# DEVOPS

What is Devops?

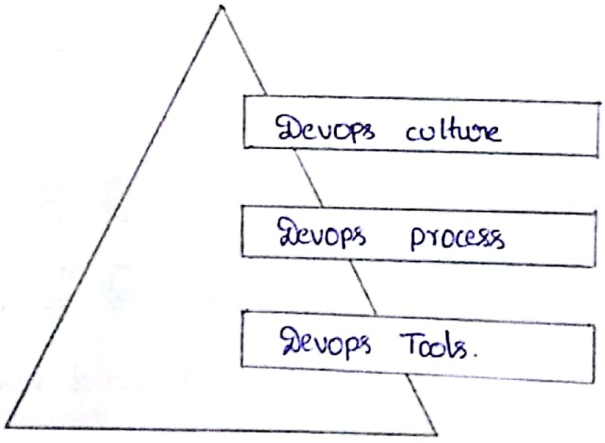
Devops is said to be a culture evoluing rapidly to automate the build, test and release phases of the application. Devops practices will help you to enable continuous Integration (CI), continuous Deployment (CD) and Infrastructure as Code (IaS) reducing the manual effort and improving the efficienty. There are numerous Tools to help us to automate various phases of SDLC. Organizations can use of any of the tools as part of Devops practices.

Why Devops?

The goal of the Devops is to improve the collaboration between development and operations team where the primary focus of both the teams is different. The Devops practices will help us minimizes the application down time and improving the availability, scalability and security of the applications.

Devops Needs:

We have to cover 3 important areas.




I. Devops culture:

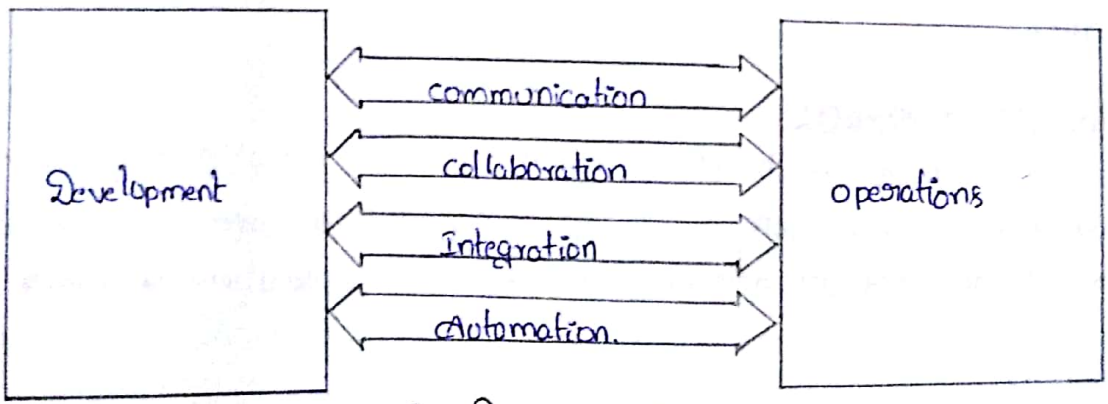



Fig: Devops culture.

II. Devops Process:

1. Continuous Integration (CI):

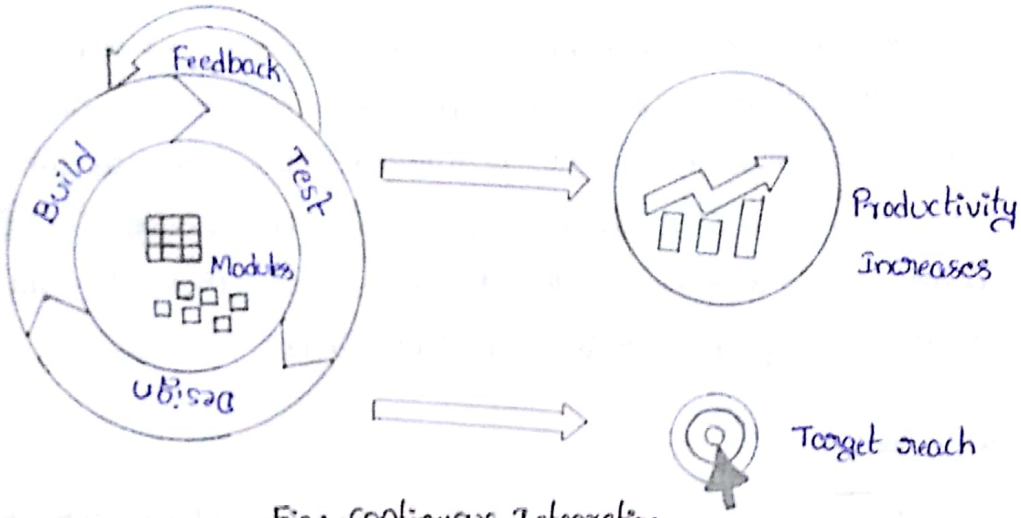


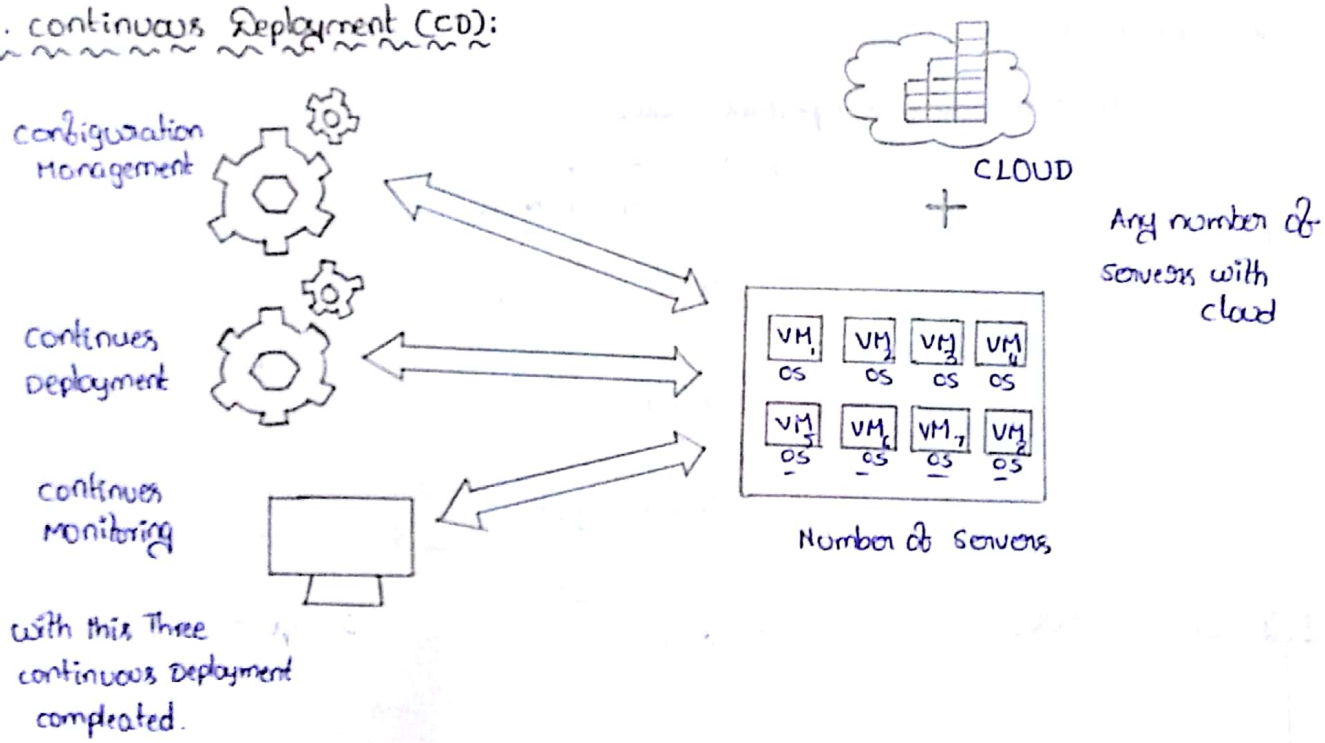

Fig: continuous Integration

Advantages:

(i) Increase in Productivity

(ii) Target reachability is more clear.

2. continuous Deployment (CD):




Configuration Management:

We have to push total information with one command one set of web servers what type of requirements we need and which type of pathes we requirment. {for) data base on desktops (configuration) } or → web servers)

Continues Deployment: After completion of build and test application is ready. That will be deployed in vertual image (or) container automatically. Then it will goes to UAT.

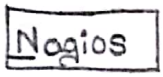
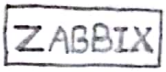
- From the continues deployment 'system standardization increase.' Automatically availability of systems for UAT & next level Testing.
- From this production deployment time will be fixed as early.

Continues Monitoring:

- From monitoring pro-active alerts will be taken for any problems.

III. Devops Tools:

| DEVOPS SERVICES | TOOLS |
|---|---|
| 1. Source Code Management (SCM) | git (distributed), mercurial, subversion (centralized), P4 |
| 2. Continues Integration | Jenkins, Teamcity, GitLab, Travis CI. SCM, Build Tools, Deployment Management all are Integrated & form a single pipeline if any code ready automatically Tested & Build & Deployed here means in this Tools |
| 3. Configuration Management | puppet labs, CHEF, SALTSTACK. Puppet, chef, saltstak, more tools are there. These are works in the way command lines, GUI, any ways. |

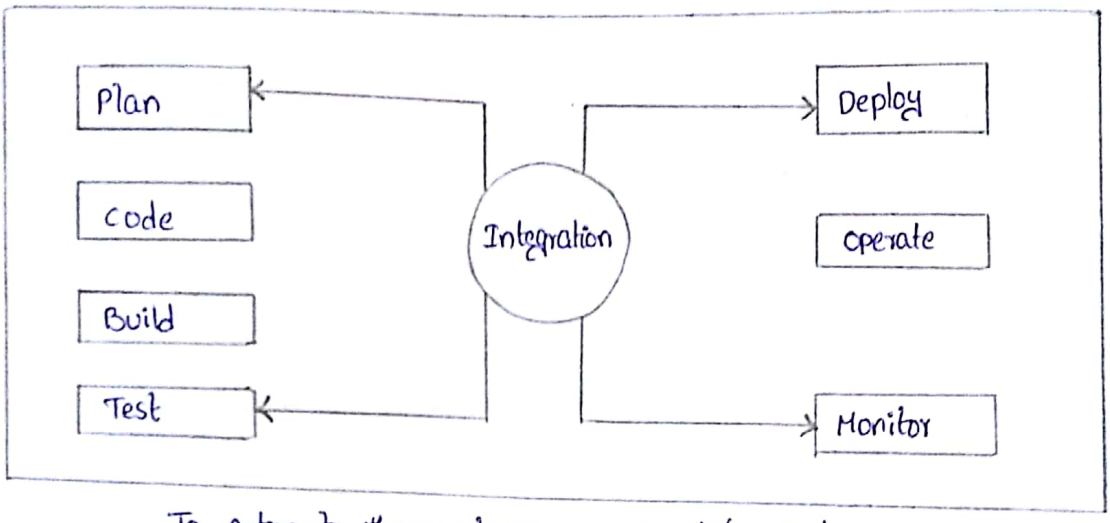
| DEVOPS SERVICES | TOOLS |
|---|---|
| 4. Continues Deployment |   Puppet   docker<br>- Normally we use puppet & chef at the same time for containerization we use 'docker'<br>- Application will be deployed in a container as it is, for creation of unmutable environment we use 'docker.' |
| 5. Continues Monitoring |   Nagios   Zabbix   PRTG Network Monitor |
| 6. Automation Scripting | Perl , Python , Roobee . Ansibil<br>- For the using these scripts these Integration process in Jenkins, Teamcity integration. we have to attached these scripts , then we improve the automation in the next level. |

Successful Devops Syntoms:

| | |
|---|---|
| 1. B Jenaralist | New Tools adoption, Every new problem gives new solution. |
| 2. continues Integration | Release Management , Agile Methodology . |
| 3. continues Deployment | Quick Deployments and easy Role-back . |
| 4. continues Monitoring | Monitoring services & alerting . |
| 5. Reciliance | Pro-active approach Recover from the incident Take caring of those incidents . |
| 6. Automation | Decreases of Repeated jobs / weast of money from activities / Time . |

Devops:

Devops is a Software Development approach which involves "continous Development", "continuous Testing", "continues Integration", "continuous Deployment" and "continuous Monitoring" of the software throughout its development lifecycle.




To Automate these stages we use below tools.

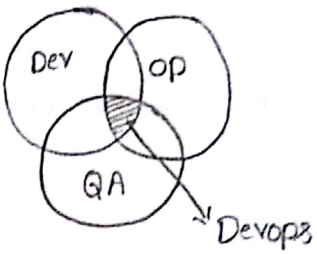
- For "source code Management" we have "Git" & "SVN".
- Both tools help use in "maintaing the code" throughout the "development" life cycle.
- Different revision's of the code can be stored and if we want to roll back on any changes.. then we just call previous version of the code & deploy to production.
- For Building we have tools like "Maven", "Ant" and "Gradle".
- Maven, Ant & Gradle tools help you package your code into executable files. which can then be produced into the testing environment.
- Then for "continous Testing" we have tools like "selenium" and "Junit".
- Then for "Continuous Integration" we have a very popular Tool called "Jenkins".
- continuous Integration: It means that you can test changes and then test that changes with other changes.
- continuous Deployment: we have tools like "Pupet", "Chef", "Saltstak" & "ansible".
- As & when the code is tested & ready. It is pushed to production or the non developer machine at this stage
- continuous Monitoring: We have tools like "New Relic", "sensu" and "Nagios".
- These Tools help us to monitor our "servers" closely & check the health of the system.

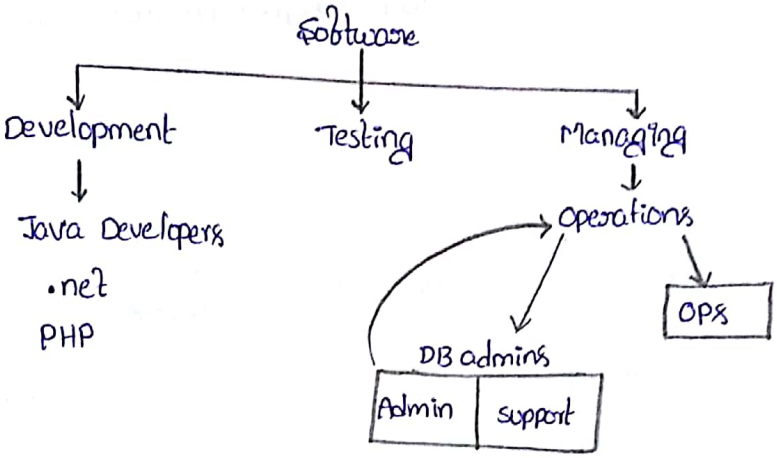
- Anything can happen to your servers at any time night. so its important they are monitored proactively rather than reactively.
- They also "improve productivity" and increase the reliability of the system & quiet possibly save lots of money and IT support costs.

9/1/18

# Devops

continuously provide support.




**Build:** The process of converting source code into an application is called as "Build."

Raw code → Build → Application

Developers → Build → Testing

Application server

| Application |
|---|
| Tomcat |
| O.S |
| H.W |

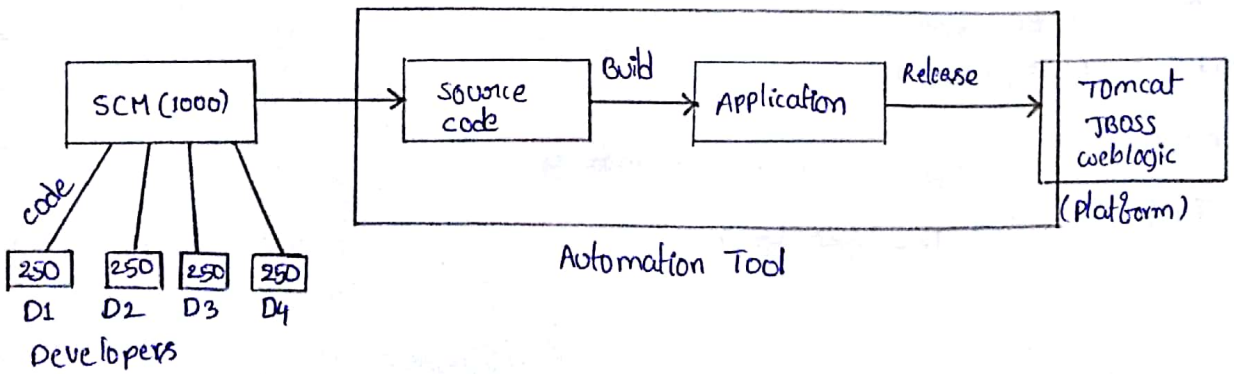
**Limitations:**

(1). Build Management
(2). Configuration Management
(3). Release Management.

→ Manual → Automation → Devops.

