DEVOPS

VERSION IT

What is Cloud ?

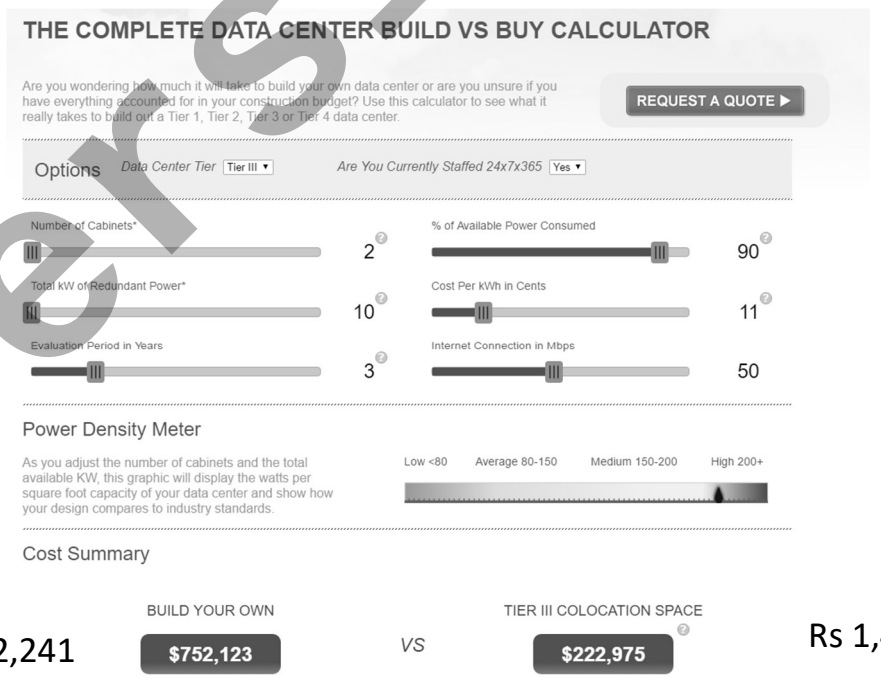
Data Center

- Tier 1 = Non-redundant capacity components (single uplink and servers).
- Tier 2 = Tier 1 + Redundant capacity components.
- Tier 3 = Tier 1 + Tier 2 + Dual-powered equipment's and multiple uplinks.
- Tier 4 = Tier 1 + Tier 2 + Tier 3 + all components are fully fault-tolerant including uplinks, storage, chillers, HVAC systems, servers etc. Everything is dual-powered.

Tier 4 data center considered as most robust and less prone to failures. Tier 4 is designed to host mission critical servers and computer systems, with fully redundant subsystems (cooling, power, network links, storage etc) and compartmentalized security zones controlled by biometric access controls methods. Naturally, the simplest is a Tier 1 data center used by small business or shops.

Data Center Availability According To Tiers

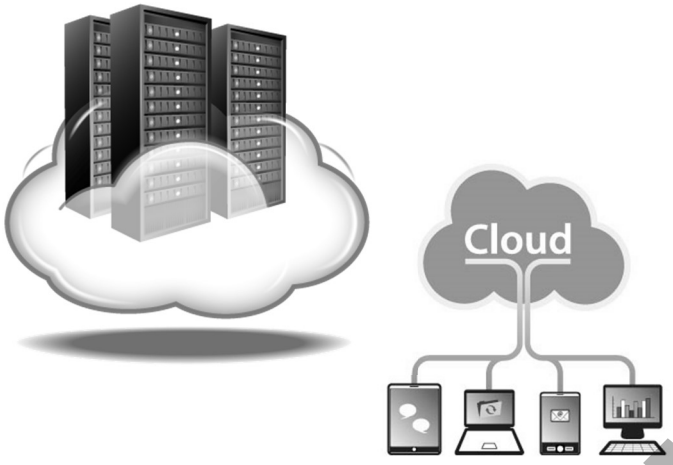
- Tier 1: Guaranteeing 99.671% availability.
- Tier 2: Guaranteeing 99.741% availability.
- Tier 3: Guaranteeing 99.982% availability.
- Tier 4: Guaranteeing 99.995% availability.



Rs 5,03,92,241

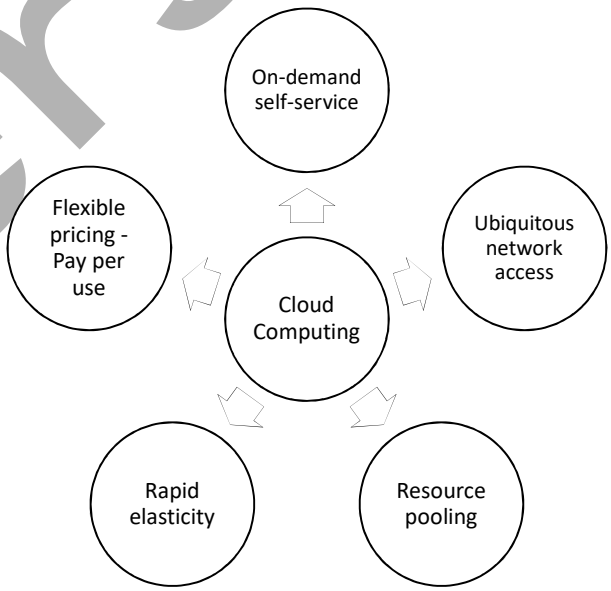
Rs 1,49,39,325

What is Cloud Computing



- Servers (Physical / Virtual)
- Storage
- Database
- Application Hosting
- Web Hosting
- Data Warehousing

Characteristics of Cloud Computing



Common characteristics

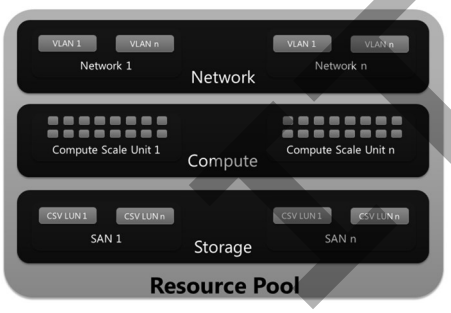
- **On-demand self-service**
- Ubiquitous network access
- Resource pooling (advanced virtualization)
- Rapid elasticity
- Flexible pricing - Pay per use

Common characteristics

- On-demand self-service
- **Ubiquitous network access**
- Resource pooling (advanced virtualization)
- Rapid elasticity
- Flexible pricing - Pay per use

