gz (right and copy the address)
  + #Sudo tar –xvf apache-maven-3.5.4-bin.tar.gz
  + #sudo mv apache-maven-3.5.4-bin.tar.gz maven3
  + #mvn package
  + Error : command not found
  + We have to go to path
  + #/opt/maven3/bin/mvn package (now it will work).
  + To avoid the above option
  + Set the path for maven
    - To add environment variable at user level use ~/.bashrc
    - To add environment variable at system level use ?????
  + #vi ~/.bashrc

M2\_HOME=/opt/maven3

PATH=$M2\_HOME/bin:$PATH

Save and exit

* + #mvn –version
  + Error: we need to logout and login to load the updated files.
  + Use force command 9if you do not want to logout)
  + #source ~/.bashrc
  + #mvn –version
  + It will the details now.

**Day 13: Date : 20th Oct 2018**

Installing git in Linux(EC2):

* #sudo yum git install –y
* #git clone “give your repository url”
* #ls
* You have to be in your project folder while building your project
* #**mvn package**
* It will download all the dependencies and generate the .war
* Build ready.
* Git branch
* Checkout to the branch and build it(mvn package)
* Dev/SIT/PORD

**Maven Dependencies**:

* Java web application:
  + Take data from database and generate excel and sent this sheet in a mail.
* An external jar(java archive) file which is used by your project.
  + Jar(java archive)
* While using maven developers must tell which dependencies they want, and maven will automatically download and adds to the project.
* POI is dependency of managing excel with java.
* Maven automatically downloads transitive dependencies as well.

IMP \*\*

**From where your maven downloads your dependencies?**

* Maven maintains repositories.
* Maven downloads dependencies from one of its repository.(IMP \*\*)
* Maven deals with following repositories:
  + 1. **Central repository**: this repository maintained by maven over internet.
    2. **Local repository** : This repository maintained by maven on our local machine i.e the machine on which you run maven commands
    3. **Remote Repository:** This is similar to central repository, But it is maintained by companies within their office network.
    4. Use cases of maintaining remote repository?
       1. Security
       2. Organization may have it’s own dependencies used by various teams then we have to maintain “remote repository” Ex: Secutiry.jar (company specific dependency)

What is the path of maven local repository?

* cd /home/ec2-user/.m2/repository/
* cd ~/.m2repository

Maven configuration file:

* **POM.xml**:
  + It is maven configuration file.
  + Pom -(Project Object Model)
  + maven function based on this configuration file(pom.xml)
  + you have to place this in project root directory
  + When you run maven commands maven looks for pom.xml in its current directory.
  + Pom.xml is created by developers while setting up project.
  + As a devops/build release we also update this file at times.
  + This file has all the details of the project and dependencies.
  + While using maven in projects, projects must follow specific directory structure, which is taken care by developers.
* Details of pom.xml:
  + **Group id** 
    - it gives information about client for whom you are developing this project.
    - Its value can be anything, but as pet the maven convention it should reflect clients name.
    - Valid groupid names
      * Javahome
      * In.cbec
      * In.javahome (can be reverse of your domain name)
  + **Artifact id** :
    - This is project name.
  + Packaging :
    - When you build this project, what is your packaging format(war/jar/ear)
  + Version:
    - Current version of this project.
  + Dependencies:
    - It contains dependencies used by this project
  + Plugins:
    - Via plugins we can add new features to maven.
* **Day 14: Date : 31sh Oct 2018**

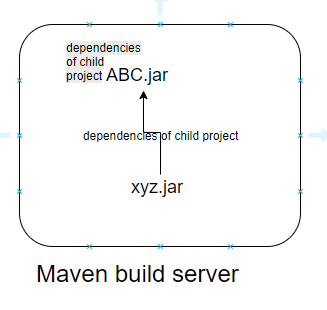
Generating maven based java project using mavan command:

* Go to maven docs from google
* Creating a project
* Please enter the below command to create new maven project.
* mvn archetype:generate -DgroupId=in.javahome -DartifactId=sms -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
* if we give “quick start” it will generate -> .jar
* if1 we give “webapp” It will generate -> .war
* after creating the project
* please enter into the project folder and mvn package
* it will generate the .war file.