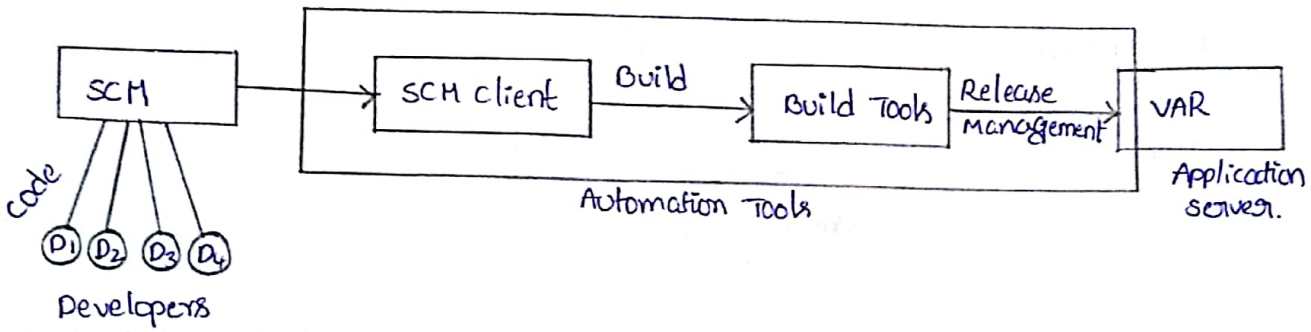
**Devops:** Devops is a process to automate the entire lifecycle of a software. Devops is process which is combination of different tools & Every Tool provides automation process

**Release:** The process of making your users to use is called as "Release."



Buid the source code → Develope an Application → launch the Application using the platform

SCM Tool: source code Management is a tool designed for managing source code.




We have
- SCM - 6 Tools
- Build - 4 Build Tools
- Release management - 3 Tools
- Automation - 3 Tools
- Application server - 9 tools
- configuration Management - 3 tools
- container management - 2 Tools.

- SCM
- Buid Tools
- Automation Tools
- Application server
- configuration Management
- container Management.

ANSI & Test — Binary files → which can't be edited directly interface to convert servers.

To Automate Application server we use tools called as "configuration Management" & "container Management".

SCM Tools:

1. Code version system (CVS) - oldest SCM Tool (not using)
2. SVN } are freeware
3. GIT } (mostly used)
4. clear case } high cost in licensed products.
5. TFS } (TFS used for .net)

## Buid Tools:

1. ANT
2. MAVEN
3. GRADLE

} Apache & freeware

4. MS-Build → Microsoft (used for .net)

ANT → (old Tool)
script based, procedural

MAVEN → (more than 80% used)
can be called as "project Management Tool"
declarative & NO scripting.

## Application / CI / CD Tools:

1. JENKINS
2. Hudson
3. Bamboo

} (Licensed products)

1. SCM Tool – SVN, GIT
2. Build Tool – client
3. Application Type – WAR, EAR, JAR
4. Application server – Tomcat
   JBoss
   WCS
   PRAMAT
   MS

## configuration Management:

1. CHEF
2. PUPPET
3. ANSIBLE

## containers Management:

1. DOCKER.

## ORDER

1. PRE-REQUISITES
2. Agile
3. LINUX
4. SVN / GIT
5. MAVEN
6. TOMCAT
7. JENKINS
8. CHEF
9. DOCKER
10. ITIL

22-Jan-2018

# PRE-REQUISETS

## Types of companies:

Software companies are aggrigated as different categories. Some of them are

1. consulting companies
2. Product based companies
3. corporate companies

### 1. consulting companies:

(i) consulting companies always get the projects from different companies & they hire the employees to work on the project. with the help of employees the project will be compleated & delivery the same to the organization.

(ii) These companies access collaboration between the project & employees.

(iii) consulting companies will never have their own projects, they always work other companies.

Ex: Wipro, capgemeni, HCL, etc.,

### 2. Product based companies:

(i) Product based companies will have their own products developed and sell it to the different customers.

(ii) They also provide support every year to the customers, who bought their products.

Ex: oracle, IBM, CA, QUIEST, etc..

"Infosys" Acts as a "consulting & product based company."

### 3. Corporate companies:

(i) These companies always work for their own organization & they never work for outside organizations.

(ii) Their projects also coming from their own organizations.

Ex: Bank of America, Frankline, JPMorgan, etc..

## Difference Between clients & vendor :

(i) The organization "who provides" service that is called "vendor".

The organization "who receives" service that is called "client."

(ii) For example if ICICI project is given to the Infosys, because Infosys providing a service is called it as "vendor" and because ICICI receives a service is called it as "client".

(iii) Depends on requirement a vender can have any number of clients and client can have any number of vendors.

24-Jan-2018

# SUPPORT MODELS

Difference Between onsite & off-shore Model:

(i) If an employee travel to client location either on short-term bases (or) on long term bases is called as "onsite".

(ii) The term onsite is not depend on any location.

(iii) If an employee working at vendor location is called as "off-shore."

(iv) Organization follows different models in term of providing support (or) administration.

There are Three types of Models.

1. On-site Model
2. Off-shore Model
3. On-site & Off-shore Model

1. On-site Model:

In on-site Model all the employees working from "client location". And providing support which is called it as a "On-site 'support Model."

2. Off-shore Model:

If all the engineers providing support by "sitting at off-shore location" is called it as "off-shore support Model."

3. Onsite & Off-shore Model:

If the engineers "sitting at both the locations" & providing support is called it as "On-site & Off-shore support Model".

Different organizations follows different support models according to their project requirement.

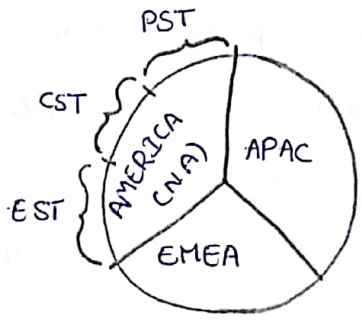
25-Jan-2018

# TIME ZONES

Every country has its own the Geographical time code. Depends on number of countries we have, that many time zone codes available. Remembering all time zones. It may be a difficult process. Because of which all 190 countries are divided into 3 main regions or zones.

There are 3 types of zones available.

1. APAC - Asia phasiphic
2. EMEA - Europ Middle East & Africa
3. AMERICA (NA) - North America.




NA

EST - Eastran standard Timezone

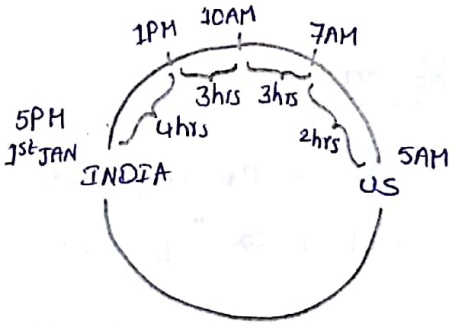
CST - central standard Timezone

PST - phashiphic standard Time zone.

If we are working from India & supporting APAC project to support in their working hours. we may have to login early hours of IST.

If we are supporting Europ based project & to support Europian working hours we may have to login Noon hours of IST.

If we are supporting American based projects & to support them we may have to login in the evening hours of IST.




Ex: 1AM 25-Jh IST

3:30PM 24th → EST

2:30PM 24th → CST

12:30PM 24th → PST

Ex: (i) To support ANZ (Australian Newzeland) we may login by 4AM. IST.

(ii) To support Bank of london we may have login by 2PM IST.

(iii) To support Bank of America we may have login around 6PM IST.

Some of the standard time zones are

IST - Indian standard Time

AUST - Australian standard Time

CST - chaina standard Time.

From US there are 3 main zones are available.

EST - Eastrian standard Time

CST - central standard Time

PST - Phasiphic standard Time.

Q. what is 24/7 coverage?

Ans. Some of the projects may reauire round the clock support as they are critical applications & may reauire support at any point of time. To achive this organizations will have employees working in every timezone to cover all 3 zones support.

If we are supporting from India we may work in APAC, EMEA & NA time zones.

Q. what is shift Rotation?

Ans. To provide eaual opportunity to every employee in an organizations it is prefered to work in rotation bases as every employee has an opportunity to work in every time zone.

Advantages with shift Rotation:

(i) Every employee gets an eaual opportunity to work in different region.

(ii) Every employee gets an opportunity to know the issues occuring in the project.

(iii) Every employee gets an opportunity to meet with client or) customer.

27-Jan-2018.

# TECHNICAL PRE-REQUISITES

Computers comes as 2 types.

1. PC
2. Server

- PC comes with a limited configuration which is suitable for using to perform independent activities as one person. The PC access is (may be) limited to few users based on requirement.
- Server comes with high configuration to accomidate multiple user request to use in parallel.
- When we join in organization 'PC' will be given to us to perform our work. using PC we connect to the server and perform server side activities.
- Based on number of clients and projects we support. There may be many servers in the organization.
- When servers are setup with required configuration. we need keep on upgrading them as when required with the help of system admins.

Configurations:

| PC | | SERVER |
|---|---|---|
| i7 | — | BLADE (mostly commonly used all companies) |
| 16GB RAM | — | 256 GB RAM |
| 2TB RAM | — | (-1) (Unlimited) |
| 4 CORES (speed) | — | 32 CORES |
| Intel / AMB / QUDE / NVIDIA (manufacturers) | — | IBM |

29 & 30 - JAN - 2018

# DATA CENTERS

## Data centers:

Data center is a location or place where we manage the servers in organization. Data centers are designed as per the specification and some of the common standards are

(i) Data centers are designed to avoid any disasters occured.

(ii) It's a secured area where most of the people not allowed to go to data center.

(iii) Data centers are designed to avoid exposer to outside to avoid virus.

(iv) Data centers are designed as high chilled rooms. which can be use to avoid heat on the servers.

There are 3 types of Data centers available.

1. Non-Prod (production) Data center.
2. Prod Data center.
3. D-R (Disaster-Recovery) Data center.

## 1. Non-Prod Data center:

Non prod Data centers are used to manage the servers used for Non-production activity. like development, Testing, etc., These data centers are managed at the location developers & testers. for better access connectivity.

These data centers are not expose to the public as their access limited with in Organization.

## 2. Prod Data center:

Prod data center is used to manage the servers used for production. These data centers are placed near to the endusers for a better access.

### 3. D-R Data center:

D-R Data center is used to manage the servers act as a back up. when production data center is down then D-R act as a primary to handle user request. D-R data center is also open for public access in case of production data center is down.

### Data center Architecture:

Data centers working in 2 architectures.

1. Active - Passive.
2. Active - Active.

### 1. Active - Passive Architecture:

In Active - Passive architecture, only production data center is active. and D-R data center will be acting as a passive. when user request comes, it is always goes to the active data center in the event of active goes down those request will be send it to passive.

In Active - Passive model one data center is always "Idle" not handling user request, to over come this most of organizations depends on Active - Active architecture.

### 2. Active - Active Architecture:

In this Architecture both the data centers are handling user request. And there is an option of distributing traffic to both of the data centers.

### D-R Drill or) make test or) D-R Exercise:

In Real situations disaster may not happend & in an active - Passive architecture. we may not get a chance to fail over to D-R & user request may not go to D-R data center. At later point of time when we want to fail over D-R. we may not 100% sure as the D-R will work as expected, to over come this organizations want to test the D-R frequently & make sure it is working as expected. to achive this organizations follow an approach called "D-R drill or) Exercise".

During D-R Drill we perposefully failure user request to D-R data centers and check how it is functioning. The frequency of D-R drill is either 3 months, 6 months (or) yearly.

1-Feb-2018

# ENVIRONMENTS

Non-Prod DC
1. Unit test / Dev
2. QA
3. UAT / Stage (User Acceptance Test)
4. PERE (Performance Test) (simulation)

Prod DC
5. PROD

DR DC
6. DR

Q. what are the diff environments you have face?

A. Total 6.

- TO support SDLC process organizations may have different environments to support software development life cycle.
- To meet SDLC we may perform different activities and for each activity, there are different environments used.

* Total no. of servers supported in project
  6 x 4 x 15 = 360
  where, 6 environments
  4 projects
  15 servers

Some of the known environments are,

1. Dev:

It is used by development team to perform unit testing. It is mainly used by development team & access is also given to development team only.

2. Q.A:

- It is used by manual testing team to perform manual testing on application or) project.
- QA access is also open to testing team only.

3. UAT / Stage:

- It is used by the client to perform user acceptance Test or) client validation.
- The access will not be given to development team & testing team it is always given to the client only.

4. PERE:

- It is used by the performance testing team to perform different levles of testing like load, endurance & performance.
- The access of performance environment is given to performance Engineering team only.

All above 4 environments together called as "Non-Prod" (or) "Pre-Prod" Environments and they are placed into Non-Prod data center only.

5. PROD:

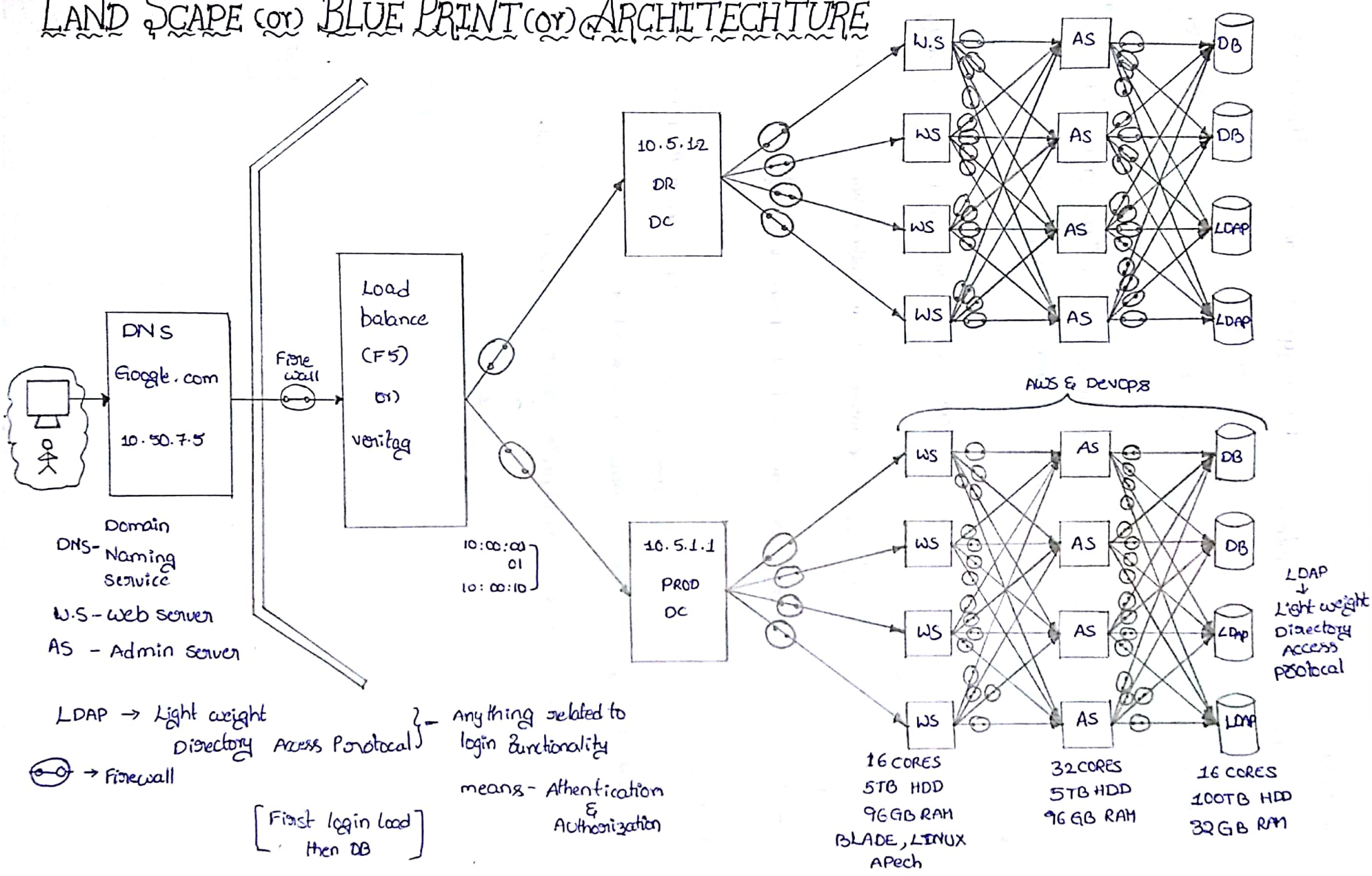
- It is used for running the applications on production, which will be used by customers (or) users.
- The access will not be given to the internal team rather than open to the public access.
- All prod environment users are placed into Prod data center.

6. D-R:

- It acts as a back-up data center and access will be given to the public in the event of prod data center is down (or) unavailable.

90% standard organizations used these above Environments

# LAND SCAPE (or) BLUE PRINT (or) ARCHITECHTURE




DNS - Domain Naming Service

W.S - Web server

AS - Admin Server

LDAP → Light weight Directory Access Protocal

→ Firewall

[First login load then DB]

Anything related to login functionality

means - Athentication & Authorization

LDAP → Light weight Directory Access protocal

5&6 - Feb - 2018

# PROJECT MANAGEMENT PROCESS

(or)

# METHODS

For Project Management different organizations follow different process to perform Project Management. In olden days organizations use to follow different approaches for performing Project Management. The approaches use to be called as "Treditional Approaches".

Example of Treditional approach is "waterFall".

Limitations in Treditional Approach:

(i) In Treditional Approach multiple features (or) changes group together as bigger project.

(ii) Multiple changes (or) features will be released as one project.

(iii) Though changes are compleated early, they have to wait untill complition of all changes to get released.

(iv) If any one change is having trouble, all changes need to be Roll-back, because we cambine all the changes.

(v) If any issue happend it is difficult to understand from which change the issue is occuring.

To over come above limitations there is a new process initiated (or) identified is called as "Agile".

6-Feb-2018

# AGILE

## Advantages with Agile:

(i) Agile recommands to divide the changes as different parts.

(ii) Agile recommands to have a separate release for every part divided.

(iii) If any change occured or issue happend only individual release get impact & other changes will not be impacted.

(iv) If any issue happend we clearly know from which change the issue happening & we can start working on solution.

(v) on priority bases (or) as business requirement the changes (or) features can be release to the production.

With all above advantages every organization started following Agile as a project management process.