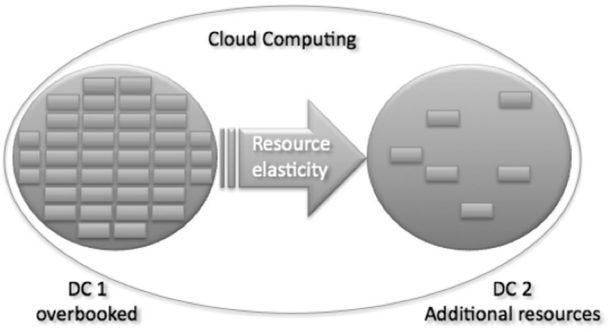
Common characteristics

- On-demand self-service
- Ubiquitous network access
- **Resource pooling (advanced virtualization)**
- Rapid elasticity
- Flexible pricing - Pay per use



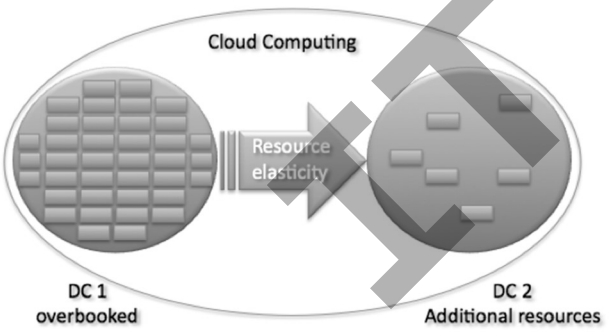
Common characteristics

- On-demand self-service
- Ubiquitous network access
- Resource pooling (advanced virtualization)
- **Rapid elasticity**
- Flexible pricing - Pay per use



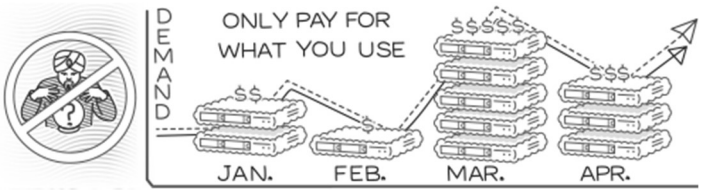
Common characteristics

- On-demand self-service
- Ubiquitous network access
- Resource pooling (advanced virtualization)
- **Rapid elasticity**
- Flexible pricing - Pay per use



Common characteristics

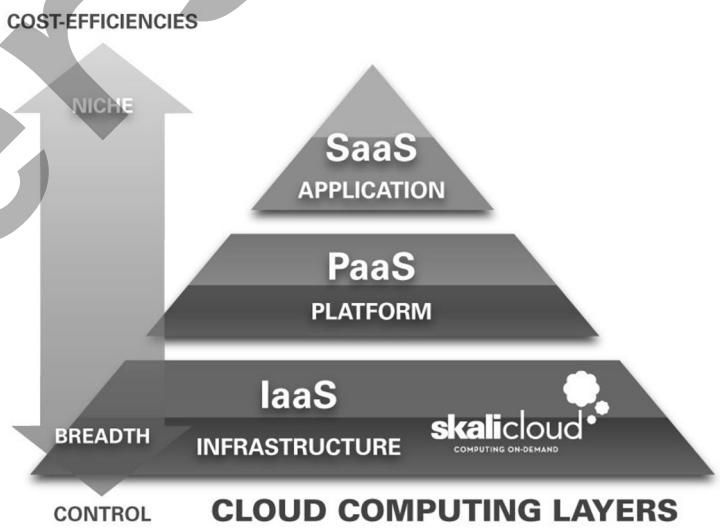
- On-demand self-service
- Ubiquitous network access
- Resource pooling (advanced virtualization)
- Rapid elasticity
- **Flexible pricing - Pay per use**



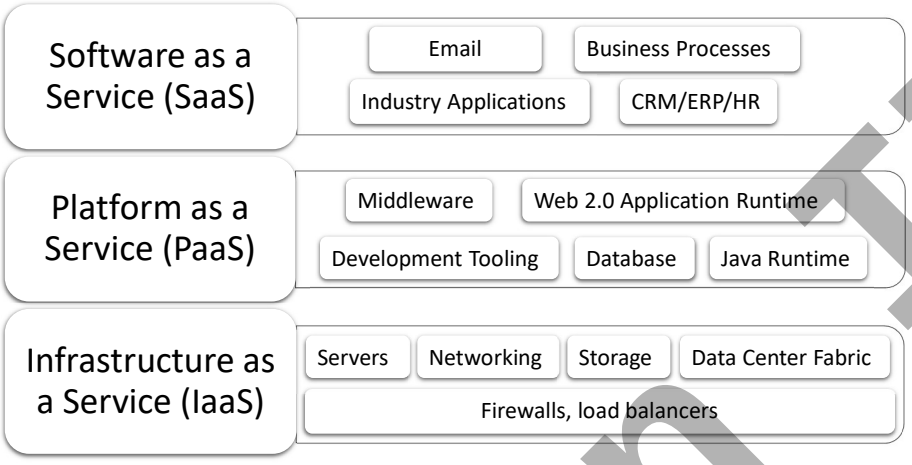
Advantages of Cloud Computing

- Lower Computing Cost
- Improved Performance
- Reduced Software Cost
- Instant Software Updates
- Unlimited Storage Capacity
- Increased Data Reliability
- Device Independence and the “always on!, anywhere and any place”
- Free From Maintenance and the “no-need-to-know”

