Plan – Code – Build ---- Integration ---- Deploy – Operate – Monitor

Planning and Coding – GIT, Surversion, JIRA

Build – maven, gradle,

Testing – Selenium, Junit

Deploy and Operations – puppet, chef, saltstack, ansible

Monitor – Sensun, Nagios

Integration - Jenkins

Planning and Coding – Continuous Development

Build and Testing – Continuous Testing

Integration – Continuous Integration

Deploy and Operate – Continuous Deployment

Monitor – Continuous Monitoring

Versioning Control System tracks the changes done on the code, if some faulty code is released, it can be rolled back using the versioning control system

GIT is the tool, GITHub is repository used as Version control system

Repository – Staging – Local Workstation

Continuous testing can find out the particular code which is breaking when the developers does continuous development

Continuous build – Continuous integration – Continuous testing >> More bugs can be found and find in particular area where the bugs are present

Automatic integration can also be scheduled and triggered.

For example,

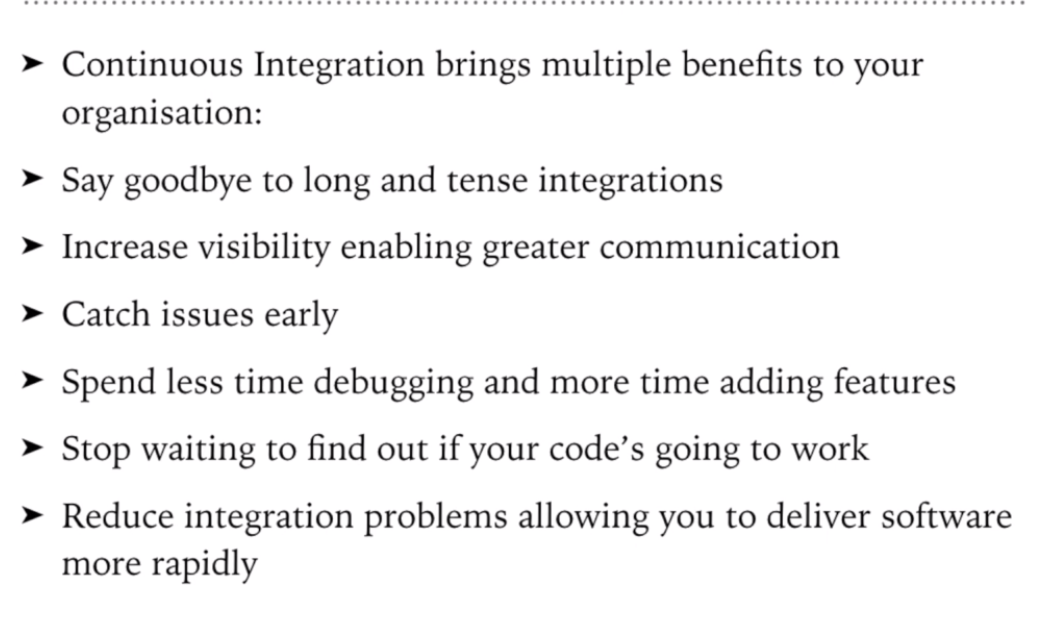
6AM Build A – Tested and no bugs

12PM Build B – Tested and no bugs

6PM Build C – Tested and found bugs – it will be easy for developers that coding done for Build C has errors and Code A, B are not involved

Because of continuous builds, integrations and testing – Deployment in production will be as scheduled and periodic without delays

Purpose of Continuous Integration



Continuous Deployment

* Configuration Management
* Containerization

Configuration Management

* Meet the functional requirements
* Release the deployment to all the servers either batches or on a go
* Schedule the updates on all the servers
* Consistency : Same resources, hardware and all should be consistent

Containerization

* Containerization is a set of tools which can do all the things for the application and is kept in a container
* Docker is the first containerization tool
* Schedule and consistency

Continuous Monitoring – Monitoring the product system and overall performance

For example monitoring of RAM, logs, taps, transactions, system memory, application memory usage, and more

GIT

Repository – Local and remote

Repository contains of files, history, configuration managed by GIT

Working Directory – All sourcing and planning is kept here

Staging Area – Once coding is done, it is saved here but this is not final commit

Commit – This is final version available on GIT

Remote Repository – The server where final code is kept (GITHUB)

Branch – Timely commits are done here

Need of version control system

Systems which defines the versions of data

Doc1 > Review done > Doc2 with revision > Review done > Doc3 with revision > >>>>>> Doc ‘x’

This is very tough (above versioning)

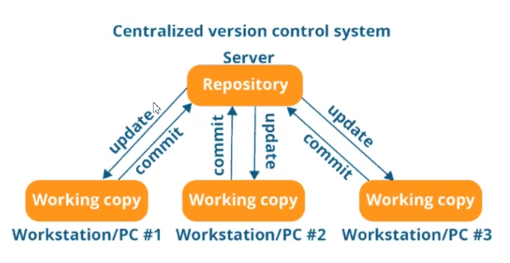
* Saves from creating multiple backups of files
* Allows multiple people work on same file (many people can be on sync)
* Tracks the changes and who have done the changes
* Easy to switch back to the older versions when required
* Improve the work productivity