## Roles in Agile Process:

Agile process has 2 Roles.

1. Agile coach
2. Agile member.

### 1. Agile coach:

(i) Agile coach plays a manager role & he is responsible for managing multiple applications.

(ii) Agile coach acts as escalation point, if any issue happend in the project Agile coach will involve & sort the issue.

### 2. Agile member:

(i) For every (paint the) project there will be a different Agile member available.

(ii) Agile member involve into the Day-to-Day activity of the project.

1. What is product owner?

A. The Application owner (or) Any one can take decission on project (or) product is called as "product owner".

In organization there can be one (or) more than one products depends on organization.

2. What is story?

A. A detailed explanation of the change (or) feature is called as "User story". As a technical team (or) Agile team we may go through user story & understand the functionality if user story is not clear we many reach out to the "Business Analyst", for any user clarification of user story.

3. What is Feature List?

A. The list of changes or Feature expected by the product owner will be written as a document and send it to Agile team which is called "Feature list."

Feature List does not follow any order as product owner remember the features & start preparing a document.

4. Priority List ?

A. After Feature List is received Agile team will set up a meeting involving all required teams & come up with the priority of different item & come up with a document as for the priority, which is called "Priority List."

5. What is sprint?

A. One project consist of many changes will be divided as sub-projects containing one feature (or) multiple features are called as "Sprint."

One project can be divided as many sprints & each sprint can contain one (or) multiple features.

6. What is Release?

A. After development & testing happends on a sprint it may be ready to released to the production. The process of moving sprint changes to production is called as "Release".

7. What is Sprint Roll-back?

A. If any issue happend in a sprint Release & if issue fixing taking time the entire sprint may get Roll-back is called as "sprint Roll-back."

8. What is Backlog item?

A. If any Feature is suppose to be released on a given date because of some reason if that is not getting released that feature or item is called as "Back-log item."

Agile member will keep on review in changes / backlog items & see if can be accommodated part of Feature sprints.

9. What is sprint-scope?

A. When there are two changes part of the sprint we define the scope of sprint. Later based on requirement we may add more features (or) remove features is called it as "sprint-scope change".

7-Feb-2018

# LINUX

Any Devops Tools we are going to support many run either on windows (or) LINUX operating system.

We may have to support those tools & to support we may need to have an idea of operating system usage. Most of organizations may prefer using LINUX as an operating system.

Differences between windows & LINUX:

(i) LINUX comes with inbuit firewall, which is used to protect the data.

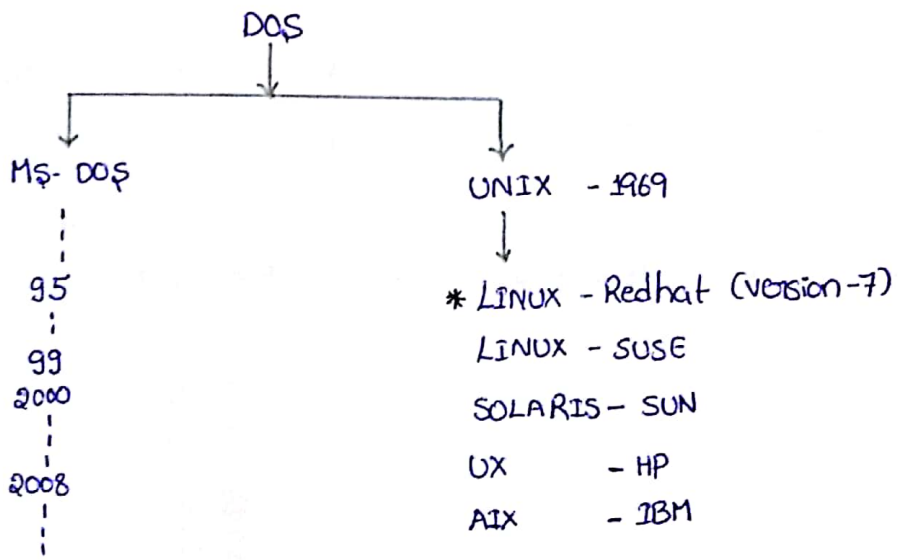
(ii) LINUX manges data in a "Tree structure" which will help to improve the performance.

(iii) By default LINUX supports concurrent access of multiple users to make use of the resources.

(iv) LINUX is also very good interms of performing the process calculations as how long the process may take time to complete.

With all above advantages most of organizations prefer to use LINUX as their operating system.

History of Operating system:



In Organization we may perform activities on our laptop running with Windows but our servers may run with LINUX. Establishing connectivity between Windows to Windows & LINUX to LINUX is not a problem. But establishing connectivity between Windows to LINUX is a challeng & may not be able to connect directly.

To establish a connectivity we may depend on 3rd party connectivity tools like PUTTY, SSH (Secure Shell).

Both PUTTY & Secure Shell are freeware it can be dowload it & set up on our system.

To connect to the LINUX servers we may need the following details.

1. IP address (or) server name.
2. Username.
3. Password.

* By default LINUX servers accept the connection on "PORT-22".

8-Feb-2018

# LINUX COMMANDS

@Admin - Ubuntu.

1. How to check Host Name (or) Sever Name (or) Computer Name?

A.

Command (or) Syntax :

administrator@ubuntu : ~$ hostname

output : Ubuntu.

2. How to check the operating system (OS) Name?

A. command (or) Syntax:

administrator @ ubuntu : ~$ uname

output : LINUX.

3. How to check the operating system version?

A.

Syntax : 1

command (or) Syntax 1 :

administrator @ ubuntu : ~$ uname -a

output :

Linux ubuntu 3.0.0-12-generic #20 -Ubuntu SMP Fri Oct 7 14:50:42 UTC 2011 - - - -

Syntax : 2

command (or) syntax 2:

administrator @ ubuntu : ~$ cat /etc/redhat-release.

output :

Linux ubuntu 3.00-12 - generic # 20 -ubuntu SMP Fri oct 7 14:50:42 UTC 2011 - - - -

4. How to check the Bit of operating system?

A.

command : ~$ uname -a

output :

Linux ubuntu 3.0.0-12-generic #20-Ubuntu SMP Fri oct 7 14:50:42 UTC 2011 i686 i686 i386 GNU/Linux

This is chars sets.

- In place of this we have x86-64 it means 64 bit
- otherwise 32 bit like as above o/p.

(i) 64 bit - more extendable configuration.

(ii) 32 bit - Limited configuration.

5. How to check the Date of a server?

A.

command : ~$ date

output : Thu Feb 8 09:19:46 IST 2018.

6. How to display the output?

(or)

How to print?

A.

command : ~$ echo "Hello good Morning" (or) echo Hello good Morning.

output :

Hello good Morning.

7. How to check the History of commands?

A.

command : ~$ history

output :

```
  :
1999  hostname
2000  uname
2001  uname -a
2002  date
2003  echo "Hello good Morning"
2004  echo Hello good Morning
2005  history
```

Delete History:

This performs only admins not for all. we don't have access.

command: ~$ history -c

8. How to clear the screen?

A. command: ~$ clear (Just move up to the fresh screen)

output :

Gives fresh screen.

9. How to check All users currently Logged-in?

A. command : ~$ who

output:

administrator pts/0 2018-02-08 09:01 (:0)

(who are all logged into server & How many members also)

which is helpful in when we are rebooting, as a devops engineers we don't want.

10. How to check the server Processor Name?

A. command : ~$ cat /proc/cpuinfo

output:

All processor details.

⋮

(manufacturer) vendor_id : GenuineIntel CPU family :6

model : 23

model name : Pentium (R) Dual-core CPU

cpu cores : 2

⋮

11. How to check the Number of CORES?

A. command : ~$ cat /proc/cpuinfo

output :

⋮

cpu cores : 2

⋮

RAM:

12. How to check RAM size of Server?

(or)

How to check the Memory?

A.

① command : ~ $ free (It shows in bytes)

output :

| | total | used | free | shared | buffers | cached |
|---|---|---|---|---|---|---|
| mem : | 2026352 | 1572548 | 453804 | 0 | 6373116 | 674140 |
| -/+ buffers/ cache: | | 261092 | 1765260 | | | |
| swap : | 262136 | 0 | 26 2136 | | | |

② command : ~ $ free - m (It shows in MBs)

output :

| | total | used | free | shared | buffers | cached |
|---|---|---|---|---|---|---|
| mem : | 1978 | 1535 | 443 | 0 | 622 | 658 |
| -/+ buffers/cache : | 0 | 1724 | | | | |
| swap : | 225 | 0 | 255 | | | |

③ command : ~ $ free -g (It shows in GBs)

output :

| | total | used | free | shared | buffers | cached |
|---|---|---|---|---|---|---|
| mem : | 1 | 1 | 0 | 0 | 0 | 0 |
| -/+ buffers/cache: | 0 | 1 | | | | |
| swap : | 0 | 0 | 0 | | | |

④ command : ~ $ vmstat (vmstat - vertual Memory statistics)

(It not give total size o/p)

output :

| procs | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| r | b | swpd | free | buff | cache | si | so | bi | bo | in |
| 1 | 0 | 0 | 453804 | 637476 | 674272 | 0 | 0 | 98 | 13 | 122 |

⑤ command : ~ $ top

output :

top :

Tasks :

CPU(s) :

Mem : 2026352k total, 1577200k used, 449152k free, ---- buff

Swap :

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

Hard Disk:

Mounts in LINUX like in C-drive, D-drive in desktops, windows.

Mounts in Linux = Drives in windows.

Mounts:

(i) In LINUX every partition of Hoarddisk is called as "Mounts".

(ii) When Linux is installed there will be few mounts created by default and we can also created our own mounts with our own naming convention.

(iii) Once mount is created we can either allocate the size, increase the size & decrease the size.

(iv) Mounts always prefix by "root"(/).

Hoarddisk
↓
Root
↓
Mount
↓
Folder

Ex:

~ $ /home → mount.

~ $ home → folder.

Ex:

~ $ /home/naren/movies....

13. How to check the mount occupency (or) the usage ?

A (1) command : ~ $ df -k (K - kilo bytes)

output :

| filesystem | 1k-blocks | used | Available | use% | Mounted on |
|---|---|---|---|---|---|
| /dev/loop0 | 17884496 | 15405944 | 1570060 | 91% | / |
| udev | 1006176 | | | | |
| tempfs | | | | | |
| home | | | | | |
| none | | | | | |
| /dev/sda5 | | | | | |

② command : ~$ df -h (h - human Readable)

output :

| Filesystem | Size | Used | Avail | Use% | Mounted on |
|---|---|---|---|---|---|
| /dev/loop0 | 1.8G | 1.5G | 1.5G | 91% | / |
| udev | 983M | 4.0K | | | |

14. How to check the Present Working Directory?

A. command : ~$ Pwd

output :

/home/administrator

↓ ↓

Mount Folder

## Directory:

15. How to create Directory?

A.

command : ~$ mkdir directoryname

Ex: ~$ mkdir batch42

Then check output

~$ Pwd.

* we have to create multiple folders at a time

command : ~$

mkdir -P batch42/teluge/movies ...

Here, P → Path.

16. How to Delete a directory?

A

command : ~$ rm -rf directoryname

Ex : ~$ rm -rf batch42

here, rm - remove

r - recursive

f - force

17. How to change Directory (or) get in to directory?

A. Command : ~$ cd directoryname

Ex :

administator@ubuntu : ~$ cd batch42

Then check output.

administator@ubuntu : ~/batch42 $ Pwd

o/p:

home/administator/batch42.

18. How to come back from Directory?

A.

command: ~$ cd..

Then check output

~$ pwd

O/p:

/home/administator

* If multiple Folders

command: ~$ cd../../../.. ------

How to create a File:

Files can be created in different commands

(i) TOUCH
(ii) CAT
(iii) Vi
(iv) CP
(v) mv
(vi) >>
(vii) >

>> → add a content aperding from bottom

> → new content, exise the content, overriding

(i) Touch: Touch will be initially create empty file. (or) zero byte files.

command: ~/batch42$ touch filename

Ex: ~/batch42$ touch touchexample.

(ii) cat: cat command displays the content of a file.

command: ~/batch42$ cat >> catexample

~/batch42$ cat >> catexample1

~/batch42$ cat > catexample2

creating new file with catexample.

↓

control+d

↓

administator@ubuntu: ~/batch42$ date

Tue Feb 13 09:11:01 IST 2018

:~/batch42$ date >> output

ls-l ↵

O/P: one file created.

:~/batch42$ hostname >> output

* control+d

↓

save & exit

(iii) vi :

vi means view & insert

vi learn in 3 three days.

(iv) cp: (copy) cp command copy the data from one file to other.

command : ~$ cp filename copied location/filename (or) new filename
(or)
backup filename.

Ex: ~/batch42$ cp catexample cpcatexample
↓
for backup

o/p: ~/batch42$ ls -l

------- catexample
---
---
------ cpcatexample
---

For backupexample :

~/batch42$ cp catexample catexample.back.13Feb2018

~/batch42$ ls -l

------- catexample
---
---
---- cp catexample.back.13Feb2018
------ cpcatexample
---

(v) mv: (move)

command : ~/batch42$ mv catexample catexample111

Then check output.

o/p: ~/batch42$ ls -l

----- catexample111
---
---
---- cp catexample.back.13.Feb2018
---- cpcatexample
---

mv:

(i) mv command move data from one file to other, old file not exists

(ii) Rename old file to newfile.

19. How to nullify a File?

A.

command: > filename

Ex: ~/batch42$ > cat example1

~/batch42$ ls -l

o/p:

content will be deleted

which causes all the data in a filename file.

20. How to delete file?

A.

command: rm filename rm - remove

Ex: ~/batch42$ rm catexample

Then check output

~/batch42$ ls -l

21. How to read the content of a file?

A To read the content of a file we use below commands.

(i) cat
(ii) more
(iii) less
(iv) head
(v) tail
(vi) vi

(i) cat:

Ex: ~/batch42$ cat catexample111

o/p:

Hello

(ii) more:

Ex: ~/batch42$ more catexample111

more
↑
↓
less

(iii) less:

Ex: ~/batch42$ less catexample111

(iv) head: head command displays top 10 lines of file.

Eg: ~/batch42$ head -5 cat example111

command: head filename

Ex: administator@ubantu :~/batch42$ head -5 catexample111

o/p: it displays first 5 records.

This is an example
Hello
fsdf
df
asd

(v) tail: tail command displays bottom 10 lines of file.

* Command: tail filename

Ex : ~/batch42$ tail catexample111

o/p:
By default it displays the last 10 records.

* command: tail -f filename.

Ex: ~/batch42$ tail -f catexample111

o/p:

* Displays the last 10 lines & waiting for any updates.

* Command:
tail -2 filename
(or)
(any number)

Ex: ~/batch42$ tail -2 catexample111

o/p:
ggf
end of the file.

* command:
tail -2f filename

Ex: ~/batch42$ tail -2f catexample111

o/p:
Displays the last 2 lines & waiting for any updates.

ls (list): 'ls' command 'list all' the files & directories available in current directory.

22. How to list files?

A

command: ls filename → displays short format (name)

ls -l filename → displays long format.

Ex: ~/batch42$ ls

~/batch42$ ls -l

* By default it displays the alphabetical order.

* If we want to reverse order.

command: ls -lr

23. How to Filter based on size?

A. command: ls -lS (S → size.)

↓ uppercase

o/p: it displays order in size.

command: ls -lSr

o/p: it displays reverse order in size.

24. How to sort files based on Modified date & time?

A

command: ls -lt filename

In reverse order.

command: ls -ltr filename

25. How to list all files?

A. command: ls -la

o/p: displays all files & hidden files also

* hidden files always start with '.'

Ex: $ touch .myhiddenfile

14-Feb-2018

# USERS IN LINUX

In Linux there are 3 types of users available.

1. Root Level User
2. Group Level user
3. User Level User.

- As Devops Engineer a user level ID will be created & added to any group
- Linux manages all the users in a file called "Passwd", which is available under "ETC mount."
- Linux manages all the passwords in a file called "Sudoers", which is available under "ETC mount."

## Permissions:

In Linux we have total 3 permissions available.

1. Read (r)
2. Write (w)
3. Execute (x)

For any file or folder, there will be total 9 permissions at root level, group level & user level.

### order of permissions

Root (or) owner permissions: Actions the owner of the file (or) perform on the file.

Group permissions: A member of the group that a file belongs to, can perform on the file.

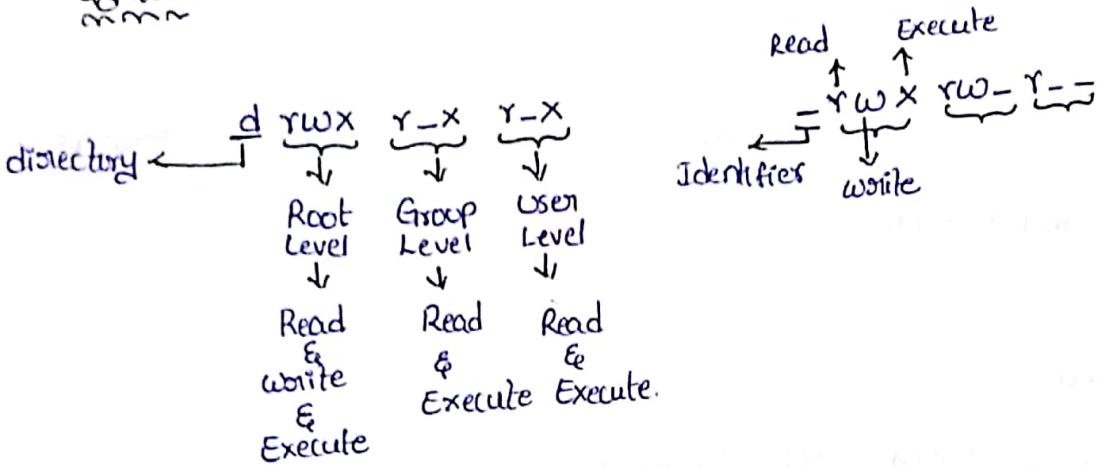
User (or) Other permissions: what action all other users can perform on the file.