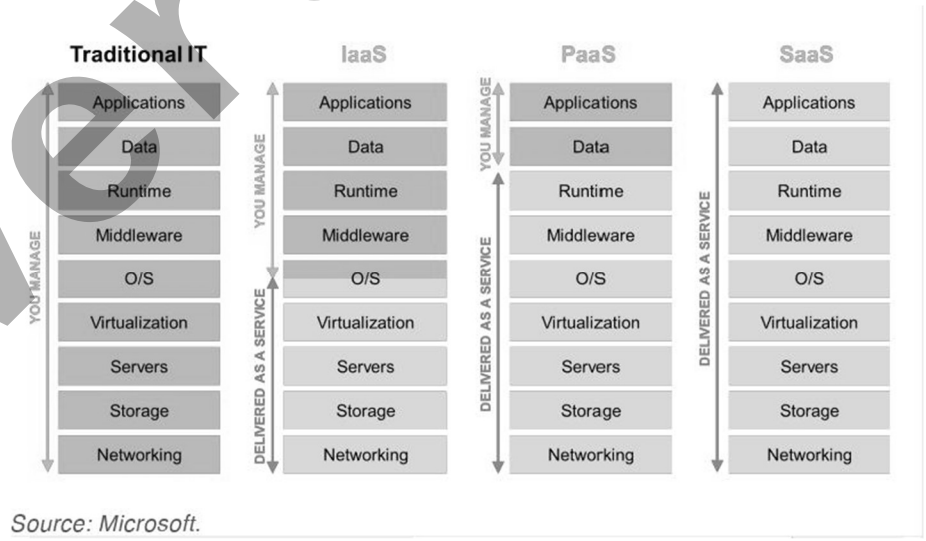
Cloud Service Layers



Cloud Service Containing

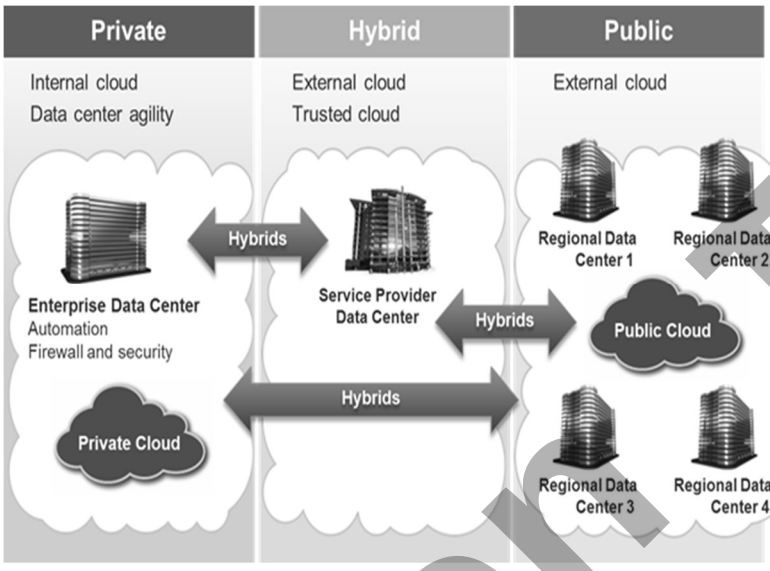


Cloud Service models - Comparison

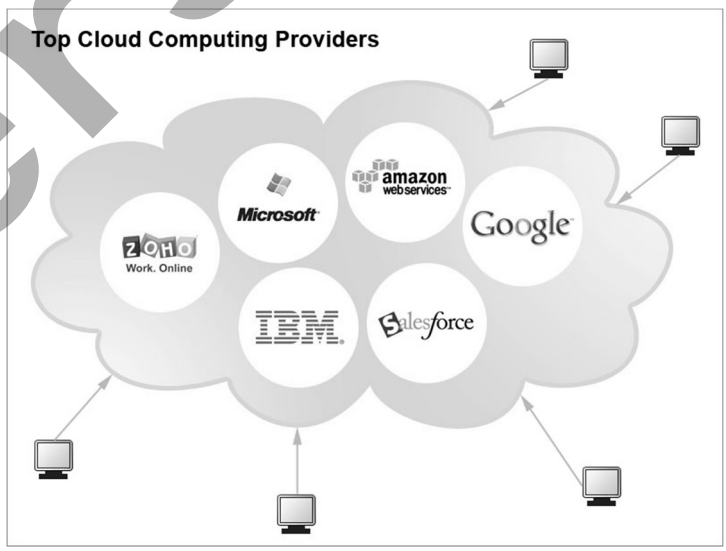


Source: Microsoft.

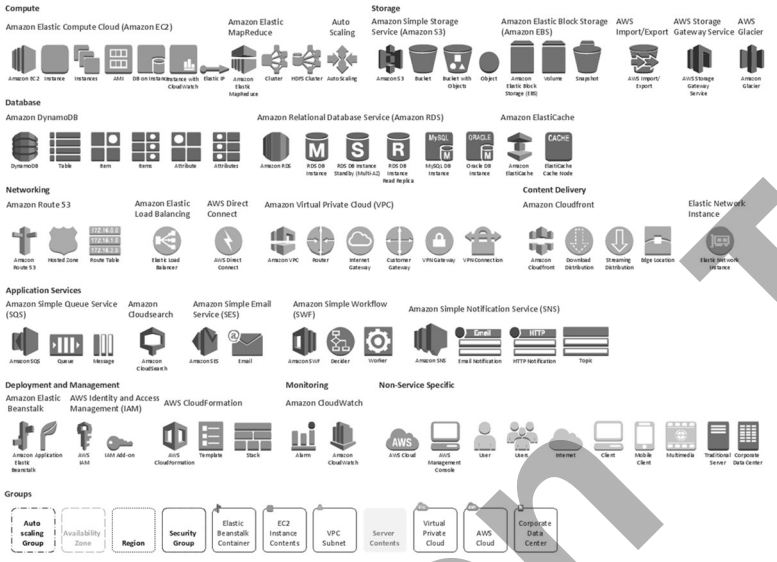
Cloud implementation types



Cloud Service Providers



Amazon Web Services



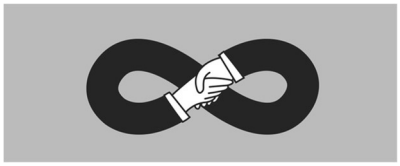
What is DevOps ?

ITS ALL ABOUT

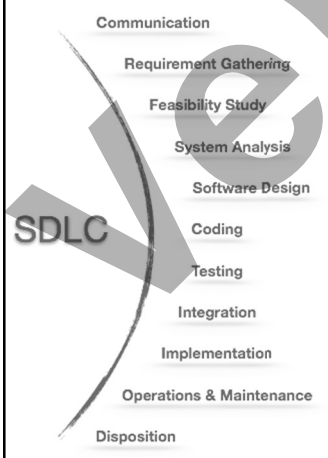
Collaboration

BETWEEN

Dev & Ops



Teams Involved



Development Team:

- Develops new features.
- Enhances the application functionality.
- Fix bugs.

Teams Involved

Build & Release Team:

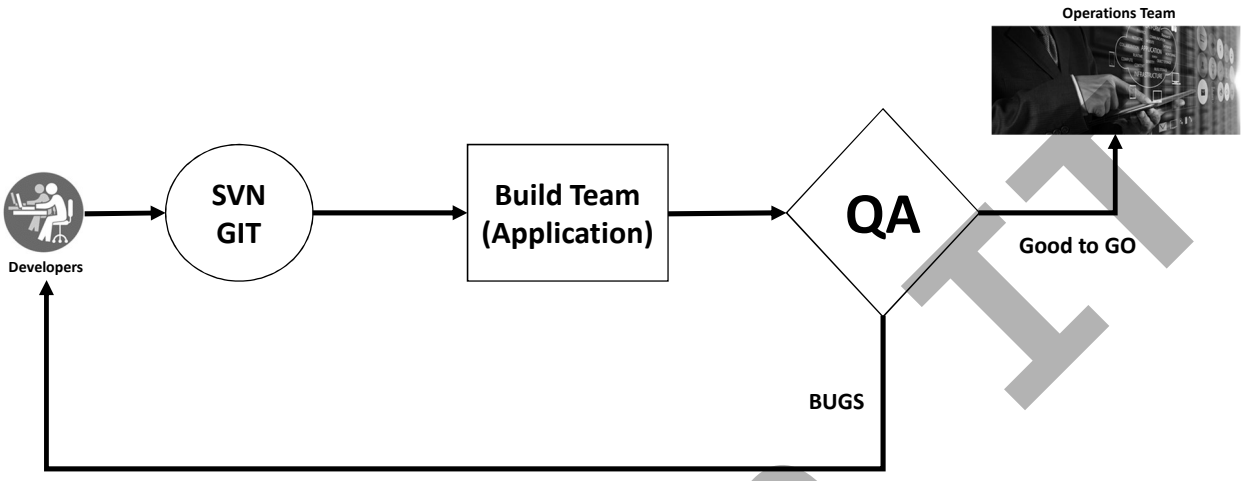
- Code repository Management.
- Branch Management.
- Working on Builds.
- Manage Repository Manager.