GIT is a version control system with all the above mentioned advantages

Version Control System

* Centralized VCS
* Distributed VCS

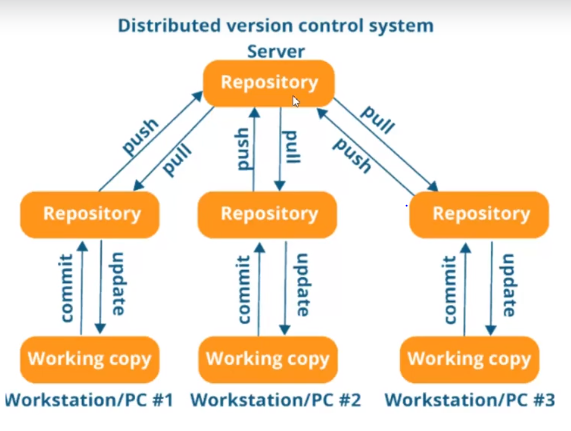
GIT is a distributed control system



Drawbacks of Centralized

* Server should always be up and there is no chance of local repository. If network is out of service, the work on centralized repository will stop completely
* If server is crashed, whole work done will be lost

Solution of these problems is the Distributed VCS



* Server –> Local Repository –> Working Copy ------Pull
* Working Copy -> Local Repository -> Server ------- Push
* Even when particular server is crashed, every one will have the required files
* Changes can be done locally without touching the server data

Linus Torvalds is the person who developed Linux and GIT

* GIT is a DVCS
* Free and Open to use
* Its fast
* Data assured since it is distributed VCS

GIT can be installed on Linux, Windows, Mac

Every command in GIT starts with command git

Git config for configuration changes

For example, git config –global user.name “Username” for changing the user name

git config –global user.email “email id”

git config –list this shows all the properties

You can give anything instead of user.name or user.email or anything. That particular field will be created and updated in the list

Pwd is the command used to check the directory (folder) we are working on

Ls is the list of directories present inside the directory shown as per pwd

Cd .. this command will take the directory to C/users

Go to required folder, right click and Gitbash here to select a particular folder. Enter git init and enter to make it the master folder for GIT

Ls – la for showing the hidden directories inside folder

Git status to find the status (tracked/untracked) of the files

Git add #filename to add a particular file and make it tracked

Git add . is for adding the whole directory

Git commit -m “Name of the commit” for committing the changes

After committing, if any changes are done on the file which was committed, GIT will recognize using git status

GIT Workflow

GIT has 3 stages – Working Area ; ->(Add) Staging Area ; -> (Commit) Repository

Working Area – Local area where we do changes

git status -----shows what all changes we have done in working area

git add <doc name> will send the file to staging

SKIPPPP>>>>>>>>>>>>>>

Maven – Build tool to help developers for building the projects. Tool to setup java project. Automates building the software project. To build a single executable entity, a tool is required. (JAR file example)

For example, if you are writing some code, directly we cannot take it and install on the server. That code needs to be built as a single executable file like .jar or .war etc. A build tool is required to make these files.

Write the code – Manage the code – Manage the dependencies – compile the whole project – executable package – deploy the code in the server

The application we develops will be depending on many factors such as open sour libraries, open source application, third party libraries, license libraries and many more. (Tomcat, etc). You need to download and import in your project normally (like inside eclipse to compile)

Example: Developer 1 will have a code, he includes all the third-party libraries, includes the dependencies and checks out the code, but Developer 2 is not having. So Dev2 will compile the code, hence all the dependencies will be gone. Also, whatever dependencies you have will not be present with all the developers working on the same code. These limitations can be overcome in Maven.

Maven have POM files where the dependencies and their version are pre-defined. Simply Maven is the tool used to build a project which includes managing the dependencies, compiling the code and create a single executable entity.

SKIPPPP>>>>>>

Jenkins – Continuous Integration

Continuous integration – Automated Build and Automated Tests

In this process, the code is collected from the shared repository, built using any tool like maven, execute the testing on the build generated.

Continuous integration helps to find the bugs easily and quickly where the error is present

Source code depository is handled by Jenkins, continuously it deploys in test environment and the testing will be performed continuously to find defects more easily and faster. Build can be scheduled for every hour or every day or once in a week or so on.

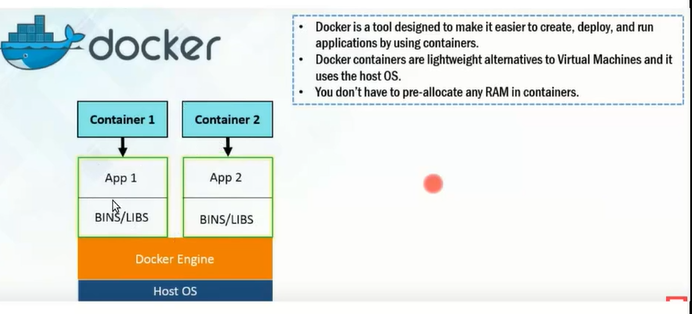
SKIPPPP>>>>

Docker

There might be some scenarios where the build developed might work in the developers local environment, but might not work in testing environment or production environment. This is because of the changes/differences in the computing environment. Docker can be used to overcome this issue. Docker contains a set of containers filled with the binaries and libraries which are used to run the build or application and the same containers are placed/installed in all the environments like dev environment, testing environment and production environment so that the build behaves similarly in all the environments.