Permission values:

1 - Execute
2 - write
4 - Read

4+2+1 = 7

Syntax:




26. How to change the ownership of a file?

A.

command :

Ex:

:~/batch42 $ chown naren : naren filename.

user name — Group name

27. How to change permissions?

A:

command :

Ex : ~/batch42 $ chmod 000 catexample1

~/batch42 $ ls -l

O/P :

---------1 administrator administrator 0 2018---- catexample111

~/batch42 $ cat catexample1

In Linux Permissions are.

- command:

~/batch42$ chmod 000 catexample1

~/batch42$ ls -l

output:

---------- 1 administator administator 0 2028. catexample1

No permissions

- command:

~/batch42$ chmod 111 catexample1

~/batch42$ ls -l

output:

---x--x--x

only execute permissions

- command:

~/batch42$ chmod 222 catexample1

~/batch42$ ls -l

output:

--w--w--w-

only write permissions

- command:

~/batch42$ chmod 333 catexample1

~/batch42$ ls -l

output:

-wx-wx-wx

write & execute permissions.

- command:

~/batch42 $ chmod 444 catexample1

~/batch42 $ ls -l

output:

-r--r--r--

only read permissions

- command:

~/batch42 $ chmod 555 catexample1

~/batch42 $ ls -l

output:

-r-xr-xr-x

read & execute permissions

- command:

~/batch42 $ chmod 666 catexample1

~/batch42 $ ls -l

output:

-rw-rw-rw-

read & write permissions.

- command:

~/batch42 $ chmod 777 catexample1

~/batch42 $ ls -l

output:

-rwxrwxrwx

read & write & execute permissions

- command:

~/batch42 $ chmod 888 catexample1

~/batch42 $ ls -l

output:

Invalid mode

In Linux,

Q → quit

wq → write & quit (or) save

Linux permissions:

0 - no permissions

1 - execute

2 - write

3 - read write & execute

4 - read

5 - read & execute

6 - read & write

7 - read, write & execute all permissions

8 - Invalid mode

## U-Mask:

By default permissions will be applied any file (or) folder created to have those default permissions applied administator may setup the default permissions at folder level (or) Mount level using U-Mask settings.

When permissions are setup with U-mask any file (or) Folders created will automatically gain the default permissions.

## File Meta data:

Any data that can be used to identify a file or folder is called as "Meta data". Some of the meta data includes filename, date & time, size, ownership, permissions, etc.

Ex:

```
drwxrwxr-x  2 administator administator  4096 2018-02-14 08:54 backup
-rwxrwxrwx  1 administator administator     0 2018-02-13 10:21 catexample1
```

## content search:

To search content in a file we use "GREP" command.

GREP: [GREP - Globally Search Regular Expression & Print]

Syntax: grep "pattern" (or) "content" filename.

- command: ~/batch42$ grep "end" catexample111

  output: end of the file.

- command: ~/batch42$ grep "END" catexample111

  output: NO o/p.

- command: ~/batch42$ grep -i "END" catexample111

  output: end of the file.

  Here, i → ignore upper (or) lower cases.

- command: ~/batch42$ grep -ic "END" catexample111

  output: 1

  Here, c → count

- command: ~/batch42$ grep -iwc "END" catexample111

output: 1

w → word

Here, count → matches character by character

word → matches only words.

28. How to find the File location?

A. command: ~/batch42$ find . -name catexample1

output:

./catexample1

command: ~/batch42$ find . -mtime +1 — Here, m → modified.

find . -mtime +2 → days

find . -mtime +0

output:

./catexample1

⋮

~$cd batch41

command: ~/batch41$ find . -mtime +90

output:

./catexample1

./redirefex

29. How to Filter based on Size?

A. command: ~/batch42$ find . -size +20

↓ give sized file above 20MB.

Help in Linux:

For every command Linux provides the complete information as usage, options, etc., The command to get "Help" is "man".

syntax: man command

Here, command → find, grep, -----

## Process ID :

For every process running in Linux there will be ID allocated. which will be used to keep Track of the process running. when process started, the process ID will be allocated and when process is compleated, the ID will be closed. Allocating of process ID is own by operating system and it cannot be control by us.

30. How to check Process ID?

A - command : ~/batch42$ ps -ef

output:

| - | Process ID | - | - | - | - | - | - |
|---|---|---|---|---|---|---|---|
| 1000 | 1753 | 1292 | 0 | 08:50 | ? | - | - |
| 1000 | 1753 | 1292 | 0 | 08:48 | ? | 00:00:00 | - - |

- command :

~/batch42$ ps -ef | grep terminal

↓

Pipe operator

output :

| - | ProcessID | - | - | - | - | - | |
|---|---|---|---|---|---|---|---|
| 1000 | 1508 | 1 | 0 | 08:48 | ? | - - - - | terminal |
| 1000 | 1988 | 1553 | 0 | 09:19 | pts/0 | - - - - | terminal |

31. How to Kill the process ID?

A.

command :

~/batch42$ Kill -9 PID

↓

what process ID we have kill, here give it.

Ex : Kill -9 1508

↓

Here, 9 → terminate signal.

## Port Number:

- Port Number acts as a physical address of any tool running on Linux.
- To connect to the given tool we prefer the PORT Number.
- Assigning PORT Number is controlled by us & we can assign upto 65,000 Ports, as per the industry standards we prefer to use 4 Number combination.
- On a given server the same PORT Number cannot be used twice. If we try to use we get an exception called "PORT bind exception" & PORT is already in used.
- Before allocating any PORT Number it is always prefered to check weather the PORT Number is used by someone else.

32. How to check the PORT availability?

A. syntax: ~/batch42$ netstat

output: don't used these output PORT Numbers.

Ex: ~/batch42$ netstat | grep 7000

33. How to findout the PORT Number & ProcessID combination?

How to findout the (or) ProcessID allocated for PORTNumber?

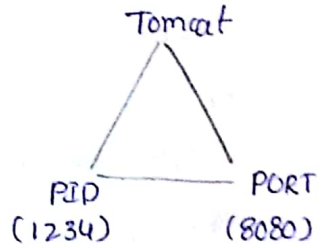
A

commands:

(i) netstat -ntlp | grep 8080

(ii) net-stat -alp | grep 8080.

ProcessID Number




* In general Redhat linux we use "ntlp".

34. How to Create a link in Linux ?

A

Syntax for Soft link:

ln -s target location Linkname

Syntax for Hard Link:

ln target location Linkname

Ex:

:~/batch42$ ln -s /home/administator/batch41/catexample Mylink

O/p:

~/batch42 $ ls -l

Then

:

lrwxrwxrwx 1 administator administator 38 2018-02-15 09:47

:

mylink → /home/adm/batch41/catexp

35. How to Create a Tar file ?

A

TAR is using only Grouping in Linux

ZIP is using Grouping & compressing in windows

ZIP, Tar, Jar → files not folders.

Syntax:

tar -cf tarname.tar tarname

Ex:

tar -cf backup.tar backup

O/p: ls -l

.

---------- backup

---------- backup.tar

36. How to extract Tar files ?

A

syntax:

tar -xf tarname.tar

Ex:

tar -xf backup.tar

O/p: ls -l

---------- backup

---------- backup.tar

37. How to view the content of tar without untaring?

A Syntax:

tar -tf backup.tar (or) tar -xvf backup.tar

-tf → displays content

-xvf → content not displays & print

Here, v → verbose

O/p:

backup/

backup/backfile1

backup/backfile3

backup/backfile2

38. How to update a tar?

A Syntax:

tar -uf backup.tar cat example111

O/p:

backup/

backup/backfile1

backup/backfile3

backup/backfile2

Help: man tar

39. How to check the server start time?

A To know the server restarted last time

command: uptime

~/batch42$ uptime

O/p:

10:23:06 up 1:44 1user loadaverage: 0.00, 0.001, 0.05

0.00 → before 1min

0.001 → before 5mins

0.05 → before 15mins

40. How to check the load average of server?

(or)

A command: uptime

O/p:

10:23:06 up 1:44 1user load average: 0.00, 0.00, 0.05

16-Feb-2018

41. How to check the cpu utilization?

(or)

How to check the processwise cpu utilization?

(or)

How to check the processwise memory utilization?

A.

command: top

Ex: ~/batch42$ top

o/p:

In o/p

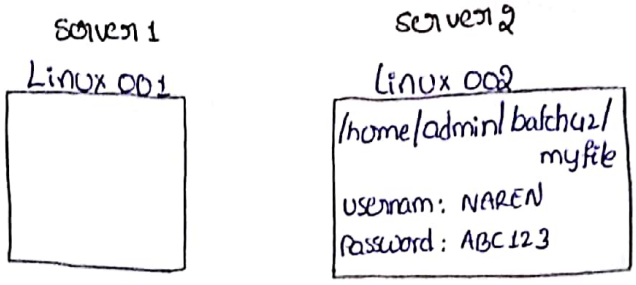
when we get, It is Ideal Time is high & wt is waiting Time is low that is cpu is good cpu

* control+d → cat
* Q (or) control+c → quit

42. How to copy file from one server to another server?

A. Linux Provides 2 commands to copy file from one server to another server.

1. Sftp (manual approach)
2. scp (automation approach)




1. Sftp:

(i) Login to Linux 001 (/home/adminstator)

(ii) sftp NAREN@Linux002 ← Login

Password: ABC123

(iii) cd batch42 ↵

(/home/NAREN)

(iv) get myfile — get & put are commands in Linux.

(v) Exit

2. SCP:

(i) Login to Linux001 (/home/adminstator)

(ii) scp NAREN@linux002 : /home/NAREN/batch42/myfile ↵

Password: ABC123

Then directly copy the file

(iii) exit.

Sftp is more used in the manual approach of copying file from one server to another server in the case of scp is mostly used for automation, where we donot have option of typing manual commands on the Target server.

# EDITORS

## Editors in Linux:

We have 3 Editors in Linux

1. Nano — (means plain one like Notepad)
2. Gedit — (Graphical Edit)
3. vi — (view & insert)

**1. Nano:** It is one of the basic editor helps to create a file, view the content, modify the content, etc.,

Syntax: nano filename

Ex: ~/batch42$ nano myfile

If we open any of the file in editor, if the file is file is already available it will open it. If the file is not available it will create a new one.

**2. Gedit:** It comes as a graphical editor & it provides similar functionality as Nano. (In real time we can't through putty. so don't have gedit).

Syntax: gedit filename

Ex: ~/batch42$ gedit geditfile

**3. vi:**

(i) vi is the one of the powerful editor which is use to apply different operations on the content of files.

(ii) vi comes with 2 modes.

I. View (V) mode: Viewmode which helps to view the content & also apply commands.

II. Insert (I) mode: where as Insert mode will help us to type content in editor (or) file.

When we issue 'vi' command by default we land into viewmode & use (type) 'i' to move from view to Insert mode. use "Escape" to come from edit to viewmode & to exit of editor we use ":wq!".
(Insert)

Syntax: vi filename

v → i - type 'i'
i → v - 'Escape' press

Ex: ~/batch12$ vi vifile

## Vi commands:

w - save

q - exit

x - delete character (incase 2,3,5,6,.... type 2x, 3x, 5x, ....).

dd - to delete line (if multiple lines ex: 5 lines → 5dd).

If 10th line to delete then type ':' (colon) cursor will goes to end line then there type 10, then cursor will goes to 10th line.

: line number - jump to typed line number.

:u - undo

/ - for searching

Ex: we have to search 'asdf'. then type '/' & 'asdf'. then the cursor goes to 'asdf' line.

:%s/old word/replaced new word/g - To replace the old word to new word

Here g, global occurance (all places)

Ex: :%s/XSXSXSXS/AAAAA/g
(XSXSXSXS → old, AAAAA → new)

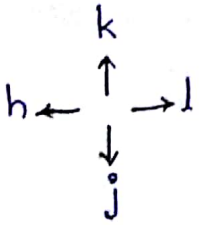
lower case 'o' - Insert new line a bottom (or) down line

upper case 'O' - Insert new line a TOP (or) upper line

upper case 'J' - merging the line (Shift + J)

'G' - starting of line

'g' - top line



## Merging:

Linux provides an option of merging two commands, the first command output will be passed as input to the second command.

The final output is as the final command execution output.

Syntax:

Ex:

```
ps -ef | grep 1234
```

For merging the two commands we use "|" pipe operator.

# SHELL SCRIPTING

## Shell scripting - Advantages:

(i) Using shell scripting we will be able to "Group Multiple Commands" and execute them, to get output of every command.

(ii) We can schedule the shell script to "run automatically" on given intervels.

(iii) We can pass "dynamic values" to the script as command line arguments so that, the script behaviour will be changed based on values passed.

(iv) We can apply "Filter conditions" & when criteria matches a set of commands will be executed & if criteria does not match another set of commands will be executed.

## Steps to write shell script:

Step-1 : create a new file with extension ".sh". Here .sh → standard notation

Step-2 : open file with vi Editor [(or) any Editor].

Step-3 : write content.

Step-4 : Save file.

Step-5 : change permissions

Step-6 : Run the script.

## Shell script in commands:

Step-1(& 2): vi myscript.sh

Step-2& 3: content

Step-4 : :wq!

Step-5 : chmod 777 myscript.sh

Step-6 : ./myscript.sh

1. Write a shell script to display 'Hello world' message.

(i) vi mymsg.sh

(ii) echo "Hello world"

(iii) :wq!

(iv) chmod 777 mymsg.sh

(v) ./mymsg.sh.

2. Write a shell script to display the current date & Time of a server.

(i) vi datetime.sh

(ii) date

(iii) :wq!

(iv) chmod 777 datetime.sh

(v) ./datetime.sh

3. Write a shell script to display current memory usage.

(i) vi cmem.sh

(ii) free -g (or) free -m (or) free

(iii) :wq!

(iv) chmod 777 cmem.sh

(v) ./cmem.sh

4. what is the First line of shell script?

A.

#!/bin/sh

This line is called as "shebang".

we have 3 shells in Linux

(i) sh — shell
(ii) bsh — bourn shell
(iii) ksh — Kron shell

V.Imp
*Note:

"Redhat" Linux by default uses "bronshell" (bsh).

5. How to include comments in the script?

A.

Comments are informational messages helps to provide information about the script.

Ex:

```
#!/bin/sh
# write by NAREN on 17th Feb 2018.
# This script is used to find ERROR in Log files.
```

6. How to declare variables in script?

A.

In script we use '$' to 'read'.

Here 'n' is the variable. Then

echo $n → is command

The value of 'n' variable will be printed.

'hello'.

variables { count = 0, n = 10, Path = /home/admin/batch42/Tomcat, n = hello }

Ex:

name = hostname  if hostname = linux001
↓ variable  ↓ variable.

echo $name

O/P: linux001

7. write a shell script to find the number of occurances of ERROR in tomcat.log store the output into variable & display the value of variable.

A.

```
#! /bin/sh
# write by Gangga on 17th Feb2018
# This script is used to find the number of occurances of error in tomcat.log

count = 'grep -iwc "ERROR" Tomcat.log'  (otherwise give complete path)
echo $count
```

8. How to Pass command line arguments?


A.

```
n = $1
echo $n
```

o/p:

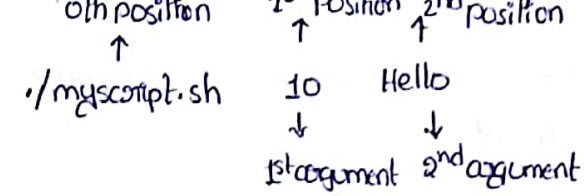
'10'

(or)

```
n = $2
echo $n
```

o/p:

'hello'




we pass numbers of arguments. we read the arguments by their positions.

Here, $1 → 10

$2 → hello

If $0 → then we will get script name

$0 → ./myscript.sh

9. How to read 10th command line argument?

A.

```
${10}
```

Here grouping({ }) is mandatory for a double digit numbers (or) values when we pass the argument.

In Linux Shell scripting,

$# - for counting number of arguments

$* - It will read all arguments

$$ - Displays the process id ID of script's

$? - Get the previous executed command status.

$! - Gives the background running process details

## Filter conditions:

In shell script we apply Filter conditions using "If-else" statement.

syntax:

(start) ← IF [condition] ; then

---

---

else

---

---

(End) ← FI

n=10

n = 10

↑ ↑

space

In Linux notations

> → -gt

< → -lt

>= → -ge

<= → -le

= → -e

=! → -ne

Ex: condition

```
IF [$# -gt 5] ; then
  echo "more than 5"
else
  echo "less than 5"
FI
```

Example - script:

```
#!/bin/sh
# Program to check error message in log
# written by GANGA on 19th Feb.
  count = 'grep -ic "ERRORS" tomcat.log'
  IF [ $count -gt 5]
      echo " more than 5 Errors"
  else
      echo " less than 5 Errors"
  FI
```

Example - script:

```
#!/bin/sh
# program to check error message in log
# written by GANGA on 19 th Feb
  count = 'grep -iwc "ERRORS" tomcat.log'
  IF [$count -gt 0]
    echo "There are $count errors in log"
  else
    echo "There are no errors in log"
  FI
```

10. How to Send Emails from Linux ?

A. In Linux direct command to send Email.

Command:

mailx -s "subject" "body" emailaddress.