Teams Involved

QA Team:

- Apply test cases.
- Run on multiple platforms.
- Report Bugs.

Teams Involved

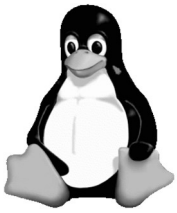


Teams Involved

Application Team:

- Maintains Applications on Production and UAT.
- Ensure Application have 100% uptime.
- Push new code to production servers to make it available for customers.

Teams Involved



Windows Server

OS Team:

- Responsible for Operating Systems.
- Responsible for 100% uptime of servers.
- Manages OS configurations.

Only 17% of IT teams can deliver fast enough*
Challenges…

More time testing, deploying and releasing than designing and building it

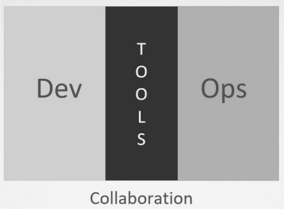
Production incidents result from human errors in manual release

IT Development and IT Operations are often not in alignment

What is Devops?

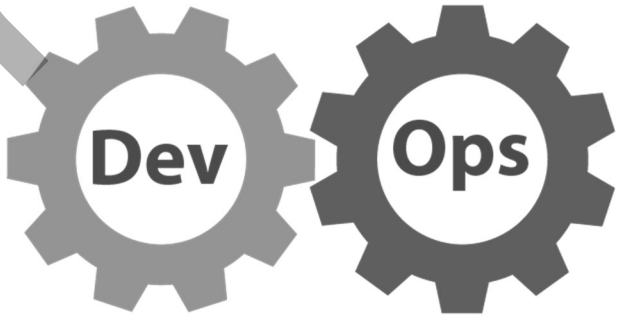
- DevOps started as a culture and set of practices to support collaboration and communication across development and operations, and to apply automation to key phases of the software delivery process.

Developers → Build Team → QA Team → Application Team → OS Team



Devops is a TOOLS based approach

DevOps



DEV Team → OPS Team

DevOps Process

- **Continuous Integration**
- **Continuous Delivery**
- **Continuous Deployment**
- **Configuration Management**

Continuous Integration (CI)

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

Continuous Integration brings multiple benefits to your organization:

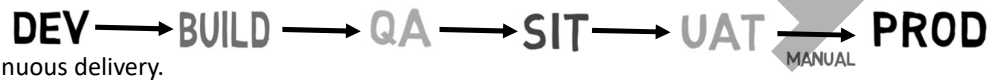
- Say goodbye to long and tense integrations
- Increase visibility which enables greater communication
- Catch issues fast and nip them in the bud
- Spend less time debugging and more time adding features
- Proceed in the confidence you're building on a solid foundation
- Stop waiting to find out if your code's going to work
- Reduce integration problems allowing you to deliver software more rapidly

Continuous Delivery (CD)

Continuous Delivery (CD) is a software development practice in which continuous integration, automated testing, and automated deployment capabilities allow high-quality software to be developed and deployed rapidly, reliably and repeatedly with minimal manual overhead.

Continuous Delivery brings multiple benefits to your organization:

- Deliver software with fewer bugs and lower risk.
- Release new features to market more frequently — and learn.
- Respond to marketing conditions more quickly.
- Life is saner for everyone: IT operations, software development, QA, product owners and business line owners.



When to use continuous delivery.

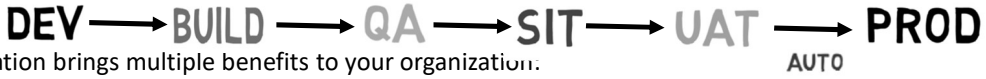
- You provide an online service
- Your release times are getting longer and longer
- You are developing a new product
- You have a big project with a lot of contributors
- You need to release frequent security patches

Continuous Deployment (CD)

Continuous deployment can be thought of as an extension of continuous integration, aiming at minimizing lead time, the time elapsed between development writing one new line of code and this new code being used by live users, in production.

To achieve continuous deployment, the team relies on infrastructure that automates and instruments the various steps leading up to deployment, so that after each integration successfully meeting these release criteria, the live application is updated with new code.

Instrumentation is needed to ensure that any suggestion of lowered quality results in aborting the deployment process, or rolling back the new features, and triggers human intervention.

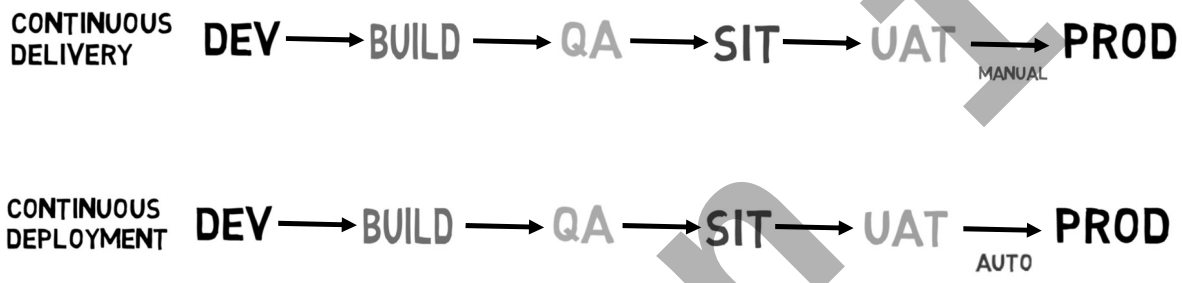


Continuous Integration brings multiple benefits to your organization.

- Early Releases
- Early Bug fixes

CD vs CD

While **Continuous Deployment** may not be right for every company, **Continuous Delivery** is an absolute requirement of **DevOps** practices. Only when you continuously deliver your code can you have true confidence that your changes will be serving value to your customers within minutes of pushing the "go" button, and that you can actually push that button any time the business is ready for it.