IMP \*\*

**Maven build life cycle**: below is the backend process when you run the “mvn package” command.

* (1)**Validate**:
  + When you run any maven command first maven **validates** your project (pom.xml).
  + #mvn validate (it just validate)
* (2)**Compile** :
  + It first download dependencies then it start compiling code.
  + #mvn compile (it compile)
* (3)**Test cases**:
  + Junit testing
* (4) **Package**:
  + It will create asper your pom.xml(war/jar/ear)
* (5) **verify**:
  + This phase has to be explicitly integrated.
* (6) **install**:
  + Install your package in local repository.
  + #mvn install
  + Copies your package to local repository.



* (7) **Deploy** :
  + It is just uploads packages to remote repository.
  + #mvn deploy
  + Error: we have not configured remote repository. We need to install the remote repository here.

Maven Goas:

* mvn package: here “package” is goas
* mvn install : here “install: is a goas.
* We can club multiple goes like #mvn compile test (compile and test are goes)

**mvn clean**:

* It will clean the output generated by mvn package.
* This deletes output folder generated by maven(target folder).
* We can use command like # **mvn clean package**

**Maven released version and snapshot version**:

* SNAPSHOT version: is a version which is currently under development.
* When our project Is depends on snapshot version, every time we build our project maven downloads latest copy.
* RELEASE version:
  + This version which are completed with development, there will not be new changes to same version, if there is a defect in this release version, it is fixed in new version.
* If the version names ends with SNAPSHOT then it is snapshot version.
* We have make modification of snapshot and release in pom.xml file based on project state(under deployment /completed)

**How to skip test cases in Maven**:

* Some time we need to skip the test cases, when we have issue with test cases.
* #**mvn package –DskipTests=true**
* When we run the above command it will skip the test cases and create .war file.

**Running maven in debug mode:**

* # mvn package –X
* Above command gives more output, helps to debug the isssues.
* To figure out the project belongs to what?
* Check the pom.xml file.

**Difference between ANT and Maven**:

ANT :

* It Is just build tool. It won’t manage any dependencies.
* Each and every task(compile,test,package,class path..etc) you have to describe in build.xml file.

Maven :

* It is build, dependency management and more.
* Maven follows “***Convention Over Configuration***” lots of tasks work automatically without explicit configuration in pom.xml.

**Day 15: Date : 01st Nov 2018**

Google: maven modules

To know about multi projects(child and parent like gst).

**Web server (Tomcat):**

1. Web server is used to deploy web application.
2. What is web application
   1. Application accessed using http(s) using web browser or mobile apps.
3. Static and Dynamic web applications.
   1. **Static** web application is developed using html,css,javascript without using server side technologies like(java,python,.Net,PHP,NodeJS…etc).
   2. Dynamic web application are build using (java,python,.Net,PHP,NodeJS…etc).

**What web servers provide**?

1. http(s) implementation
2. Multithreading support / multitasking support
3. Security, to secure your web application…etc.

**Different web servers available in the market**:

1. Apache web server -> use cases :deploy application, Load Balancer,proxy
2. Apache Tomcat.
3. Oracle WebLogic
4. IBM WebSphere.
5. Jboss
6. Sun Glassfish
7. Pramati App server
8. Nginx -> use cases :deploy application, Load Balancer, proxy
9. Windows IIS (only for .Net base application(Internet Information Service))
10. ….etc

Difference between Webserver and App server?

* **Web servers** have basic feature like
  + http(s) implementation
  + Multithreading
  + Basic security
* **App server** have all the features of web server plus
  + Messaging support
  + Name registry support
  + transaction Management support
  + advanced security
  + EJB(Enterprise Java Beans)

**Apache tomcat**:

* Install and configure tomcat on linux
* Learn how to deploy java web application
* Troubleshooting deployment issues.