Ex:

mailx -s "ERROR in Log" "There are $coont errors" ganga@gmail.

Here, s → send.

script:

```
#!/bin/sh
# program to check error messages in log
# written by GANGA on 19th Feb

count = 'grep -iwc "ERROR" tomcat.log'
IF [$count -gt 0]
   mailx -s "ERRORS" "There are $count errors in log" ganga@gmail.com
else
   mail x -s "ERRORS" "NO ERRORS" ganga@gmail.com
FI
```

## Automation - Process:

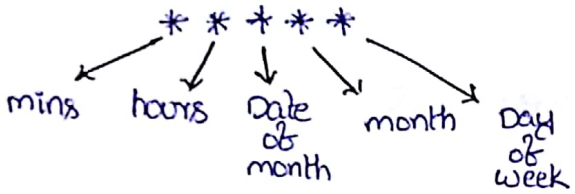
In Linux, to automate the script we have feature called "crontab".

For scheduling the script we use crontab.

Ex:

| Time | script |
|---|---|
| 5:00AM | /home/admin/batch42/ERROR.sh |
| 7:00PM | /home/admin/batch42/filecopy.sh |
| 3:00AM | echo "good morning" |

Sytax:




In crontab

* → every minute
* → every hour
* → every date of month
* → every month
* → every Day of week.

*It follows 24 hours format.

| Cron Expression | script |
|---|---|
| * * * * * | |

Ex:

For everyday 5:00AM — Cron Expression: 0 5 * * *

* * * * *

Ex:

March 5:00AM — 0 5 * 3 *

Ex:

March 5:00 AM Every week day — 0 5 * 3 1,2,3,4,5 (or) 1-5

Ex:

My Criteria In June 15th minute of 7PM June 15 weekday — 15 19 15 6 1-5

min hour day month weekdays

11. How to List (or) view crontab?

A. -login to linux.

command:

$ crontab -l

o/p:

# 0 5 * * * /home/administator

12. How to Edit crontab?

A. command:

$ crontab -e

automatically opened 'nano' edit.

GNU nano 2.2.6 File /tmp/crontab.IAY0159/Gontab.

Here,

# 0 5 * 1 * → cron Job

# * * * * * echo "hello" → cron Job

adding a new cron expression.

# * * * * * echo "Good Morning" then press control+S

13. How to Run shell script in the background?

A. command:

$ nohup ./script.sh &

(or)

$ nohup scriptname &

(or)

$ nohup scriptname command line arguments &

Note: '&' always last.

# UNIX

## Introduction:

(i) Unix is a multitasking, multiuser computer O.S.

(ii) Unix was originally developed in 1969 by the group.

## UNIX commands:

(1) ls command: ls <option>

'ls' command 'List all' the files & directories available in current directory.

(2) ls -l: list all files & directories from current directory in long format.

(3) ls -a: Display all files including hidden fiels current & parent directory.

(4) ls -A: Display all files including hidden files from current directory.

(5) ls [first letter(s) of file(s)]*:

Display all files/directories whose name starts with any letter in pattern.

(6) ls [!_]*: Don't display all files / directories whose name starts with any letter in pattern.

(7) ls -l -a (or) ls -la: using multiple options at a time.

(8) touch: touch <filename>

touch command creates zero byte files.

(9) cat: cat <filename>

cat command displays the content of a file.

cat <filename1> . <filename2> <----

cat command displays the content of multiple files.

cat:

cat > <filename> - opens as existing file in override mode.

cat >> <filename> - opens as existing file in appending mode.

(10) cp :(copy): cp <filename> <newfilename>

cp command copy the data from one file to other.

(11) mv (move): mv : mv <filename> <newfilename>

- mv command move data from one file to other, old file not exists.
- Rename a old file to new file.

(12) rm (remove): rm <option> <filename>

- rm command remove files from current directory.
- rm command can't remove directories.

(13) wc (word count): wc <option> <filename>

- wc command displays no. of newlines, words & no. of bytes from file.

(14) wc -l : wc -l <filename>

- Display no. of lines in a file.

(15) sort : sort <filename>

- sort command will sort the data in file based on first column.

(16) uniq : uniq <filename>

- Uniq command remove the duplicates that are sequentially present in a file.

(17) Pwd : Pwd

- Pwd (present working Directory) command displays current directory.

(18) chmod :

- chmod (change mode) command changes the permissions of a File.

order of permissions in unix:

- owner permissions: Actions the owner of the file (or) perform on the file.

Group permissions : A member of the group that a file belongs to can perform on the file.

other permissions : what action all over users can perform on the file.

Permission values:

1 - Execute 2 - write 4 - Read 4+2+1 = 7

(19) cut: cut <option> <filename>

- cut command to extract portion of text from a file by selecting columns.

(20) Grep: [Globally search Regular Expression & print]

grep <pattern> <filename>

- grep command search for a pattern in a file & print.

(21). Head: head <option> <filename>

- Head command displays top 10 lines of file.

(22). tail: tail <option> <filename>

- Tail command displays bottom 10 lines of file.

(23) head & tail: head <option> <filename> / tail <option>

(24) mkdir: (make directory)

- mkdir command is used to create directories.

(25) rmdir: (rmdir - remove directory)

rm dir <directory name>

- rm dir command is used to remove directories.

(26) cd: (change directory)

cd <directory name>

- cd command is used to move to other directories.

(27) du: (disk usage)

- du command gives usage details. (bytes by default).

(28) du -h: (disk usage human Readable format).

- du -h command gives "disk usage details in human understandable" format.

(29) df: (disk free)

- df command gives space availability on disk.

(30) df -h:

- df -h command gives space availability on disk in human standable format.

(31) ps: (Process)

- ps command gives all processes details that are currently running.

(32) ps-ef:

- ps-ef command gives all process details that are currently running along with username.

(33) kill: kill <option> <process id>

- Kill command kills the process running with given id.

(34) Date: date command displays current date.

(35) sed: (Steam editor) - sed <option> <filename>

- sed command is used to modify data in files automatically.

(36) nohup: (no hang up) -nohup <options> &

- nohup is used to run the process even after log out from a shell.

(37) who am i:

- 'who am i' gives the details of current user who is logged into the unix machine.

(38) man: (manual) man <command name>

- man command displays the manual for given command.

(39) find: find <location to search> <type to search> <value>

- find command displays the location of given file.

Questions:

① How to find current running process in unix?

A. $ ps-ef

② How to kill a particular process?

A. $kill.pid

③ What is the command to display first 3 lines of the file?

A. $head -4

④ How to find Remaining diskspace?

A. $df -kl  df - disk free.

⑤ How to change the password? (Password for world)

A. $ passwd

⑥ How to know the version of unix?

A $ uname -a

⑦ Pwd - present working Directory

⑧ How to display all the background job running?

A. (i) Press ctrl+z - stop current job } only for jobs.
(ii) bg

⑨ How to get a job from background to a foreground?

A. (i) $jobs
(ii) $fg %1

⑩. How to know the no. of records in a Flat file?

A. $wc -l → no of lines

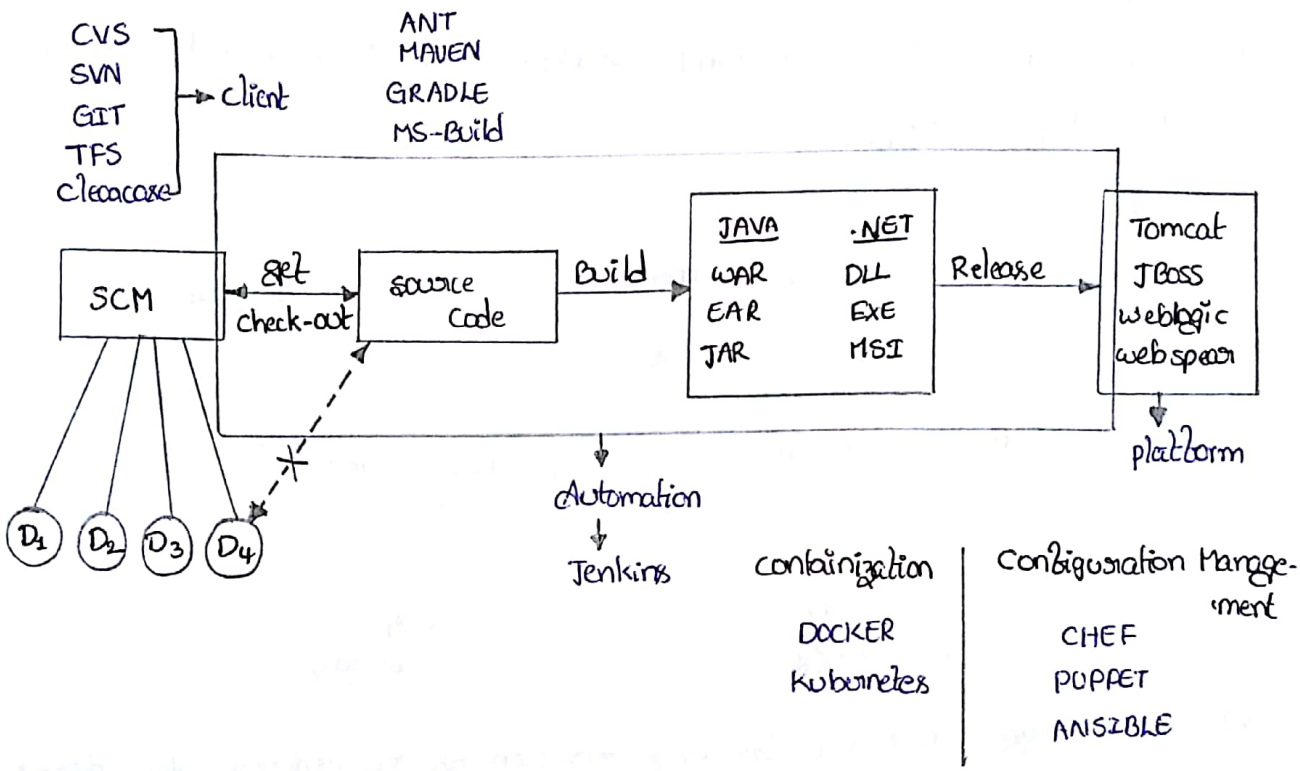
$wc - c → no. of chars

⑪ What is the diffence between cp (copy) & mv (move)?

A. Both commands will be used to copy from one location to another location. But with 'cp' file will remain in same location. where as in 'mv' file will be moved to the destination.

20&21- Feb-2018

# SCM
# SOURCE CODE MANAGEMENT




SCM:

If we manage the source code in a common server without using SCM. We may have the following limitations (or) disavantages.

(i) Every time the file is modifying the previous copy of the file will be overrided.

(ii) Because the previous files are not available we may not be able to go back to the previous copy of files.

(iii) To check the files are modified by which owner (user, author) which date purpose of the changes.

(iv) It may not be possible to have the concurrent development of projects.

To over come all above limitations we may perfer to use SCM Tools.

## Advantages with the SCM Tools:

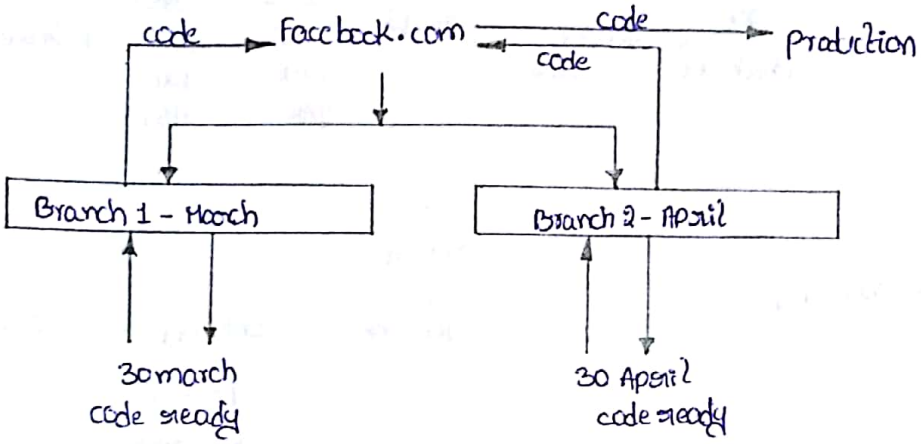
(i) SCM Tools does not know how to overtick the changes when files are modified.

(ii) Every time the file is modified a new version (or) revision will be created.

(iii) We can always go back to any revision to Roll-back the changes as required.

(iv) SCM supports the con-current development of the projects by following Branching strategy.

Ex:




(v) SCM manages metadata for every revision as how modified, why modified & when modified.

Some of the available SCM Tools are

(1) CVS
(2) GIT
(3) SVN
(4) Clearcase
(5) TFS

22-Feb-2018

## SCM Architecture:

SCM has 2 components.

(1) Server
(2) Client

(i) Server is centrally placed in organization where source code is being managed.

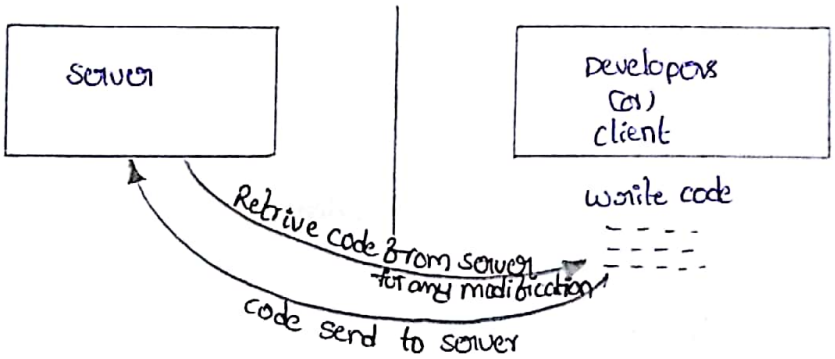
(ii) client is installed in the local system & we can connect to the server using following details like

- Server URL
- User name
- Password.

After join in any organization for the server URL, we contact to the admins (or) team members.

## Sever-client Assumption:

In Real time server was in somewhere & client was in somewhere.




In our laptop for practicing purpose assume C-drive is server & D-drive is a developer (or) client.

Laptop
↓
C-Drive
↓
Create folder
SVN server

leave this folder

Tortoise SVN Installation:

Now goto google page

↓

Download SVN for windows

↓

Download - Tortoise SVN

↓

For 64-bit OS

↓

Tortoise SVN 1.9.7-64-bit

↓

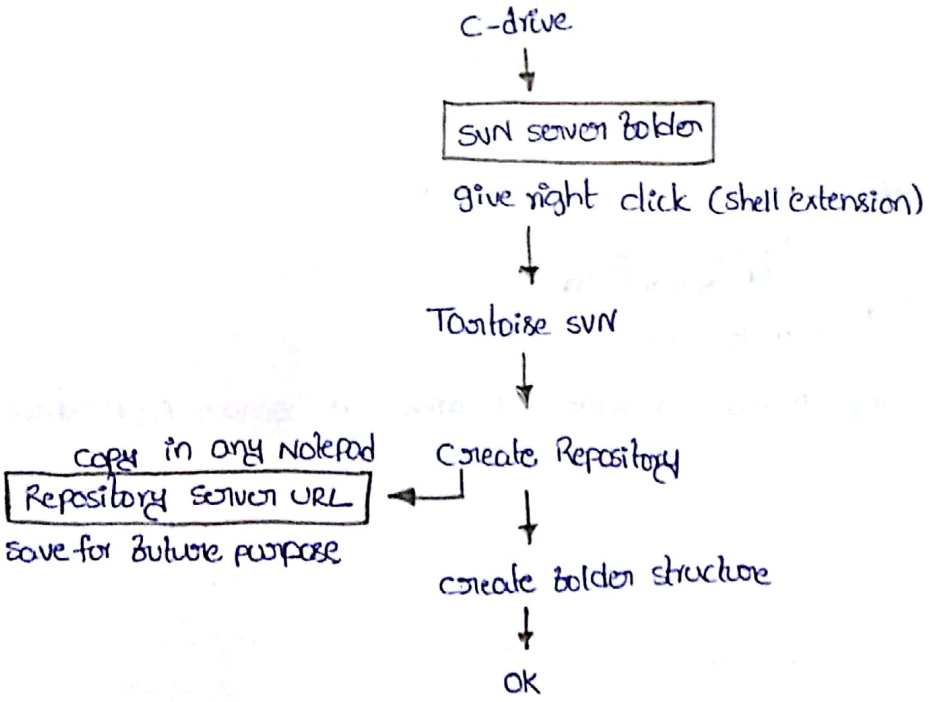
Download & Install

Shell extension:

Shell extension means, right (2) click of mouse gives one small prompt, that is the "shell extension".

Server:

After installion of Tortoise SVN, then goto c-drive. In that we already created folder "SVN server folder."




Now, SVN server is created & Ready to use.

client:

D - Drive

↓

Create Normal Folder (Developer)

↓

Create Text Document → write some Text & Save & Rename

Create html file → write some text & save & save as with in " " Then only it is html file

Then come out of developer folder.

Very first (or right) time we write new code. Then

Developer folder

↓

right click on this

↓

Tortoise SVN → Import

↓

URL of Repository

Import message

Recent messages

This is my first SVN import

* 'Import' is the only First time, then it is called as "commit."

ok

↓

Import Finished

Action command

Path Import-team

ok

Import: Very first time when we are adding source code to the SVN "SVN server" for a given project is called as "Import".

Steps:

(i) Go to the "source code" location.

(ii) Select Import from SVN options & just provide "SVNURL". "Enter comment message" & say ok

When we perform the above activity the source code will be added to the SVN repository.