Configuration Management

Configuration Management is a set of interrelated processes, management techniques, and supporting tools that assure:

1. Our work products are as they should be and conform to requirements.
2. Changes to our the requirements are properly evaluated, authorized, and implemented.
3. All information/data necessary to manage our end items and other related work products is:

    a) kept current and accurate,
    b) properly structured for users' needs, and
    c) readily available to all who requires.

Configuration Management (CM) ensures that an organization is making informed business decisions, performing correct actions, and that all work products are what they are intended to be at every point in the lifecycle.

CM is about knowing what we did yesterday, what we are doing today, and what we will be doing in the future, as well as understanding the reasoning and authority behind every action/change that got us there.

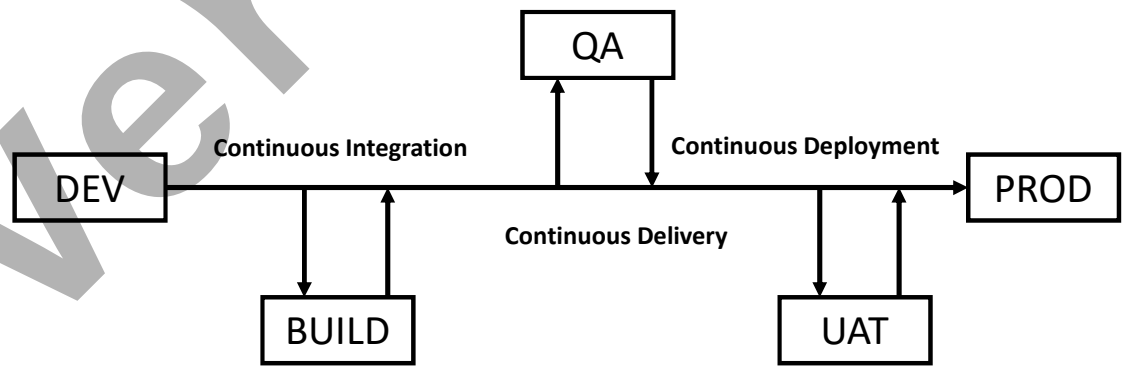
DevOps Culture

DevOps is less about what we do and more about how we do it. Technology infrastructure and evolving processes are critical in successfully transforming an organization to DevOps principles. But, at the end of the day, DevOps is about how work gets done, and how people interact with each other and with technology to drive performance.

DevOps is not an off-the-shelf solution that can just be implemented. Unfortunately, there is no one-size-fits-all approach for how DevOps should be implemented. Again, since DevOps is fundamentally a cultural shift in the way work gets done, it will take slightly different forms based on the organization.

The people side is critical to the equation. While most of the DevOps articles I have come across do make passing mention of the people side of the DevOps equation, very few go beyond a brief affirmation that it's essential before moving on to the process and technology/infrastructure components of the transformation. If we truly want to change the way people work to drive velocity in the tech world, then we must take a deeper look at this human side of DevOps, and the ways it can support or derail a sustainable DevOps transformation.

DevOps



Linux Basics

Command: **date**

1. Display Date from a String Value using --date Option

If you have a static date or time value in a string, you can use -d or --date option to convert the input string into date format as shown below.

Please note that this doesn't use the current date and time value. Instead is uses the date and time value that you pass as string.

The following examples takes an input date only string, and displays the output in date format. If you don't specify time, it uses 00:00:00 for time.

```
$ date --date="12/2/2014"
Tue Dec  2 00:00:00 PST 2014

$ date --date="2 Feb 2014"
Sun Feb  2 00:00:00 PST 2014

$ date --date="Feb 2 2014"
Sun Feb  2 00:00:00 PST 2014
```

The following example takes an input date and time string, and displays the output in date format.

```
$ date --date="Feb 2 2014 13:12:10"
Sun Feb  2 13:12:10 PST 2014
```

2. Read Date Patterns from a file using --file option

This is similar to the -d or --date option that we discussed above. But, you can do it for multiple date strings. If you have a file that contains various static date strings, you can use -f or --file option as shown below.

In this example, we can see that datefile contained 2 date strings. Each line of datefile is parsed by date command and date is outputted for each line.

```
$ cat datefile
Sept 9 1986
Aug 23 1987

$ date --file=datefile
Tue Sep  9 00:00:00 PDT 1986
Sun Aug 23 00:00:00 PDT 1987
```

3. Get Relative Date Using --date option

You can also use date command to get a future date using relative values.

For example, the following examples gets date of next Monday.

```
$ date --date="next mon"
Mon May 27 00:00:00 PDT 2013
```

If string=@is given to date command, then date command convert seconds since the epoch (1970-01-01 UTC) to a date.

It displays date in which 5 seconds are elapsed since epoch 1970-01-01 UTC:

```
$ date --date=@5
Wed Dec 31 16:00:05 PST 1969
```

It displays date in which 10 seconds are elapsed since epoch 1970-01-01 UTC:

```
$ date --date=@10
Wed Dec 31 16:00:10 PST 1969
```

It displays date in which 1 minute (i.e. 60 seconds) is elapsed since epoch 1970-01-01 UTC:

```
$ date --date=@60
Wed Dec 31 16:01:00 PST 1969
```

4. Display Past Date

You can display a past date using the -date command. Few possibilities are shown below.

```
$ date --date='3 seconds ago'
Mon May 20 21:59:20 PDT 2013

$ date --date="1 day ago"
Sun May 19 21:59:36 PDT 2013

$ date --date="yesterday"
Sun May 19 22:00:26 PDT 2013

$ date --date="1 month ago"
Sat Apr 20 21:59:58 PDT 2013

$ date --date="1 year ago"
Sun May 20 22:00:09 PDT 2012
```

5. Set Date and Time using –set option

You can set date and time of your system using -s or –set option as shown below..

In this example, initially it displayed the time as 20:09:31. We then used date command to change it to 21:00:00.

```
$ date
Sun May 20 20:09:31 PDT 2013

$ date -s "Sun May 20 21:00:00 PDT 2013"
Sun May 20 21:00:00 PDT 2013

$ date
Sun May 20 21:00:05 PDT 2013
```

5. Display Universal Time using -u option

You can display date in UTC format using -u, or –utc, or –universal option as shown below.

```
$ date
Mon May 20 22:07:53 PDT 2013

$ date -u
Tue May 21 05:07:55 UTC 2013
```

6. Display Last Modification Time using -r option

In this example, the current time is 20:25:48

```
$ date
Sun May 20 20:25:48 PDT 2013
```

The timestamp of datefile is changed using touch command. This was done few seconds after the above date command's output.

```
$ touch datefile
```

The current time after the above touch command is 20:26:12

```
$ date
Sun May 20 20:26:12 PDT 2013
```

Finally, use the date command -r option to display the last modified timestamp of a file as shown below. In this example, it displays last modified time of datefile as 20:25:57. It is somewhere between 20:25:48 and 20:26:12 (which is when we execute the above touch command to modify the timestamp).

```
$ date -r datefile
Sun May 20 20:25:57 PDT 2013
```

