**DEVOPS**

**Waterfall model**

   - Requirements are clear and fixed

   - Product definition is stable

**Agile model**

   - Jira tool

   - Requirement changes frequently

   - Development needs to be fast

**DevOps approach**

   - Development + Operation

   - In waterfall model, the client gives requirement and after the entire project is completed, we show demo to client

   - In Agile model, entire application is divided into small modules and works for 10 days (1 sprint) both developers and testers will be working and finally shows demo to product owner. It reduces gap between customer and developer

   - DevOps reduces the gap between developer and operation team, so both developer and operation team will work together

   - Requirement changes frequently

   - Development needs to be agile

   - Operations needs to be agile

**DevOps tools**

1. Plan - Jira

2. Code - After the coding is completed, we put in GIT or subversion

3. Build - called packaging, used to build the application and put all dependency and creates jar or war files - Maven, Ant, Gradle

4. Test - JUnit, Mockito, Testng, cucumber, selenium

Plan, Code, Build, Test - done by development team

Deploy, Operate, Monitor - done by Operation team

1. Deploy and Operate - Puppet, Chef, SaltStack, Ansible

2. Monitor - Splunk, Nagios, Sensu

1. Integration - integrate development and operation - JENKINS

**DevOps stage**

1. Version control - which maintains different version of the code - GIT

      - how to maintain the source code

      - If developerA commits the code and that commit has some errors, so we want to know which developer made a commit and at what time and what changes that cause an error so that we can revert back to that commit and do necessary changes

2. Continuous Integration - compile, validate, code review, unit testing, integration testing - 66% of organization using Jenkins, Teamcity, Bamboo

   - CI will pull the latest code committed inside GIT, after that it builds the application, build is not only compiling the code, but also validating the source code (i.e.) checks the code quality whether packages, variables, methods declared are user or not, review the code, performs unit testing (method by method) and integration testing (layer by layer)

3. Continuous delivery - deploying the application in test server and performing UAT (User Acceptance Testing)

      - Until now your product is ready, it is tested in preproduction stage

      - Depending on the project, if we want to manually get approval from client it goes to continuous delivery stage, if the approval is automatic means it goes to continuous deployment stage

4. Continuous deployment - deployment the application in production server for release

5. Continuous Monitoring - once the product is delivered it is continuously monitored using splunk or sensu etc., and provide the feedback and based on that feedback the cycle will continuously working

**Version Control System**

   - It is management system that manages the changes that u made in your project till the end

   - Whenever u changes, it creates a snapshot of entire project and save it, these snapshot are called different version which represent entire state of application at particular time

Why Version Control?

1. Collaboration

    - version control provides with a shared workspace and it continuously tells who have made what kind of change and what was changed

2. Storing version

    - In version control system we don’t want to worry about how much code we save and how u name them because version control system acknowledges that there is only one project or one version in disc and all the changes that u made in past is neatly packed inside GIT and if u want to go back to previous version we can revert back all changes

3. Backup

    - The developer will fetch all the project files from the central repository (GIT repo) and store in your local machine (i.e.) local copy

    - Now the developer has to do changes in local copy after that they will commit the latest code to central repo

    - If central repo crashes also they have backup in local repo and vice versa

Version control tools

1. GIT - both central and local repo

2. Subversion – doesn’t have local repo

3. CVS (Concurrent Version System) – doesn’t have local repo

4. Mercurial

GitHub is a central repo and git is local repo we can perform to push data from local to central and pull data from central to local

What is GIT?

   GIT is a distributed version control tool to create local repo in your local machine and it will help u to access your remote repo

What is repository?

    - It is a directory or storage space where all your projects can live in your local machine

Git always refers the latest version which is called pooling repository, head pointer always points to latest commit version and reset command will change from one version to another version

Steps

1. Create GitHub account

2. Create new repository

3. create demo folder

git init - used to create local repo

4. To link the repo for push and pull purpose, for that we need to add origin using

      > git remote add origin "reponame"

5. Inside demo folder we create file1.txt

6. git status - to check the status of file in staging area or not

7. git add file1.txt

8. git commit -m "First file commit" - move file from staging area to local repo

9. git push -u origin master - move from local to central repo

Branching

Branches are pointers to a specific commit. Let’s say u r working on the master branch and u have just made a change and u decided to add some new features onto it.

1. >git branch firstbranch  - to create new branch

2. > git branch - to list all branch

3. >git checkout firstbranch - used to move to firstbranch

Instead of creating new branch and checkout to that branch separately we can create in single command using -b

    >git checkout -b test\_branch

To rename the old branch to new branch name

     > git branch -m test\_branch testing\_branch

    Now check whether the name is renamed by

      > git branch

17. To delete the branch

     > git branch -D testing\_branch

 We should not be in the current branch that we are deleting, whether it is merged or not merged it delete

18. To delete the branch

     > git branch -d testing\_branch

 It will prompt the message, before merge the data to another branch then it will delete

To push the information to a branch

>git push --set-upstream origin secondbranch

Merging means combining the work of different branches altogether.