Q. What is (Re) SVN Repository?

A. It is a place (or) location where the source code is being managed. It is also called as "source code Repository."

Q. What is Repo-Browser?

A. Repo-Browser is the option, to open the Repository & view the source code available. This option is only for viewing & we cannot modify any thing.

Goto server location

C:/ SVN server

↓

Right click

↓

Tortoise SVN → Repo-Browser

* In Real time we don't have this Repo-Browser for view of source code.

Check-out:

The process of getting the source code from SVN server is called it as "checkout".

When developer want to make changes on existing code, we will checkout the code from SVN server & make changes.

Steps:

(i) Go to the location where you want to check-out the code.

(ii) Select SVN checkout option.

(iii) Provide SVN URL, Target locations.

(iv) Save the changes.

Goto D-drive

↓

create New folder "checkout"

↓

give right click on checkout folder

↓

SVN checkout

↓

click "ok".

Adding New file to exsisting source code:

(i) First check out the source code.

(ii) Add a files to the source code

(iii) From the file select "Add" option.

(iv) From the file select "commit" option.

Go to Developer (D:/)

↓

checkout Folder

↓

create new folder → create new text doc & save

↓

give right click

↓

Adding & commit

* Any modifications do in checkout location only.

How to modify an Existing file:

(i) check out the source code.

(ii) open file to be modified

(iii) make the changes

(iv) Then select "commit" option from SVN

23-Feb-2018

Q. How to check the Meta data of a file?

A. (i) Go to the file for which we want to check the metadata. Select "show log option".

(ii) Show log displays the output as every revision / version of a file.

D-drive
↓
checkout folder
↓
modify folder
↓
give right click
↓
Tortoise SVN → show log message & watch it.

Q. How to revert changes?

A.
(i) Go to the file for which we want to revert & select revert option.

(ii) When we revert the latest copy from the server will be copied to the local

(iii) And our changes will be reverted.

D-drive
↓
checkout folder
↓
modify any file
↓
give right click
↓
Tortoise SVN → Revert.

TRUNK - like original copy (consider it as master copy)

BRANCH - xerox copy.

TAG - Now not uses this Tag.

## Differences Between TRUNK, BRANCH, TAG:

### TRUNK:

(i) TRUNK is considered as a master code. Depends on how many projects we support, we may have that many TRUNKS available.

(ii) TRUNK access will not be given to the developer as to the direct changes to the TRUNK is not supported.

### BRANCH:

(i) For every sprint initiated there will be a new branch created.

(ii) Developers will checkout from branch, make changes & check in back to the branch.

(iii) From Trunk we can create any number of branches as required.

### TAG:

(i) We can use tag in 2 different purposes.

- We can set the bench mark of code which will be used to roll-back to specific tagging (or) snapeshot.
- We can also use for locking purpose, where, the code commits will not be allowed but checkout can be done.

07-Feb-2018

# BRANCHING STRATEGY

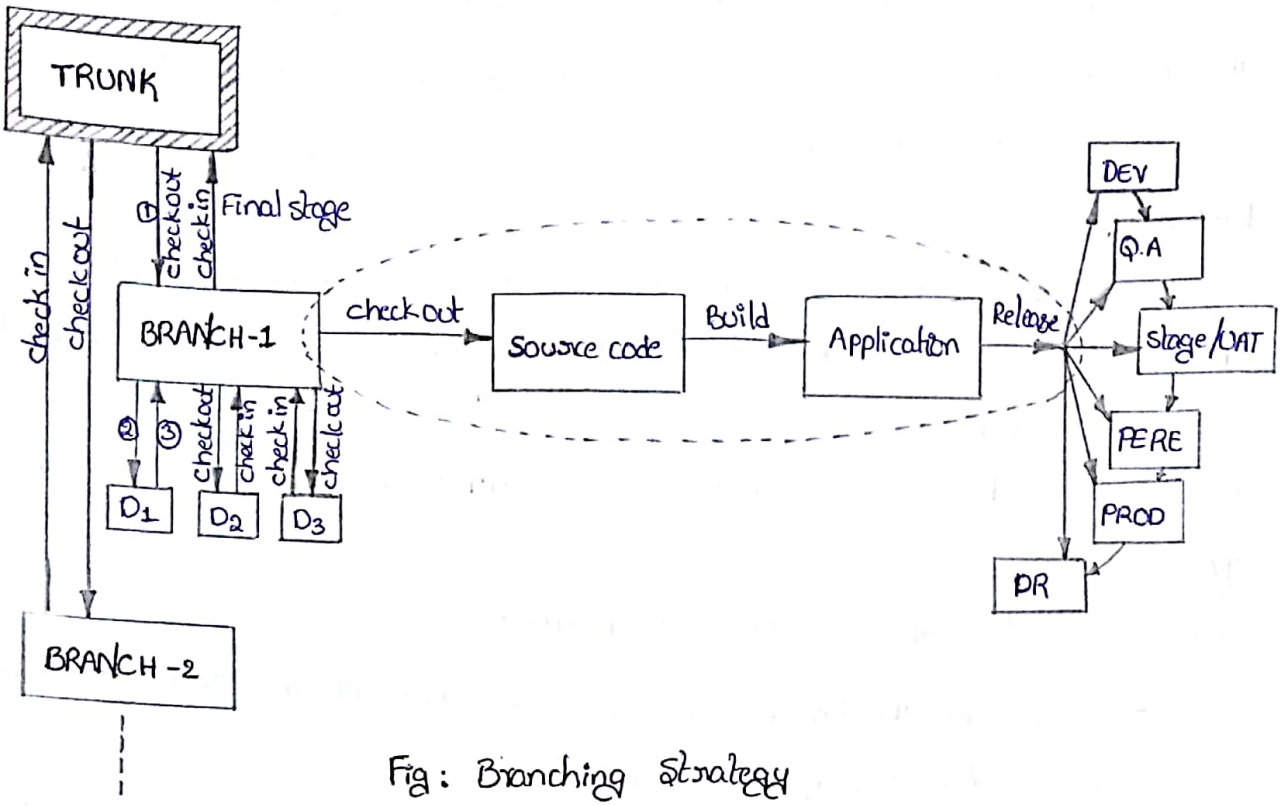



Fig: Branching Strategy

## Branching Strategy:

(i) Branching strategy is mainly used to support sprint in Agile Process.

(ii) When sprint is initiated there will be a Branch created & provide the details through development team.

(iii) Developers check out the code from Branch, make the changes & check-in back to the Branch.

(iv) The Build will be taken from Branch & generate the application.

(v) Application will be released to the different environments to perform different levels of Testing.

(vi) Once Testing is done, the application will be released to the production forward by OR.

(vii) After Releasing changes to the production we will observer for a week time.

(viii) If no errors happend with a week, we consider the code is stabilized. & plan plan to merge changes from Branch to Trunk.

28-Feb-2018

# MERGING STRATEGY

1. Stable Merge
2. Blind Merge
3. Manual Merge

## 1. Stable Merge:

After observing the code in production & decide the code is stable we plan to merge code from Branch to Trunk, which is called as "Stable Merge."

## 2. Blind Merge:

When Developer made changes in two many files manually comparing the difference may not be possible. In this senario we may treat the code is stable & we merge to Trunk is called as "Blind Merge."

## 3. Manual Merge:

When developer made changes in very few files we may verify every file & make sure all changes are correct, when we commit file by file is called as Manual Merge.

Based on requirement the merging will happend from Branch to Branch (or) Branch to Trunk.

SCM terminology it is Merging → Devops terminology CI continuous Integration.

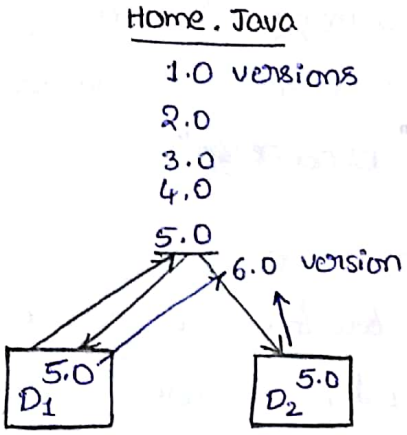
## Conflicts:

When we checkout the source code, we may checkout on a specific version & before we check-in back the changes there may be other developers might have check in the changes for which new version is created. In this scenario, SCM shows as "conflict issue".

To resolve this, before we commit any changes, checkout the latest code, make changes on top of it & commit back the changes.

In regular cases SCM always recommand to checkout the code before making any changes & check-in back.

Ex:




$D_1$ make changes & check-in. it is now 6.0 version

$D_2$ check-out 5.0 version first. But now it is 6.0 version. Then, check out by 'SVN update' to over come the conflict.

# Hooks

Using Hooks we will be able to perform some of the Filters (or) Actions on source code management.

There are 2 types of Hooks available

1. Pre-commit Hooks.
2. Post-commit Hooks.

## 1. Pre-commit Hooks:

Using Pre-commit Hooks we will be able to apply filters before commit happends.

Ex:

Every commit should have a commit message.

Every commit message should have a user ID.

Every commit should have project code (or) change number.

## 2. Post-commit Hooks:

Using post commit Hooks we will be able to apply few actions after commit happends.

Ex:

Sending an email.

Invoking other tool (or) process.

02-March-2018

# BUILD

When developers write a Java code it is a human code with the extension of ".Java".

".Java" files will not be understandable by the system to make it understand, we may need to convert the Java code into byte code which is with the extension of ".Class".

Java code → .Java

Byte code → .class

The process of converting Java files into class files is called as "compile."

Syntax to compile a Java file.

"Javac Home.Java"

where,

c → convert

Java code —convert/compile→ Byte code

Ex:

If dev. have 250 Java files

250 Java files —convert into class→ 250 .Java files + 250 .class files. ⇒ 500 files

we required only 250 .class files.

The final output will be provided by ".class file". After compiling all the Java files we may need to Group them together & bundle as a "package". which is called "Application."

The overall process of compiling & bundle together as package is called as "BUILD."

Software Lifecycle

- Requirement
- Analysis
- Design
- coding
- Testing
- Implimentation
- support

The overall process of compiling & bundle together as package is called as "Build".

If we perform above activities manually it may take longer time to compleate the process & there may be chance of commiting mistakes also.

To overcome this, there are few tools available to perform the Build. which are called as "Build Tools".

Some of the available Tools are.

(i) ANT from Apache
(ii) MAVEN from Apache
(iii) GRADLE from Apache
(iv) MS-Build from Microsoft.

MAVEN is a Freeware Build Tool. which can be downloaded from "maven.apache.org"

The Latest version of MAVEN is "3.5" (3.x)

# MAVEN

## Difference Between ANT & MAVEN:

| ANT | MAVEN |
|---|---|
| (i) ANT is a pretty much old Build Tool. which is exclusively developed for only Build. | (i) MAVEN is Advanced tool, which performs the Build. |
| | (ii) MAVEN is called as Project management tool. which is developed for performing different activities. |
| | (iii) Build is just one activity of MAVEN process. |
| (ii) ANT Runs on script based & for every Build implementation we may need to write the script based configuration. | (iv) MAVEN runs on XML & all its configuration files based on XML only. |
| (iii) ANT is a procedural | (v) MAVEN is a declarative. |

Q. How to setup MAVEN?

A. (i) MAVEN software is available on apache.maven.org and we can download the required version for required platform.

(ii) If it is a window we can download the Zip file & Extract to any location. where we want to keep the maven software

(iii) If it is Linux download the TAR file & Extract to any location.

Q. What is MAVEN Home?

A. It's a location, where we have the MAVEN software is placed is called as the "MAVEN Home".

Ex:

Windows - c:/maven_3.5.0

Linux - /opt/maven_3.5.0

# MAVEN Setup:

Google

↓

Download MAVEN for windows

(maven.apache.org)

↓

Binary .zip archiv apache-maven-3.5.2-bin.zip

↓

Zip file Extract it & keep it in c-drive

↓

Apache-maven-3.5.2 folder

↓

bin (& copy the path)

↓

Goto Run with cmd

↓

>cd c:\apache-maven-3.5.2\bin → Paste path.

>mvn ↵

Error : JAVA-Home should point to a JDK not coded JRE

JDK - can compile the program & it's convert into byte file (Complete set)

JRE - only Running environment.

Then download Java

(Now Jdk download)

Download JDK for windows.

↓

Java SE Development kit a Downloads.

↓

click on

Windows 375.56MB ↓ Jdk-9.04_windows-x64-bin.exe

↓

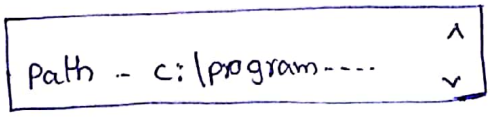
Install

↓

Goto path & Copy path.

## Setup of JAVA_HOME + Path:

This PC
(my computer)
↓
Properties
↓
Advanced settings
(in right top of prompt)
↓
Environment variables
↓
System variables
[New] ↵
↓
variable Name : [JAVA_HOME]
variable value : [C:\program....] (Paste path here)
↓
Paste path
system variables




[Edit]
↓
Edit system variable
variable name : [Path]
Variable value : [--------/bin] → bin path
Paste Path\bin.
↓
Then
close cmd old prompt
&
open new cmd prompt.
↓
>cd
copy bin path (copy maven path under bin)
>cd paste path
>mvn
No Errors :)

## Setup Maven_Home + Path & bin:

First copy c:\maven_3.5.2

Now again go tho

This PC
(my computer)
↓
Properties
↓
Advanced settings
↓
Environment settings variables
↓
system variables
↓
New
↓
MAVEN_HOME
↓
Paste Path
↓
Edit Path

Paste path

____/bin; c:\mvn_maven-3.5.2

Then
goto cmd
↓
>mvn

## MAVEN setup:

- To set up the MAVEN we use Jdk as a dependent software. Once Jdk software is downloaded. We may setup system variable to make sure JAVA is accessable from system.
- To have MAVEN access anywhere from the system, we may setup system variable for MAVEN is setup.

Q. How to test weather MAVEN is setup properly?

A. (i) Goto command prompt & type mvn command.

(ii) As an output we should get the output in a proper maven format, but it should not through any error.

## MAVEN Life cycle:

MAVEN has 3 type cycles methods (or) steps.

1. clean
2. Build
3. Deploy

### 1. clean:

Before any Build starts MAVEN (look for) check for any unwanted files available part of previous build. If any files it is going to clean them & start a new build.

### 2. Build:

During this step MAVEN will perform the actual build process & generate artifact. (Artifact is also called as application).

### 3. Deploy:

When Artifact is ready. The same will be copied to the repository is called as Deploy.

## Main configuration of MAVEN:

MAVEN has 2 main configuration of files.

1. Pom.xml (Project object Model) – { Parent Pom → outer
   child pom → inner
2. settings.xml

## settings.xml:

If any configuration details are common across all the projects can be defined in settings.xml.

settings.xml can be managed at 2 levels.

1. Global level
2. User level

If we manage setting.xml at global level it is applicable for every project for every user. where as if we apply at user level all the projects managed by that user will be impacted or applied.

### Global Level Path:

<maven_Home> / conf / settings.xml

### User level Path:

⇒ Windows

c: /users /username / settings.xml

⇒ Linux

/home/username/settings.xml

## Pom.xml:

Any project independent details will be placed into Pom.xml & for every project there will be individual pom.xml available.

For suppose if we are supporting 50 different projects there will be 50 different pom.xml available because every project has got pom.xml

Every project got pom.xml it will be managed along with the source code.

When we check-out source code both source code & pom.xml will be checked out.

syntax:

```
<Profiles>
    <profile>
      - - - -
      - - - -
      - - - -
    </profile>
</Profile>
```

## Dependencies:

- During development developers may depend on different JAR files which are also suppose to be available during build.
- During build process MAVEN understands, what are the dependencies are suppose to be downloaded & they will be downloaded.
- Because these dependencies are independent to each project it is preferred to define them in Pom.xml.

syntax to define dependencies:

```
<dependencies>
    <dependency>
        <name> Email.Jar </name>
        <id>        "       </id>
        <version> 1.0 </version>
        <scope> Runtime </scope>
    </dependency>
    <dependency>
        <name> Epoch.Jar </name>
        - - - - - -
        - - - - - -
        - - - - - -
    </dependency>
</dependencies>
```

## Transact (or) Transaction:

When we have application with in other application, both applications both applications with have own Pom. In this Relation, the parent Pom can have the child poms. depending on how many internal applications we have that may child poms available.

If we define any dependency in the parent pom we will be able to use the same in child Pom. such dependencies are called as "Transaction dependencies".

## Repositories:

This PC
(My computer)
↓
C:
↓
Users
↓
Username folder
↓
.m2
↓
 Repository
↓
This is local
Repository.

## Repositories:

Repository is a location where we manage all the dependencies.

MAVEN supports 3 types of Repositories.

1. Local Repository
2. Central Repository
3. Remote Repository.

### 1. Local Repository:

(i) Local Repository is available in the same system, where we perform the Build.

(ii) The Local repository is created by default by maven & the location is

⇒ windows

c:/→ users/→ username/→ .m2/→ Repository.

⇒ Linux

/Home/username/.m2/Repository.

### 2. central Repository:

(i) It is a common location managed with in organization.

(ii) when we have the central Repository managed the URL need to be define either in the pom (or) in the settings, depends on it is either common (or) project dependent.

(iii) There are few tools also available to use as central Repository like.

Nexus, Apache are have etc.,

### 3. Remote Repository:

(i) Remote Repository is a public websites we can be managed (or) used by any organization (or) project.

(ii) Even in the case of Remote repository we need to define the URL either in pom (or) settings.xml.

## Order of looking for Repositories:

(i) When build is initiated MAVEN look for local Repository for dependencies if it is not identified, it look for central Repository followed by Remote Repository.

(ii) very first time MAVEN download the dependencies & place them into local Repository.