7. Various Date Command Formats

You can use formatting option to display date command in various formats using the following syntax:

```
$ date +%<format-option>
```

Command : **cal**

date displays the Linux or UNIX system current date and time. The cal command displays a simple calendar in traditional format.

cal command
Just enter cal command as follows:

```
[ec2-user@demohost ~]$ cal
    October 2016
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
[ec2-user@demohost ~]$
```

Print calendar for year 2008

```
[ec2-user@demohost ~]$ cal 2008
                                2008

       January               February                March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                  1  2                     1
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   2  3  4  5  6  7  8
13 14 15 16 17 18 19  10 11 12 13 14 15 16   9 10 11 12 13 14 15
20 21 22 23 24 25 26  17 18 19 20 21 22 23  16 17 18 19 20 21 22
27 28 29 30 31        24 25 26 27 28 29     23 24 25 26 27 28 29
                                            30 31

       April                 May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                  1  2  3   1  2  3  4  5  6  7
 6  7  8  9 10 11 12   4  5  6  7  8  9 10   8  9 10 11 12 13 14
13 14 15 16 17 18 19  11 12 13 14 15 16 17  15 16 17 18 19 20 21
20 21 22 23 24 25 26  18 19 20 21 22 23 24  22 23 24 25 26 27 28
27 28 29 30           25 26 27 28 29 30 31  29 30

       July                  August                September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                  1  2      1  2  3  4  5  6
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   7  8  9 10 11 12 13
13 14 15 16 17 18 19  10 11 12 13 14 15 16  14 15 16 17 18 19 20
20 21 22 23 24 25 26  17 18 19 20 21 22 23  21 22 23 24 25 26 27
27 28 29 30 31        24 25 26 27 28 29 30  28 29 30
                      31
       October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4                     1      1  2  3  4  5  6
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   7  8  9 10 11 12 13
12 13 14 15 16 17 18   9 10 11 12 13 14 15  14 15 16 17 18 19 20
19 20 21 22 23 24 25  16 17 18 19 20 21 22  21 22 23 24 25 26 27
26 27 28 29 30 31     23 24 25 26 27 28 29  28 29 30 31
                      30
```

Print calendar for particular month and year.

```
[ec2-user@demohost ~]$ cal 1 2008
    January 2008
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

[ec2-user@demohost ~]$
```

Getting HELP:

Command : **info**

info command provides read Info documents.

```
[ec2-user@demohost ~]$ info cal
[ec2-user@demohost ~]$


File: *manpages*,  Node: cal,  Up: (dir)

CAL(1)                          User Commands                          CAL(1)



NAME
       cal - display a calendar

SYNOPSIS
       cal [options] [[[day] month] year]

DESCRIPTION
       cal  displays  a  simple  calendar.  If no arguments are specified, the
       current month is displayed.

OPTIONS
       -1, --one
              Display single month output.  (This is the default.)

       -3, --three
              Display prev/current/next month output.
```

You can press ‘q’ to come out of info documents.

Command : **man**

man command provides an interface to the on-line reference manuals.
Manual pages of cal command looks like.

```
CAL(1)                                                          User Commands                                                          CAL(1)

NAME
       cal - display a calendar

SYNOPSIS
       cal [options] [[[day] month] year]

DESCRIPTION
       cal displays a simple calendar.  If no arguments are specified, the current month is displayed.

OPTIONS
       -1, --one
              Display single month output.  (This is the default.)

       -3, --three
```

You can press ‘q’ to come out of manual pages

Command : **whatis**

whatis command display manual page descriptions.

```
[ec2-user@demohost ~]$ whatis whatis
whatis (1)           - display manual page descriptions
[ec2-user@demohost ~]$ whatis cal
cal (1)              - display a calendar
[ec2-user@demohost ~]$ whatis date
date (1)             - print or set the system date and time
[ec2-user@demohost ~]$
```

Using **--help / -h** options for help.

On the command line if you need all the options which has to be used for that command then you can use command-name and then followed with --help or -h as option.

```
[ec2-user@demohost ~]$ cal --help

Usage:
 cal [options] [[[day] month] year]

Options:
 -1, --one        show only current month (default)
 -3, --three      show previous, current and next month
 -s, --sunday     Sunday as first day of week
 -m, --monday     Monday as first day of week
 -j, --julian     output Julian dates
 -y, --year       show whole current year
 -V, --version    display version information and exit
 -h, --help       display this help text and exit
```

```
[ec2-user@demohost ~]$ cal -h

Usage:
 cal [options] [[[day] month] year]

Options:
 -1, --one        show only current month (default)
 -3, --three      show previous, current and next month
 -s, --sunday     Sunday as first day of week
 -m, --monday     Monday as first day of week
 -j, --julian     output Julian dates
 -y, --year       show whole current year
 -V, --version    display version information and exit
 -h, --help       display this help text and exit
```

Handy Shortcuts:

```
# up(down)_key              scrolls through command history
# <something-incomplete> TAB     completes path/file_name
# Ctrl+a                  # cursor to beginning of command line
# Ctrl+e                  # cursor to end of command line
# Ctrl+d                  # delete character under cursor
# Ctrl+k                  # cut line from cursor into kill buffer
# Ctrl+y                  # paste content from Ctrl k
```

Command: **uname**

Command uname provides the system information like architecture and kernel version.

```
[ec2-user@demohost ~]$ uname
Linux
[ec2-user@demohost ~]$ uname -r
3.10.0-327.10.1.el7.x86_64
[ec2-user@demohost ~]$ uname -a
Linux demohost 3.10.0-327.10.1.el7.x86_64 #1 SMP Sat Jan 23 04:54:55 EST 2016 x86_64 x86_64 x86_64 GNU/Linux
[ec2-user@demohost ~]$ uname -m
x86_64
[ec2-user@demohost ~]$ uname -p
x86_64
[ec2-user@demohost ~]$
```

Command: **ls**

Command ls helps you display the files under the present directory. (In Linux every thing is a file)

```
[ec2-user@demohost ~]$ ls
abc  file1.txt  Maven.pem  newfile.txt  passwd
[ec2-user@demohost ~]$
```

To show the long list.

```
[ec2-user@demohost ~]$ ls -l
total 16
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:36 file1.txt
-r--------. 1 ec2-user ec2-user 1675 Oct  3 09:49 Maven.pem
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:38 newfile.txt
-rw-r--r--. 1 ec2-user ec2-user 1145 Oct  3 22:39 passwd
[ec2-user@demohost ~]$
```

To sort the files in time order.

```
[ec2-user@demohost ~]$ ls -lt
total 16
-rw-r--r--. 1 ec2-user ec2-user 1145 Oct  3 22:39 passwd
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:38 newfile.txt
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:36 file1.txt
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc
-r--------. 1 ec2-user ec2-user 1675 Oct  3 09:49 Maven.pem
[ec2-user@demohost ~]$
```