Micro Services : For example, if we consider a website like Flipkart, the huge website can be divided into different servers or builds like Cart server, payment gateway server, products catalog, gift vouchers, etc. And all these micro services can be merged to form the full project. This will be very helpful in production if there is any bug in a particular component, a particular service will be affected without impacting the other functionalities. If there is a need of upgrading or downgrading any particular functionality, it can be done without affecting any other working features in the project. Many big companies like flipkart, amazon are using the micro services.

For each micro service, a virtual machine is required, and these virtual machines can be installed on host machines. Few micro services take more memory and few takes less. So in order to manage the space, ram, memory and all, docker can be the solution. All micro services can be installed on the same virtual machine running various docker containers for each micro service. Which means for example, 10 docker containers will be running on single virtual machine for 10 different micro services.



Docker Swarm

In order to manage the load on a particular website and automate the infrastructure, Docker swarm is required.

When Docker containers crashes or fails, docker swarm is helpful

Helps to upgrade the service with zero downtime

Manage the containers and load on VM’s and nodes

Docker swarm is a clustering and scheduling tool for docker containers

Orchestration : It is like defining the nodes and services, set up how many nodes you want to run, when to run and where to stop

Swarm uses multiple docker engines running on different hosts and lets you use them together.

Docker swarm have two types of nodes – Master or manager and Worker

Every swarm starts with manager node called as the leader.

Swarm works on Raft Algorithm

Raft Algorithm : Leader node continuously checks with the fellow manager nodes and sync with their states. If leader/master node goes down, then one of the fellow nodes will take its place.

Docker swarm uses

* Task scheduling
* Load Balancing
* Rolling updates : For example, there are 10 container in a server. If there are any updates, the two containers goes down and 8 more will be continuously running. So remaining containers will be still up and running with the old code so there is zero downtime. In the same way all the containers will be updated with zero runtime.
* Security. Docker has tokens which can be decrypted by docker swarm only.

SKIPPPPPPP>>>>>

Kubernetes

Kubernetes is implemented by Google whereas Docker swarm is implemented by Docker. Kubernetes is open source orchestration system.

Requirements of Text Editor in the GIT

Just to make it easy to use the GIT

Windows – Notepad++, Bash Alias

MacOS – TextMate 2

Linux – Sublime Tex

1. Add the Notepad++ installed path in Environment variables of desktop. And the check in GIT Bash by typing the command Notepad++. If installed, the command will be accepted. If not, command not found will be shown.
2. Configure Notepad++ with bash\_profile using command notepad.. bash\_profile and add the alias using command alias npp=’notepad++.exe –multiInst --nosession’
3. Configure notepad++ with GIT. Use the command git config --global core.editor "\"C:\Program Files\Notepad++\notepad++.exe\""
4. To check if the GIT is successfully installed or not, write a command git config –global –e the editor should be opened in notepad++

How to delete a file using GIT

Delete the file from the repository manually or use the command git rm <filename> and check the status. Delete file will be shown. Use the command git commit -m “Delete using OS” to delete the file completely

GIT is Distributed Versioning Control tool and GITHub is the repository hosting service or central repository or hosting platform

GIT provides the central server like cloud for the firms or companies to work on.

Local == Working directory (git add) – Staging Area (git commit) – Local repository (git push) 🡺 Remote repository

Remote repository (git pull) – local repository (git checkout, git merge) – Working repository

How to create central repository for GITHUB

To link the local repository to the remote repository command is git remote add origin <https://github.com/2AKP/Test1.git>

To pull files from remote repository to the local repository command is git pull origin <branchname>

To push files from local repo to remote repo, command used is git push origin <branchname>

Branches in GIT

For example, there are 4 teams working together for a project. And for example, let Master be the branch name where all the code is merged and is the main source of full code. If all the teams are committing few changes to the code, directly merging the code with the main master branch would be risky since there is a risk of defects. So each team can have different branches. Team 1 can have a branch named Team1 to commit their code, and then test the code, run the use cases and then when the code is tested, it can be merged with the master branch. Similarly all the other teams also can have their own branches, and merge the code with the master branch once the testing is completely performed.

Branches are just pointers, they will not take extra space.

Any file when synced with the branch of their particular team, the file will be pointed to that particular team1 branch only. Once the code or file of Team1 is merged with the master branch, then only the same file is pointing to the master branch and team1 branch also.

To create a new branch, command required is git branch <branchname>

To switch to any branch, command required is git checkout <branchname>

To see the existing branches, use the command git branch