(iii) From Next time it is use from Local Repository rather than going to central (or) Remote Repositories.

## MAVEN Target:

To support different activities maven provides different options. for different activities. which are called as Targets (or) goals.

Some of the common Targets are,

(i) clean
(ii) compile
(iii) Package
(iv) Test
(v) site
(vi) install
(vii) Deploy

} Build

Ex: Install
⇒ windows
mvn install
⇒ Linux
./mvn.sh install

(i) clean:

To clean the previous build files.

(ii) compile:

The class files will be generated by compiling Java files.

(iii) Package:

If class files are available it is going to bundle as package.

(iv) Test:

we can Run the Test cases.

(v) site:

Technical documentation will be created.

(vi) Install:

Includes all steps for a compleate Installation.

(vii) Deploy:

To deploy the artibat to remote (or) central Repository.

Syntax to Run:

⇒ Windows

```
>mvn install
> mvn clean
> mvn compile
```

⇒ Linux

```
./mvn.sh install
```

To know the version of Java:

```
Java --version.
```

It displays the version of Java.

Plug-ins:

(i) Plug-ins are used (or) to support different MAVEN Targets. For Example, to perform compile target, it is suppose to have the compile plug-in support.

(ii) By default MAVEN comes with different plug-ins & we can also add the custom plug-in in pom.xml.

Syntax:

```
<plug-ins>
    <plug-in>
        <name>------
        <version>----
        ---------
    </plug-in>
    <plug-in>
        ----------
        ----------
    </plug-in>
</plug-ins>
```

05-March-2018

## Profiles:

When we have multiple environments available we may have the different configurations for different environments. changing of those details are manually is going to be compilcated, because of which we depend on Roles where we can define each role of each environment in settings.xml.

Every profile is got an own profile id & profile id should not be duplicated.

### syntax:

```
<profiles>
    <profile>
        -----
    </profile>
    <profile>
        -----
        -----
        -----
    </profile>
</profiles>
```

## Types of Builds:

Using MAVEN we can perform 2 types of builds.

1. Snapshot Build — Non prod
2. Real Build. — prod

Snapshot build is ment for non-prod environment, where as Real Build is meant for production environment. When we defined version in pom.xml we can include "SNAPSHOT" keyword along with version tag in Pom.xml.

Q. what are the GAV Parameters?

A. In pom.xml there are 3 main tags are mandatory.

They are GAV

G — Group ID
A — Artifact ID
V — Version ID

↓
checkout folder
↓
Pom
↓
Pom tags → G,A,V.

## The Detailed steps of MAVEN Build process:

When MAVEN Build is triggered (or) initiated –

(i) It checks for Pom.xml to scan the projects.

(ii) MAVEN reads Pom.xml & understand all project related details.

(iii) MAVEN understand the dependency details & plug-in details to download.

(iv) MAVEN download all dependencies & plug-ins from Repository.

(v) MAVEN look for GroupID in pom.xml to understand the project structure.

(vi) MAVEN finds all existing Java files in project (means in source code).

(vii) MAVEN compile the Java files to the copy them to the Target folder.

(viii) MAVEN Group all the class files into target (files) folder.

(ix) MAVEN bundle all class files & required files & package them as application (WAR, JAR, EAR).

(x) MAVEN copy the generated application into local Repository.

(xi) During the above build process both test & site will be running & Generate reports into Target folder.

# JENKINS

- When we perform the Build & Release activities manually. we may take longer time to complete.
- using Manual approach we can only perform Build & Release activity for only one project.
- Performing multiple projects & for multiple projects may be a challenge using (different) manual approach.

To over come all above issues, we may depend on automation using different Tools like Jenkins.

## Advantages of using Jenkins:

(i) Using Jenkins we will be able to automate Build & Release activity for agiven project.

(ii) Using automation process we will be able to run multiple builds in parallel.

(iii) Using Jenkins we can scheduled different projects to run at different schedules automatically.

(iv) Using Jenkins we can run multiple Builds in seauencial bases.

(v) using Jenkins we can also setup the notifications (or) alerts in the event of either Build failure (or) success.

(vi) Jenkins is a Freeware software & it can be downloaded from official Jenkins website.

(vii) To run Jenkins we may need container support like Tomcat, J-Boss, weblogic etc,.

(viii) Jenkins can be downloaded as an Installer (or) Jenkins service like Jenkins.war.

## Pre-requisites for Jenkins:

(i) Java should be installed in the system.

(ii) The application server should be installed (If Jenkins downloaded as '.war')

If Jenkins downloaded as ".war".

Note: By default Jenkins runs on a default port called "8080".

The latest version of Jenkins is '2.10'. But the running version as "2.8".

## How to access Jenkins:

Open Browser & type URL as.

" http:// servername : portnumber / jenkins."

Ex:

http:// localhost : 8080 / jenkins.

## Practical:

Download Tomcat

In webapp copy the app

bin

startup

Jenkins 2.6

↓

Download Jenkins.war

↓

click on 2.5(or) 2.6

Run Jenkins  Installing after Jenkins

http://localhost:8080/jenkins/

↓

copy password in · jenkins file

↓

Administrative password

↓

Give username & Pwd

↓

Open Jenkins console

↓

Go to people.

Manage Jenkins

↓

second option

configure global security

☑ Enable remember me

steps:

Log-in to Jenkins console

↓

New Version

↓

Provide new name

↓

Select the view option

Source code checkout from svn → Manage Jenkins

↓

configure system

↓

subversion → select -1.8

↓

Apply & save.

New Item

↓

SVN checkout → Name

↓

free style project

↓

ok

scm → subversion - ok

↓

Repository URL → save & ok

SVN checkout store in → Jenkins → workspace

↓

open job

↓

configure

↓

Build Periodically

↓

Crown expression

* * * * *

## Security with Jenkins:

- To prevent the misuse of Jenkins & to control the resource access, Jenkins provides a feature called "security."
- If any user want to access any resource they should have a proper access with proper access levels. which is refered as authentication & authorization.
- Any users registered with Jenkins (or) create with the Jenkins will be stored in one of the options.
    (i) using in-built database called Jenkins own database.
    (ii) using any LDAP server.
    (iii) using any custom solution.
- The level of access (or) authorization will be controlled using different options.
    (i) Any user can do any thing (No log-in required)
    (ii) Logged-in user can do any thing.
    (iii) Different project access.
    (iv) Action based access.

Q. How to view Existing users in Jenkins?

A (i) Log-in to Jenkins console &
(ii) select "People".

This option helps us to view what are the different users available on Jenkins.

Q. How to manage users?

A. To create a new users, modify users & delete users Jenkins provides an option called "manage users."

Steps:
(i) Log-in to Jenkins
(ii) Manage Jenkins & manage users.

* Jenkins works in backend also.

Every time users are created in Jenkins, it manages all user details in the form of xml & the location is

⇒ Windows

C:/users/username folder/.jenkins/Jenkins user

⇒ Linux

/Home/username folder/username/.jenkins/jenkins user.

## Build History:

To view How many total builds happend in a given day out of which, how many of them failure & success can we using (see) Build history options.

Steps:

(i) Log-in to Jenkins console.

(ii) Build History.

## View:

- when we have huge number of projects available on Jenkins, by default we may have an option to "view all projects".
- Incase of we want to have an option of having only few projects access we can create view by adding our own projects.
- Every time we log-in to jenkins we can view our own projects by going to my view option.
- while creating new view it's provides 2 options.
  1. Gives us option to select what projects need to be added.
  2. By default all the new projects associated will be automatically added.

Steps:

(i) log in to jenkins console new view provide newname

(ii) select the view option.

Manage Jenkins → configure system → subversion → 1.8

project svn checkout → configure → Build periodically → cron expression (time).

09-March-2018.

# Manage Jenkins:

## 1. Prepare for shutdown:

(i) This option is used to shutdown Jenkins application alone without disturbing other application.

(ii) When Jenkins is shutdown no new builds will be initiated but it will continue with running builds & make sure they are compleated successfully.

## 2. Manage old Data:

(i) When we have different versions of files available in case of we don't want old data we can either move them (or) take back-up (or) delete them.

(ii) To perform the above activity Jenkins provide an option called "Manage old Data."

## 3. About Jenkins:

(i) This option gives as a information about Jenkins as version and all its internal components alogg with licencing information.

(ii) This is just for veewing information we cannot either create (or) delete.

## 4. Manage Nodes:

We will discuss later classes.

## 5. Script console:

(i) By default Jenkins follow "Groovy scripting".

(ii) In case of we want customize (or) develope any functionality. we can write a custom groovy scripting & execute them in Groovy script console.

(iii) this option is used when we go for customization only.

## 6. Jenkins CLI:

Jenkin-cli.jar → download

Goto path of CLI Jar & copy

then go to 'cmd' then paste & Run it.

## 6. Jenkins CLI:

Jenkins adminstration can be perform using 2 options.

1. using Jenkins console
2. using CLI

(i) To perform Jenkins administration using command line, Jenkins provides a set of commands to perform administration.

(ii) The Jenkins CLI in the manage Jenkins option provides the whole list of commands to perform administration.

(iii) To use CLI we may need to download Jenkins-cli.jar & run the same command line.

## 7. Load statistics:

(i) This option is used to check the health of Jenkins to make sure Jenkins is running in normal (or) it running with heavy load.

(ii) If Jenkins is running with heavy load, there may change of performance degrade.

(iii) To over come this we may change the Jenkins architecture to distributed architecture.

## 8. System log:

(i) Jenkins manages its own log file called "systemlog", which keeps track of every build information.

(ii) Incase of any build failure happend we can use this log as a source to find out the error message (or) reason of having build failure.

## 9. System Information:

(i) Jenkins may use different components (or) resources at times if it is not pointed to the right version & right path it may cause build failure.

(ii) To check weather Jenkins is pointing to the right version & path, we may use system information.

10. Manage plug-ins:

we will discuss later classes.

11. Reload configuration from Disk:

(i) If we make any changes directly into backend configuration usally jenkins load them all the updates.

(ii) In case if the Jenkins is taking them reload the configuration. we can force to reload the configuration, by choosing an option called "Reload configuration from Disk."

12. Global Tool configuration:

(i) If we want to use any tool across Jenkins we may need to configure Globle tool configuration.

(ii) In the Globle Tool configuration we may add the tool path as where it is located.

13. configure Global security:

we will discuss later classes

14. Configure system:

we will discuss later classes

10. Manage plug-ins:

If Jenkins want to interact with different components by default it may not be able to communicate rather it is suppose to have a plug-in support to provide that capability. we may need to have the plug-in uploaded. so that Jenkins gain that capability to communicate with different tools.

To perform different Operations with plug-in Jenkins provide 4 options.

1. Available 2. Installed 3. Updates 4. Advanced.

1. Available :

Which provides the list of all available plug-ins can be configured with Jenkins.

2. Installed :

The list of already installed plug-ins.

3. Updates :

If there are any updates available for already installed plug-ins.

4. Advanced :

Which helps us to upload plug-ins manually.

Usally Jenkins plug-ins are with extension of ".hpi" or ".jpi".

hpi → hutson plug-in
jpi → jenkins plug-in

Steps:

(i) login to jenkins console
(ii) Manage Jenkins
(iii) manage plug-ins.

## Jenkins Architecture :

Jenkins works into Architecture

1. stand alone Architecture (Master only)
2. Distributed Architecture (Master & slave)

Manage Jenkins → Manage nodes → New Node
↓
Name → Jenkins slave1

## Jenkins Architecture:

Jenkins works into 2 Architecture.

1. Standalone Architecture (Master only)
2. Distributed Architecture (Master & slave)

### 1. Standalone Architecture:

(i) In standalone Architecture there is only one instance available which performs the build for different projects.

(ii) When there are more no. of project to run at the same time the one Jenkins instance may not be able to handle the load eventually the performance will be degraded.

To over come this Jenkins prefered to run as distributed Architecture

### 2. Distributed Architecture:

(i) In Distributed Architecture we have Master - slave Architecture available.

(ii) When Jenkins is set up there will be a default instance called 'Master' & any Nodes created either are refered as 'slave'.

(iii) Based on number of projects and available configuration we may come up with number of slaves as per requirement.

### Steps to create Nodes:

(i) Log-in to Jenkins console

(ii) Manage Jenkins

(iii) Manage Node

(iv) New Node

(v) Provide Node Name, Number of executors, usage option, availability option & start up option.

Once slave is created we can open the slave & start as per "start up" option selected.

Q. How to view existing Node?

A. (i) Log-in to Jenkins console

(ii) Manage Jenkins & Manage Nodes.

20-March-2018

# CONFIGURATION MANAGEMENT

When configuration is perform manually it may take huge amount of time to install something (or) configure something. The delay may cause the project timelines impacted.

To over come this DevOps recommand to use automation Tools to perform an automation which are called as "configuration Manggement Tools".

Some(time) of the available configuration manggement Tools are.

1. CHEF
2. PUPPET
3. ANSIBLE

## Architecture of configuration Management Tools:

configuration Management Tools have main 3 components.

1. Workstation
2. server
3. client.

### 1. Workstation:

(i) To write CHEF configuration we may need to install "chefdk" in our system. which enables us to write chef configuration.

(ii) where we install "CHEFdk", we call that system as "workstation".

(iii) Depends on number of DevOps Engineers works in organization, we may have that many number of workstations available.

### 2. Server:

(i) server act as "central component". where all the configurations will be deciding.

(ii) CHEF clients will be able to communicate the server & fetch the required configurations.

(iii) For a given environment, server is only one component available

3. client:

(i) The End system which we wanted to manage is called as a "client".

(ii) Depends on how many servers we want to manage that many clients we have in the environment. (Here servers means physical servers not chef servers).

(iii) For example, if we want to install Tomcat on 50 servers, which means I have 50 clients available in the project.

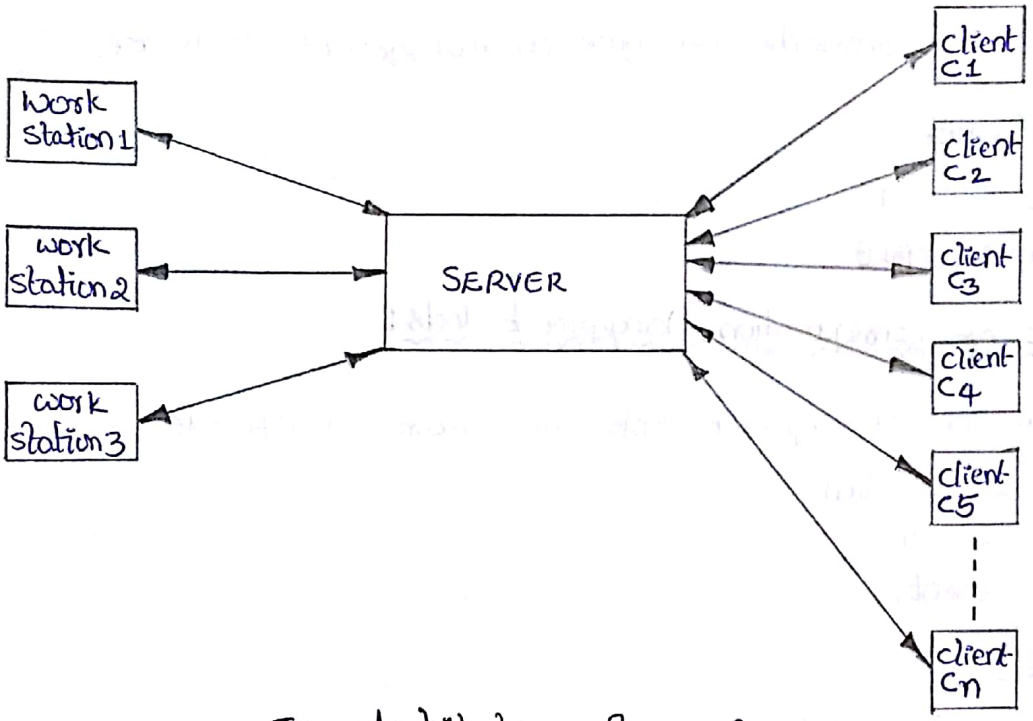



Fig: Architecture of configuration Mangement.

# CHEF

CHEF is designed & managed by CHEF INC (Incorporation) and website is chef.io.

## Advantages using CHEF:

(i) CHEF provides hosted CHEF server concept where the server is managed by CHEF itself we can just register & start using.

(ii) CHEF provides free licence managing upto 5 clients over & above we need to start paying.

(iii) CHEF has official tie-up with AWS which makes a good combination.

(iv) Most of the CHEF configurations are written as pre-defind & made them as available as public, where every one can start downloading & make use of it.

(v) most of the CHEF information can be identify on website called chef.io.

Some of the important URLS.

1. chef.io - Its a main website of CHEF.
2. learn.chef.io - All learning modules available.
3. manage.chef.io - Acts as a CHEF server.
4. supermarket.chef.io - (Acts as) where all CHEF configurations available

Q. Why do you go for CHEF?

A. To manage Resource.

## Important Terms:

1. chefdk
2. Workstation
3. Server
4. client
5. Resource
6. Recipe
7. cookbook
8. knife
9. Bootstrap

### chefdk:

(i) It's a software used for development (or) writing of configurations.

(ii) If we want to write configuration we (managed) we may need to have chefdk installed on the system.

(iii) The latest chefdk version is '2.5'.

(iv) The latest chef version is '13'.

### Resource:

(i) Any component we try to manage with chef is called as Resource.

(ii) Using CHEF we can either manage one (or) more than one resource.

(iii) Some of the examples of resources are file, folder, package, etc,.

### Recipe:

(i) Recipe is used to manage resources using one recipe we can either manage one resource (or) multiple resources.