To sort the file in reverse time order.

```
[ec2-user@demohost ~]$ ls -ltr
total 16
-r--------. 1 ec2-user ec2-user 1675 Oct  3 09:49 Maven.pem
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:36 file1.txt
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:38 newfile.txt
-rw-r--r--. 1 ec2-user ec2-user 1145 Oct  3 22:39 passwd
[ec2-user@demohost ~]$
```

To show the hidden files.

```
[ec2-user@demohost ~]$ ls -a
.  ..  abc  .bash_history  .bash_logout  .bash_profile  .bashrc  file1.txt  Maven.pem  newfile.txt  passwd  .ssh
[ec2-user@demohost ~]$ ls -A
abc  .bash_history  .bash_logout  .bash_profile  .bashrc  file1.txt  Maven.pem  newfile.txt  passwd  .ssh
[ec2-user@demohost ~]$
```

To show the files sizes in human readable format.

```
[ec2-user@demohost ~]$ ls -lh
total 16K
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:36 file1.txt
-r--------. 1 ec2-user ec2-user 1.7K Oct  3 09:49 Maven.pem
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:38 newfile.txt
-rw-r--r--. 1 ec2-user ec2-user 1.2K Oct  3 22:39 passwd
[ec2-user@demohost ~]$
```

To show the type of a file at the end of file.

```
[ec2-user@demohost ~]$ ls -lF
total 16
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc|
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:36 file1.txt
-r--------. 1 ec2-user ec2-user 1675 Oct  3 09:49 Maven.pem
-r--------. 1 ec2-user ec2-user  385 Oct  3 22:38 newfile.txt
-rwxr-xr-x. 1 ec2-user ec2-user 1145 Oct  3 22:39 passwd*
[ec2-user@demohost ~]$
```

To show all the files which are ending with .log in that directory.

```
[ec2-user@demohost ~]$ ls -l *.txt
-r--------. 1 ec2-user ec2-user 385 Oct  3 22:36 file1.txt
-r--------. 1 ec2-user ec2-user 385 Oct  3 22:38 newfile.txt
[ec2-user@demohost ~]$
```

Commands for Files.

Command: **touch**

Command touch is used to create empty files in Linux OS.
If you try to create a file then it creates empty file and if the file already exists then it updates the time stamp of the file.

```
[ec2-user@demohost ~]$ ls
abc  Maven.pem  passwd
[ec2-user@demohost ~]$ touch file1.txt
[ec2-user@demohost ~]$ ls
abc  file1.txt  Maven.pem  passwd
[ec2-user@demohost ~]$ ls -l
total 8
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc
-rw-rw-r--. 1 ec2-user ec2-user    0 Oct  4 02:15 file1.txt
-r--------. 1 ec2-user ec2-user 1675 Oct  3 09:49 Maven.pem
-rwxr-xr-x. 1 ec2-user ec2-user 1145 Oct  3 22:39 passwd
[ec2-user@demohost ~]$
```

You can touch a file which already exists and you can see the change in timestamp.

```
[ec2-user@demohost ~]$ ls -l Maven.pem
-r--------. 1 ec2-user ec2-user 1675 Oct  3 09:49 Maven.pem
[ec2-user@demohost ~]$ date
Tue Oct  4 02:16:43 EDT 2016
[ec2-user@demohost ~]$ touch Maven.pem
[ec2-user@demohost ~]$ ls -l Maven.pem
-r--------. 1 ec2-user ec2-user 1675 Oct  4 02:16 Maven.pem
[ec2-user@demohost ~]$
```

You can create multiple files in one touch command.

```
[ec2-user@demohost ~]$ touch file{1..3}.txt
[ec2-user@demohost ~]$ ls -l *.txt
-rw-rw-r--. 1 ec2-user ec2-user 0 Oct  4 02:24 file1.txt
-rw-rw-r--. 1 ec2-user ec2-user 0 Oct  4 02:24 file2.txt
-rw-rw-r--. 1 ec2-user ec2-user 0 Oct  4 02:24 file3.txt
[ec2-user@demohost ~]$
```

Command: **rm**

Command rm is used to remove the files.

```
[ec2-user@demohost ~]$ rm file1.txt
rm: remove write-protected regular empty file ‘file1.txt’? y
[ec2-user@demohost ~]$
```

You can forcefully remove the files without prompting Y/N message.

```
[ec2-user@demohost ~]$ ls -l *.txt
-r--------. 1 ec2-user ec2-user 0 Oct  4 02:31 file1.txt
-r--------. 1 ec2-user ec2-user 0 Oct  4 02:31 file2.txt
-r--------. 1 ec2-user ec2-user 0 Oct  4 02:31 file3.txt
[ec2-user@demohost ~]$ rm -vf *.txt
removed ‘file1.txt’
removed ‘file2.txt’
removed ‘file3.txt’
[ec2-user@demohost ~]$
```

Command: **cp**

Command cp is to copy the files in same location or to another location.

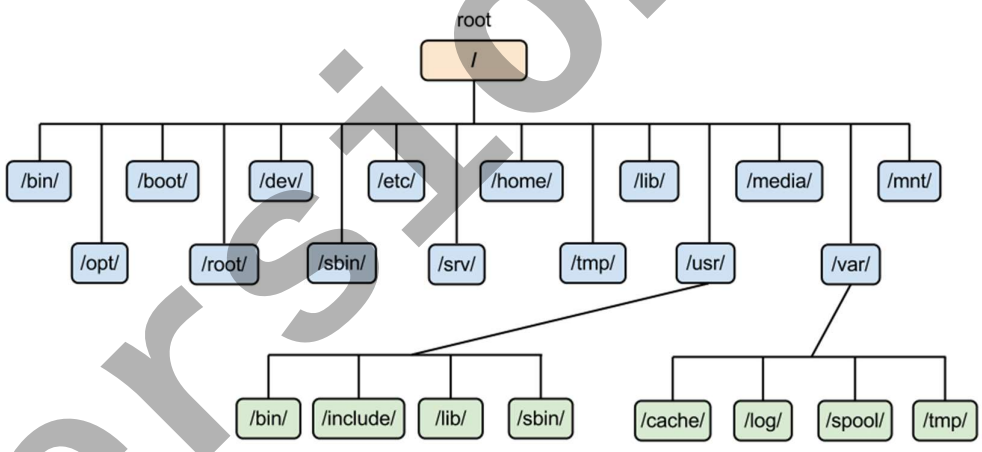
```
[ec2-user@demohost ~]$ ls -l messages
-rw-rw-rw-. 1 ec2-user ec2-user 157307 Oct  4 02:33 messages
[ec2-user@demohost ~]$ ls -lh messages
-rw-rw-rw-. 1 ec2-user ec2-user 154K Oct  4 02:33 messages
[ec2-user@demohost ~]$ cp messages newfile
[ec2-user@demohost ~]$ ls -lh newfile
-rw-rw-r--. 1 ec2-user ec2-user 154K Oct  4 02:34 newfile
[ec2-user@demohost ~]$
```