1. Now we want to combine demo4 from firstbranch to master branch. One important thing is we want to move from firstbranch to master, so master is destination and we should be in master branch

   >git merge firstbranch

Reverting

   If we want to revert to previous version

1. Create demo7.txt inside demo folder with some content

2. Add to index using

     >git add demo7.txt

3. Commit the changes

   >git commit -m "revert file"

4. Now modify some content in demo7.txt

5. Commit this file again

   >git commit -a -m "revert commit"

6. Now if we want to revert back to previous version

   >git log

--display all commit log

7. Copy the first 8 digit of commit log of version1

   >git checkout "8 digit commit" demo7.txt

8. >cat demo7.txt

   -- it will contain the old info

To pull the data

create a folder

git init

git pull "git url"

Git Merging and Rebasing

In Git there are 2 ways to integrate changes from one branch to another. To push the data from child to master branch

For example, u have 2 branches in project and u want to merge the changes we have 2 ways

1. git merge 2. git rebase

git merge

1. Is a non destructive operation

2. Existing branches are not changed in any way

3. Creates a new merge commit in the feature branch

git rebase

1. Moves the entire branch to begin on the tip of the master branch

2. Rewrites the project history

3. we get much cleaner and linear project history

Continous Integration - Jenkins

Whenever we commit code in GIT, jenkins will automicatically fetch the latest code - build the appl - compile, validate, code review, unit testing, integration testing

- The developers required to make frequent commit to the source code repo, because of that any changes made in the source code will be pulled the latest version called as pooling repository by the CI server and that code will vbe build, then it will be deploy on test server for testing and once testing is done it also deployed in production for release and developers get continuous feedback on the same

CI tools

1. Jenkins written in Java - default contain Tomcat server - Everything is in the form of plugin

2. Teamcity

3. Bamboo

4. BuildBot

4. Travis CI

Software - jenkins.war

>java -jar jenkins.war --httpPort=2000

whenever we do any jobs in Jenkins dashboard by default all programs will be stored in users/name/.jenkins

Like maven how it creates .m2 folder, jenkins also creates .jenkins folder which contains what job we created, plugin installed everything will present.

Whenever we do any work in jenkins dashboard, by default all programs are stored in c:/users/name/.jenkins folder because JENKINS\_HOME is mapped to this particular folder.

Static Code Analysis - Sonarqube

- open source platform for continuos inspection of code quality whether any unused package, unused variable, any todo statement in methods

- written in java lang,and support other lang like C#, PHP, Ruby

- provides reports for code quality such as duplicate code, how many unit test case are in ur project and what is % of unit test cases that is passed or failed

- provides code complexity which tells whether coding we have done is in simple way or complex way

- SonarLint plugin in eclipse - the developer can analyses the code inside eclipe

- Code coverage - The percentage of code which is tested by junit test is called code coverage

- Sonarqube dosent have any feature to do code coverage, then we have use jacoco or eclemma

- Sonarqube server is a central processing unit to process the analysis provided by sonar runner. Sonar runner is a tool which actually analysis your project code and send the report to sonarqube server to process the result

- 3 types of issues in Code quality

1. Bug - coding error will break ur code and needs to be fixed immediately

2. Vulnerability - A point in ur code that opens to attack

3. Code smell - an issue that makes ur code confusing and difficult to maintain

- Sonarqube paltform - contains 5 components

1. Analysers

- we run mvn sonar:sonar, Analyzer is responsible for running line by line code analysis which provide information about technical debt, code coverage, code complexity, code quality

2. database - when analysis is done , all info is stored inside database (by default H2 database)

3. server - provide final result in web page

4. plugin

5. sonarqube scanner(higher version) or sonarqube runner(lower version) - run ur anlayser, when we give mvn sonar:sonar, the scanner will enable by default and call analyser to analyse the report and store in db and finally server will display report in web page

- Serverity - Blocker, Critical, Major, Minor, Info

- Static code analysis tool

- Sonarqube - open source to check code quality and we can embed this to maven or gradle, so that we can use sonarqube along with maven project

- SonarLint - used with eclipse

- Sonar cloud - in cloud env to check code quality

Steps

1. Download sonarqube

2. Add <profiles> tag in pom.xml

3. start ur sonarqube - c:/sonarqube/bin/windows-x86-64/startsonar.bat

4. Go to project location - mvn sonar:sonar

In case of freestyle project we created everything individually,now we write groovy script and that script will do all task completely which is called pipeline project

- Groovy script provides 2 way to create pipeline

1. Scripted pipeline

2. Declarative pipeline

Scripted pipeline

- Pipeline script will be starting with node, inside this node whatever the task we are performing like pull,build, test, review code using sonarqube. Each and every task is called as stage