(ii) usually recipes are with the extension of '.rb'. Ruby → .rb.

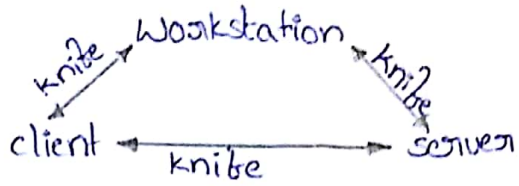
### cookbook:

(i) cookbook is a compleate structure of configuration. where recipes are existing inside cookbook.

(ii) cookbook has a standard folder structure & recipe is one of main folder, where all recipes are placed inside.

## Knife:

(i) Knife is an utility used to establish communication between workstation to server, workstation to client and server to client.




(ii) Knife is also used to upload (or) download cookbooks from server.

## Bootstrap:

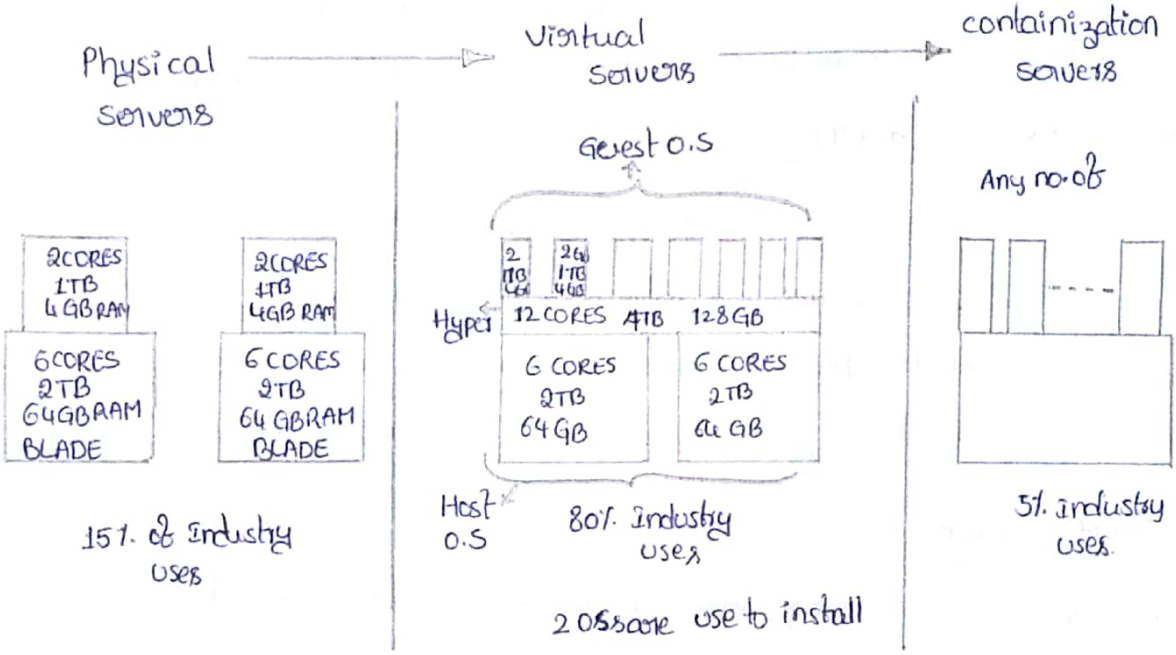
(i) The process of bringing up a physical server (or) virtual server as CHEF client is called as Bootstrap.

(ii) Bootstrap need to be done only once, from next time we can use directly without Bootstraping.

(iii) During Bootstrap process CHEF copy all the required files & permissions to client.

25-March-2018

# DOCKER CONTINIZATION




In physical sever architecture if we use the partial configuration. The remaining configuration will be wasted and some times it cannot be reused. To overcome this, the concept of virtualization is introduced. Where on a given physical server we will be able to configure multiple virtual servers. which is called "virtualization."

Even in virtualization we may not be able to use the configuration effectively to over come this containization is been introduced.

In the containization, we can have multiple containers will be running. But each container will occupies the configuration based on the real usage only.

## Advantages of using Docker as containization:

(i) The configuration will be used based on real usage.

(ii) We don't need to have 2 operating systems intalled, rather only one enough is used.

(iii) Starting of containers is very fast & performance effective.

(iv) Each container can have any no. of components included.

## Architecture of Docker:

Docker mainly contains 3 components.

1. Docker Build - where we create images
2. Docker Hub - where we places Docker images.
3. client - where we Run Docker Container.

### 1. Docker Image:

(i) All required components (or) softwares can be group together which will be formed as an image.

(ii) once Image is build it can be uploaded to the docker Hub.

### 2. Docker container:

(i) Using image we can create an instance which is called Docker container.

(ii) Using one image we can create any no. of containers.

(iii) For every container there is a processID and the port number will be allocated.

(iv) Using port number we will be able to access the required containers.

21-April-2018

# GIT

## Git:

Git is a 'version control system' for tracking changes in 'computer files' and coordinating work on those files among multiple people. It is primarily used for 'source code Management' in 'Software development', but it can be used to keep track of changes in any set of files. As a 'distributed revision control' system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

## Git Hub:

GitHub is a code hosting platform for version control & collaboration. It lets you and others work together on projects from anywhere.

## Advantages of Git:

(i) Free & open source

(ii) Fast & small

(iii) Implicit backup

(iv) Security

(v) No need of powerful Hardware

(vi) Easier Branching

## Local Repository:

Every VCS tool provides a private workplace as a working copy. Developers make changes in their private workplace and after commit, these changes become a part of the repository. Git takes it one step further by providing them a private copy of the whole repository. Users can perform many operations with this repository such as add files, remove file, rename file, move file, commit changes & many more.

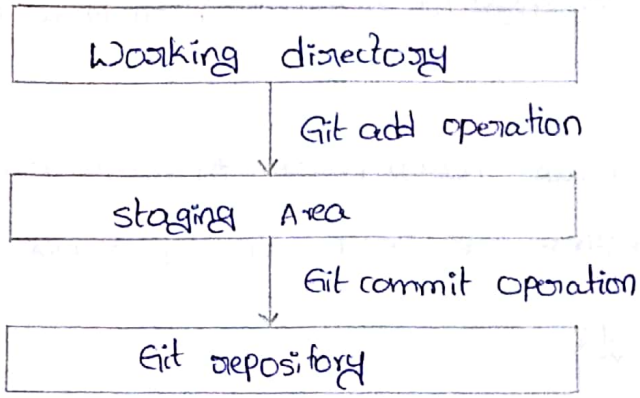
## Working Directory & Staging Area:

Let us see the basic workflow of Git.

Step 1: You modify a file from the working directory.

Step 2: You add these files to the staging area.

Step 3: You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.




Suppose you modified two files, namely "sort.c" & "search.c" and you want two different commits for each operation. You can add one file in the staging area & do commit. After the first commit, repeat the same procedure for another file.

- # First commit.

  $ git add sort.c

- adds file to the staging area

  $ git commit -m "Added sort operation"

- second commit.

  $ git add search.c

- adds file to the staging area

  $ git commit -m "Added search operation"

## Blobs:

Blob stands for "Binary Large object". Each version of a file is represented by Blob. A blob holds the file data but doesn't contain any metadata about the file. It is a binary file & in Git database, it is named as SHA1 hash of that file. In Git, files are not addressed by names. Everything is content-addressed.

## Trees:

Tree as an object, which represents a directory. It holds blobs as well as other sub-directories. A tree is a binary file that stores references to blobs and trees which are also named as SHA1 hash of the tree object.

## Commits:

Commit holds the current state of the repository. A commit is also named by SHA1 hash. You can consider a commit object as a node of the linked list. Every commit object has a pointer to the parent commit object. From a given commit, you can traverse back by looking at the parent pointer to view the history of the commit. If a commit has multiple parent commits, then that particular commit has been created by merging two branches.

## Branches:

Branches are used to create another line of development. By default, Git has a master branch, which is same as trunk in subversion. Usually, a branch is created to work on a new feature. Once the feature is completed, it is merged back with the master branch and we delete the branch. Every branch is referenced by HEAD, which points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit.

## Tags:

Tag assigns a meaningful name with a specific version in the repository. Tags are very similar to branches, but the difference is that tags are immutable. It means, tag is a branch, which nobody intends to modify. Once a tag is created for a particular commit, even if you create a new commit, it will not be updated. usually, developers create tags for product releases.

## Clone:

Clone operation creates the instance of the repository. Clone operation not only checks out the working copy, but it also mirrors the complete repository. Users can perform many operations with this local repository. The only time networking gets involved is when the repository instances are being synchronized.

## Pull:

Pull operation copies the changes from a remote repository instance to a local one. The pull operation is used for the synchronization b/w 2 repository instances. This is same as the update operation in subversion.

## Push:

Push operation copies the changes from a local repository instance to a remote one. This is used to store the changes permanently into the Git repository. This is same as the commit operation in subversion.

## HEAD:

HEAD is a pointer, which always points to the latest commit in the branch. whenever you make a commit, HEAD is updated with the latest commit. The heads of the branches are stored in ".git/refs/heads/" directory.

Revision:

Revision represents the version of the source code. Revisions in Git are represented by commits. These commits are identified by SHA1 secure hashes.

URL:

URL represents the location of the Git repository. Git URL is stored in config file.

Difference between Git & SVN:

(i) Git is a Distributed version control system (VCS),

SVN is a Non-distributed version control system.

(ii) Git has a centralized server & repository,

SVN is does not have a centralized server (or) repository.

(iii) The content in Git is stored as metadata,

SVN stores files of content.

(iv) Git branches are easier to work with than SVN branches.

(v) Git does not have the Global revision number feature like SVN has.

(vi) Git has better content protection than SVN.

(vii) Git was developed for Linux kernel by Linus Torvalds.

SVN was developed by CollabNet, INC.

(viii) Git is distributed under GNU, and its maintainance overseen by Junio Hamano; Apache subversion, or SVN, is distributed under the open source license.

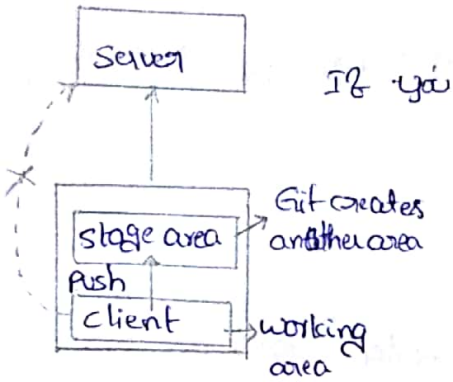
CVS

↓

SVN → SVN is centralized architecture

↓

GIT → Git is distributed architecture




If you don't have connection b/w server & client Then storage are comes into picture it is called "offline mode".

**GIT**

Client → staging area → Server

commit

Push (or) Remote push

(or) Remote location

**SVN**

client → server

Push

| SVN | GIT |
|---|---|
| Trunk | Master |
| Branch | Branch |
| Tag | Tag |

Terms in Git Getting the code.

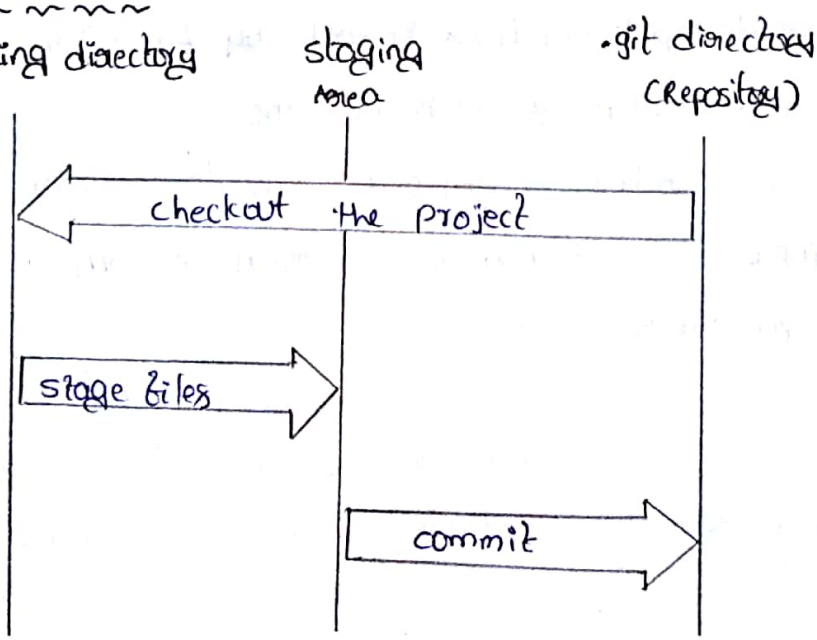
Pull.

Clone

Fetch

rebasing

Architecture of GIT:



# Environment setup:

## 1. Server setup in online:

Google

↓

GitHub.com

↓

step-1 : Registration

Name : Gangadhara018

username / Email : gangadharreddy.nallagari@gmail.com

Password : Weblogic

↓

choose

◉ unlimited public repositories for free.

○ ------

[continue] ↵

↓

step-2:

Q. How would you describe your level of programming experience?

○ very experienced ○ some what experienced ◉ Totally new to Programing

Q. What do you plan to use GitHub for?

☐ Development ☐ Design ☐ school projects

☐ Project manage-ment ☐ Research ☑ other (please specify) Learning

Q. Which is closest to how you would describe yourself?

○ I'm a hobbyist ○ I'm a professional ◉ I'm a student

○ other (please specify)

submit ↵ skip this step

↓

Learn Git & GitHub without any code!

Read the Guide | Start a project ↵

↓

Please verify your Email address.

Then go to Gmail & click on sended link of GitHub.

↓

Learn Git & GitHub without any code!

Read the Guide | Start a project ↵

↓

Create a New Repository

Owner: Gangadhar2018 / Repository Name: git-1

Description (optional)

It is my first repository

◉ public

○ private

☐ Initialize this repository with a README

Create Repository ↵

↓

Server is a ready GitHub.

Gangadhar018/git1

<> code

Quick setup - if you've done this kind of thing before

Set up in Desktop (or) HTTPS/SSH/https://github.com/Gangadhar018/git1.git

This is URL like in SVN URL

... or create a new repository on the command line

```
echo "#git1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Gangadhar018/git1.git
git push -u origin Master
```

... or push an existing repository from the command line

```
git remote add origin https://github.com/Gangadhar018/git1.git
git push -u origin master
```

... or import code from another repository

You can initialize this repository with code from a subversion, mercurial, or TFS project

Import code

ProTip! use the URL for this page when adding GitHub as a remote.

## 2. client setup in ocm system:

Google

↓

download git for windows

↓

Git for windows

↓

Download

'Git-2.17.0-64-bit.exe' downloaded

Then

Install

Click on 'Git-2.17.0-64-bit.exe'.

↓

Run

↓

Yes

↓

GNU Licence

↓

NEXT

↓

NEXT

↓

NEXT

↓

NEXT

↓

NEXT

↓

◉ use Git from Git Bash only

↓

NEXT

↓

NEXT

↓

NEXT

↓

NEXT

↓

INSTALL

↓

☑ Launch Git Bash

☒ view Release Notes

↓

Finish

↓

Git Bash opened. It is like Linux prompt.

HP @ HP-HP MINGW64 ~

$

Then we can make a directory our own

$ mkdir Project1

$ cd project1

HP @ HP-HP MINGW64 ~/Project1

$ pwd

/c/Users/HP/Project1

Step-1:

Then goto GitHub Repository token in google chrome page, Then

copy " echo "#git1" >> README.md "

&

Paste it in git bash & run it.

- $ echo "#git1" >> README.md

$ ls

o/p: [code is Ready]

README.md

step-2:

Then goto GitHub Repository

copy "git init" & paste it in git bash.

- $ git init → Initialization for New project.

o/p:

Initialized empty Git repository in c:/Users/HP/project1/.git/

Step-3:

Then goto gitHub Repository

copy "git add README.md" & paste it in git bash

- $ git add README.md

Now, the code will be in work area. Then it will add into staging area.

Step-4:

Then go to git Hub Repository.

copy "git commit -m "first commit" " & paste it in git bash

- $ git commit -m "first commit"

Op:

*** Please tell me who you are.

Run

git config --global user.email "you@example.com"

git config --global user.name "your Name".

to set your account's default identity.

Omit --global to set the identity only in this repository.

Then in git bash.

- $ git config --global user.email "gangadhaarreddy.nallagari@gmail.com"

Then

- $ git commit -m "first commit"

Olp:

[master (root-commit) 9ca4064] first commit

1 file changed, 1 insertion(+)

create mode 100644 README.md

step-5:

Then goto github Repository copy the other line & past it in git bash.

- $ git remote add origin http://github.com/Gangadhar018/git1.git/

Now, the code is in staging area, then code will pushed into Remote location.

step-6:

Then goto github Repository

copy the line "git push -u origin master" & paste it in gitbash.

- $ git push -u origin master

o/p: ::::: It will take some time for pushing the code
if it may be too long time taken then
press control+c & paste the same command.

UP to this first commit is completed.

Then it will ask username & password in new prompt.

Then go for next second commit, it means any updates in first commit.

Goto,

Computer
↓
Local disk "C-drive"
↓
Users
↓
HP
↓
Project 1
↓
README.md

make changes here & save this file.

Then go to Git bash.

- $ git commit -m "second commit"

o/p:
1 file changed, 2 insertions (+)

- $ git remote add origin https://github.com/Gangadhar018/git1.git

o/p:

fatal : remote origin already exists.

- $ git push -u origin master

o/p:

.....

To see the output in GitHub

click on

Gangadhar018 /git1 → open in new window

git 1

Gangadhar018

clone download

if you want to 'clone' download here

README.md

git1

updated content