Live :

* Launch one EC2 for tomcat
* Choose first AMI while lauching
* Tag : Tomcat
* Launch the ec2
* Connect to EC2
* Google : download tomcat
* Go for first link
* Choose Tomcat 8 from the tomcat home page.
* Make sure choose under Binary Distributions->
* Core: (any one below)
* zip (pgp, sha512)
* tar.gz (pgp, sha512)
* here we choose **zip**
* zip can used in windows and non windows as well.
* Right click on zip and copy the link address.
* <http://mirrors.estointernet.in/apache/tomcat/tomcat-8/v8.5.34/bin/apache-tomcat-8.5.34.zip>
* Goto EC2
* #Cd /opt
* #sudo wget <http://mirrors.estointernet.in/apache/tomcat/tomcat-8/v8.5.34/bin/apache-tomcat-8.5.34.zip>
* now unzip the file
* #sudo unzip apache-tomcat-8.5.34.zip
* After unzip
* Lets rename
* #sudo ln –s apache-tomcat-8.5.34.zip tomcat8
* Above I have created soft link by name of tomcat8.
* Need to write Story about link |??????????
* #sudo chown –R ec2-user:ec2-user tomcat8
* Have change the ownership to our user and group.
* Apply sam ownership for link also
* #sudo chown –R ec2-user:tomcat8
* Now lets start Tomcat
* Tomcat has prerequisites, it will not work **without java**.
* In EC2 java will be inbuild
* #cd /opt/tomcat8/bin
* Location of tomcat : /opt/tomcat8/bin
* #ls –l
* #chmod +x \*.sh
* You should run the above command from bin directory.
* You should install JDK in the tomcat server.
* #./startup.sh (tomcat start command)
* Tomcat by default runs on port 8080
* Goto browser enter <http://ip:8080>

**How to deploy web application**:

Deployment directory for tomcat is : TOMCAT\_HOME/webapps

In our case : /opt/tomcat8/webapps

Lets login to **maven server**:

* Login EC2 maven server
* Maven and tomcat will be in different machines.
* Connect to maven server
* Lets build from starch
* #git clone <https://github.com/javahonetech/my-aap>
* #cd my-app
* #mvn package
* Your war file will be in target folder
* Now move war file from maven to tomcat server
* Open the .pem file key
* In Ec2 come out of the project
* #vi key.pem
* Paste and :wq
* #chmod –R 600 key.pem
* #ssh - I /key.pem ec2-user@ip
* Now
* scp (secure copy)
* #scp - i /key.pem my-app/target/myweb-0.0.04.war ec-user@ipoftomcatserver:/opt/tomcat8/webapps
* To check
* Goto broser enter tomcatip:8080/myweb.0.0.4

**Day 16: Date : 02st Nov 2018**

Continue to last classs:

* Before we scp the war file. We need to rename the war file.
* Myweb-0.0.04 to myweb.war
* Then move the war file from maven ec2 to tomcat ec2
* #ssh –I ~/.key.aws target/myweb.war ec2-user@ip.........
* IMP\*\*
* Running commands on remote server.
* #ssh – i ~./awskey.pem ec2-user@ip /opt/tomcat8/bin/shutdown.sh
* #ssh – i ~./awskey.pem ec2-user@ip /opt/tomcat8/bin/startup.sh
* Now got o browser
* Enter <http://ec2publiciptomcat:8080/myweb>

**Another deployment with ver0.4**:

* First goto **maven ec2**
* #Git pull
* #Mvn clean package
* #Build will be ready
* #ssh –I ~./aws.pem ec2-user@ip rm –rf /opt/tomcat8/webapps/myweb\*
* Above command removes old war file details from tomcat server
* Now rename
* # rm target/myweb-0.0.04.war target/myweb.war
* #scp –i ~./aws.pem target/myweb.war [ec2-user@ip:/opt/tomcat8/webapps/myweb.war](mailto:ec2-user@ip:/opt/tomcat8/webapps/myweb.war)
* #ssh – i ~./awskey.pem ec2-user@ip /opt/tomcat8/bin/shutdown.sh
* #ssh – i ~./awskey.pem ec2-user@ip /opt/tomcat8/bin/startup.sh
* Now goto browser
* Enter <http://ec2publiciptomcat:8080/myweb>

**Troubleshooting deployments**:

* Lets try with one failed deployment.
* Make something worn on code
* #git pull
* #mvn clean package
* It will be failed
* You need to share the logs with development team they will fix the issue and confirm.

If deployment fails, what to check and where to check:

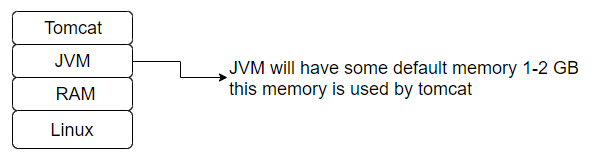
* #ssh to tomcat ec2
* #cd /opt/tomcat8
* #cd logs
* # ls –l
* Catalina.out is latest log file, which not mentioned time slot(bcz it is latest one)
* #tail -100 catalina.out
* It will show the logs with error details.
* Note : web.xml used by tomcat for deployment.

How to change the port no of tomcat:

* Goto tomcat ec2
* # cd /opt/tomcat8
* #cd conf
* #ls –l
* #vi server.xml
* Update the port no here.
* Goto connector…. Port=’8080’ make changes here.
* Then :wq
* Then restart the services. It will be effective.
* #../bin/shutdown.sh
* #../bin/startup.sh
* Now try to access the app by new port no

Have you ever seen out of memory issue?

* Soluton1: increase the JVM (heap) memory.



H.M

How to increase JVM heap size?

* Heap means (the location where your objects are stored.)

Goto thought he below link to set the JVM memory:

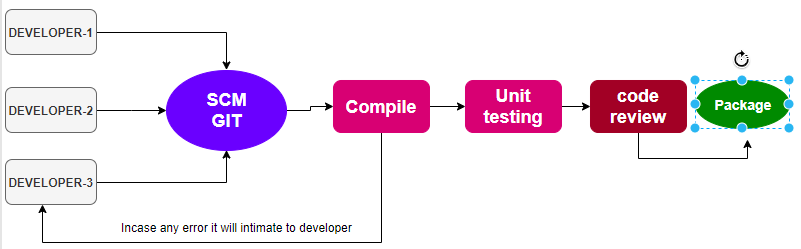
<https://crunchify.com/how-to-change-jvm-heap-setting-xms-xmx-of-tomcat/>

**Day 17: Date : 05st Nov 2018**

**Jenkins**

* It is very important,
* It is open source and popular automation
* It is started with CI tool and it is now with CD
* Jenkins is popular leading open source automation tool
* But today Jenkins and automate anything.
* Jenkins is also used for CD(continuous deployment/continuous delivery)
* Jenkins is all about plug and play, if you wanna add new features to Jenkins, it’s done by installing a plugins.
* We can automate anything and everything without writing our own scripts.
* We can easily integrate Jenkins with tools like, chef,ansible,docker,artifactory,nuxux,sonarQube,sonar,tomcat…..etc.
* Note : Jenkins supports two types of jobs:
  + Pipeline Jobs
  + Freestyle jobs
* Pipeline jobs :
  + This feature is available from 2.X onwards
  + It is highly important.

**CI – Continuous integration:**



* It is an automated process, which is executed several times a day, whenever there is a new in the remote repository.
* This CI process verifies the code for error, if any error found it triggers automated feedback(email,slck notification) to development team.
* Benefits of CI:

1. It is automated process
2. It delivers immediate feedback
3. Immediate feedback improves productivity
4. It improves code quality
5. It gives customer satisfaction
6. You can aggressively invent new features for customers.

CD – Continuous Deployment:

* This is continuous to CI
* This is automated process and deploys the code
* This picks the package(war,jar,ear) created by CI and automates the deployment from dev,test,till production.
* Devops is all about frequent releases.

CD – Continues Delivery:

* Everything I same as continuous deployment, there will be an approval process before releasing code to production.
* Many companies follow the continuous delivery. Here approval is required
* Production is the environment where customer uses the software.
* When we follow incremental development bugs will free from production.
* If there is no approval button means that will be continuous deployment.

Dev environment:

This used by developer for integration test.

QA environment:

This is used by testing team to test(using selenium/QTP)

Performance:

Used by performance testing engineers to check performance.

UAT : (User acceptance test)

Used by product owner to test.

Prod :

This is live servers, where customers use your software.

**Installing Jenkins**:

* Prerequisites
  + Java must be installed to run Jenkins, because Jenkins is written in java.
* Git is written in C, C++
* Maven ,Jenkins,ansible ..are written in java.
* Installing Jenkins in Linux
* Goto Jenkins.io
* Click on download
* Click on redhat/fedaro/CentOS
* Follow the steps.
* Launch one EC2 for Jenkins from AWS
* Tag: Jenkins
* Connect to ec2 Jenkins
* By default jave comes with ec2, but we need latest version for jenkins. Lets intall java(jdk)
* #sudo yum list | grep java-1.8
* Choose java-1.8.0-openjdk-devel
* #sudo yum install java-1.8.0-open-jdk-devel
* It will install the jdk
* Now check the version
* #java –version
* Still it is pointing to old version.
* Make changes to point ot latest version
* #sudo update –alternatives - - config java
* It will show the options
* Choose the latest one :2
* Then check
* #java –version.
* Same like,if wanna set for javac
* #sudo update –alternatives - - config javac

Install config of Jenkins :

1. Launch EC2
2. Install jdk package
3. Install Jenkins

**Day 18: Date: 06th Nov 2018**

Continue to last class.

* We have installed java dependencies
* Let’s install Jenkins now.
* We have many ways to install Jenkins
  + Through Docker
  + Download file and put it on webserver
  + Installation on Linux by yum
* Go to jenkins.io
* Click on download
* click redhat/Ferodara/CentOS
* enter the below commands
* #sudo wget -O /etc/yum.repos.d/jenkins.repo <https://pkg.jenkins.io/redhat-stable/jenkins.repo>
* #sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
* #yum install Jenkins
* Now Jenkins installation is done.
* #sudo chkconfig jenkins on
* It auto start jenkin services when your vm reboots.
* #sudo service jenkins start
* Jenkins services are started
* Jenkins by default runs on port no 8080

Configuring jenkins:

* Access Jenkins by browser
* EC2 publicip:8080
* It will ask for password. Copy the password from jenkins EC2 and enter.
* This password works for first time. It is temporary password.
* Click on “install suggested plugins”
* Create first admin user.
* It is mandatory
* After filling your data click “save and continue”.

Creating first Jenkins job:

Requiremnets:

* Clone or pull code from git
* Build this code using maven

Create your first job:

* Click on create new job
* Give your job name ‘ ‘
* Select “free style project “
* Click on source code
* Select git
* Repository URL (https:/github.com/javahome/my-aap)
* It will fail now,bcz git is not installed on jenkin server.
* Goto EC2 of jenkins
* #sudo install git –y
* Git installed in jenkin EC2
* Adding git crendentials:
  + Click add button
  + Kind – username with
  + Give all the details
  + Id : github
* We have stored git credentials now.
* Choose the branch where your code is available.
* Click on build tab
* Choose in option -> “invoke top level maven target”.
* Goals : clean package
* Save and apply
* Click on build button
* It will fail now as we have not installed Maven in jenkin server.

Installing maven on Jenkins server:

* We are installing maven from jenkins GUI access, (else we can install manually in jenkins linux box also).
* Goto Jenkins home page
* Select manage Jenkins
* Select “global tool configuration”.
* Maven-> add maven
* Choose version -> 3.6.0 -> enable install automatically
* Click on save. It will install automatically
* Come back to job configuration
* Under build
  + Maven version -> maven-3
* Save and apply
* Now click on build button
* BUILD SUCCESS
* Start your tomcat EC2

Requirement to do automation:

1. Connect to tomcat
2. Remove old war file
3. Copy new war file
4. Restart tomcat

* Goto configuration -> post build action
* We have multiple option build automation, But we will using here through plugins.
* Google : publish over ssh plugin
* Install publish over ssh plugin :
  + Goto jenkins home page
  + Manage Jenkins
  + Manage plugins
  + Select available
  + Search publish over ssh
  + Choose it
  + Click Install without restart
* Configure ssh plugin
  + Manage Jenkins
  + configure system
  + publish over ssh
  + key -> copy and paste aws Ubedulla-AWS-virginia.pem key
* ssh servers : click on add
  + name : Tomcat
  + hostname: private ip address
  + username : ec2-user
  + remote directory : /opt/tomcat8/webapps
  + where we have installed the tomcat
  + click on test configuration
  + save
* open the job configuration
* under build -> add build step
* “send files or excute command over SSH”
* Name : tomcat
* Transfer
* Source file -> target/myweb(\*).war
* Above no need mention \*, if we have written script for rename the war file.
* Execute command :

/opt/tomcat8/bin/shutdown.sh

rm –rf /opt/tomcat8/webapps/myweb.\* (have some doubt)

/opt/tomcat8/bin/startup.sh

* Remove prefix : target
* Save
* Click on build button now
* Now access the the application through browser.

Rename of your web file :

* Configure
* Execute shell script
* Command
  + mv target/web\*.war target/myweb.war