Command: **mv**

Command mv is to rename the files and as well as you can move the file from one location to another location.

```
[ec2-user@demohost ~]$ ls -l newfile
-rw-rw-r--. 1 ec2-user ec2-user 157307 Oct  4 02:34 newfile
[ec2-user@demohost ~]$ mv newfile newfile.bkp
[ec2-user@demohost ~]$ ls -l newfile
ls: cannot access newfile: No such file or directory
[ec2-user@demohost ~]$ ls -l newfile.bkp
-rw-rw-r--. 1 ec2-user ec2-user 157307 Oct  4 02:34 newfile.bkp
[ec2-user@demohost ~]$
```

Commands for Directories.



Command: **pwd**

Command pwd tell you in which directory you are navigated in.

```
[ec2-user@demohost ~]$ pwd
/home/ec2-user
[ec2-user@demohost ~]$ cd /boot
[ec2-user@demohost boot]$ pwd
/boot
[ec2-user@demohost boot]$ cd /tmp
[ec2-user@demohost tmp]$ pwd
/tmp
[ec2-user@demohost tmp]$
```

Command: **cd**

Command cd helps you in navigating from one directory to another directory.

```
[ec2-user@demohost ~]$ pwd
/home/ec2-user
[ec2-user@demohost ~]$ cd /boot
[ec2-user@demohost boot]$ pwd
/boot
[ec2-user@demohost boot]$ cd /tmp
[ec2-user@demohost tmp]$ pwd
/tmp
[ec2-user@demohost tmp]$
```

You can even run cd command without any options which it will take you to home directory.

```
[ec2-user@demohost tmp]$ pwd
/tmp
[ec2-user@demohost tmp]$ cd
[ec2-user@demohost ~]$ pwd
/home/ec2-user
[ec2-user@demohost ~]$
```

Command: **mkdir**

Command mkdir is used to create empty directory.

```
[ec2-user@demohost ~]$ ls
abc  Maven.pem
[ec2-user@demohost ~]$ mkdir newdir
[ec2-user@demohost ~]$ ls
abc  Maven.pem  newdir
[ec2-user@demohost ~]$ ls -l
total 4
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc
-r--------. 1 ec2-user ec2-user 1675 Oct  4 02:16 Maven.pem
drwxrwxr-x. 2 ec2-user ec2-user    6 Oct  4 02:39 newdir
[ec2-user@demohost ~]$
```

You can create recursive directories using -p option in mkdir command.

```
[ec2-user@demohost ~]$ ls
abc  Maven.pem
[ec2-user@demohost ~]$ mkdir newdir/dir1
mkdir: cannot create directory ‘newdir/dir1’: No such file or directory
[ec2-user@demohost ~]$ mkdir -p newdir/dir1
[ec2-user@demohost ~]$ ls -l
total 4
prw-rw-r--. 1 ec2-user ec2-user    0 Oct  3 22:27 abc
-r--------. 1 ec2-user ec2-user 1675 Oct  4 02:16 Maven.pem
drwxrwxr-x. 3 ec2-user ec2-user   17 Oct  4 02:40 newdir
[ec2-user@demohost ~]$ ls -l newdir/
total 0
drwxrwxr-x. 2 ec2-user ec2-user 6 Oct  4 02:40 dir1
[ec2-user@demohost ~]$
```

Command: **rmdir**

Command rmdir is to remove the empty directories.

```
[ec2-user@demohost ~]$ mkdir newdir
[ec2-user@demohost ~]$ ls
abc  Maven.pem  newdir
[ec2-user@demohost ~]$ rmdir newdir
[ec2-user@demohost ~]$ ls
abc  Maven.pem
[ec2-user@demohost ~]$
```

In case if you have any other files and any directories inside this directory then rmdir command cannot remove the directory.

```
[ec2-user@demohost ~]$ ls
abc  Maven.pem
[ec2-user@demohost ~]$ mkdir dir1
[ec2-user@demohost ~]$ cd dir1/
[ec2-user@demohost dir1]$ touch 1 2 3
[ec2-user@demohost dir1]$ cd ..
[ec2-user@demohost ~]$ rmdir dir1
rmdir: failed to remove ‘dir1’: Directory not empty
[ec2-user@demohost ~]$
```

You can remove the directories also by using **rm** command with -r option.

```
[ec2-user@demohost ~]$ ls dir1
1  2  3
[ec2-user@demohost ~]$ rm -r dir1
[ec2-user@demohost ~]$ ls
abc  Maven.pem
[ec2-user@demohost ~]$
```

Command: **cp**

Command cp is used to copy files as well as directories also. But when you deal with directories you need to use -r option which is almost same like rm command.

```
[ec2-user@demohost ~]$ ls dir1/
1  2  3
[ec2-user@demohost ~]$ cp dir1 dir2
cp: omitting directory ‘dir1’
[ec2-user@demohost ~]$ cp -r dir1 dir2
[ec2-user@demohost ~]$ ls dir2
1  2  3
[ec2-user@demohost ~]$
```

You can use -v option to show the output of files getting copied on the screen. -v stands for enabling verbose output.

```
[ec2-user@demohost ~]$ cp -rv dir1 dir2
‘dir1’ -> ‘dir2’
‘dir1/1’ -> ‘dir2/1’
‘dir1/2’ -> ‘dir2/2’
‘dir1/3’ -> ‘dir2/3’
[ec2-user@demohost ~]$
```

Command: **mv**

Command mv is used to rename directories as like files but also can move a file from one directory to another directory.

```
[ec2-user@demohost ~]$ ls
abc  dir1  dir2  Maven.pem
[ec2-user@demohost ~]$ mv dir2 dir3
[ec2-user@demohost ~]$ ls
abc  dir1  dir3  Maven.pem
[ec2-user@demohost ~]$
```

In above example you just renamed directory. But if you provide a directory in target location and if that directory already exists then it will move the directory to that new directory.

```
[ec2-user@demohost ~]$ ls
abc  dir1  dir3  Maven.pem
[ec2-user@demohost ~]$ mv dir1 dir3
[ec2-user@demohost ~]$ ls
abc  dir3  Maven.pem
[ec2-user@demohost ~]$ ls dir3
1  2  3  dir1
[ec2-user@demohost ~]$
```

Likewise you can move a file to another directory also by using mv command.

```
[ec2-user@demohost ~]$ ls
abc  dir3  Maven.pem  new.log
[ec2-user@demohost ~]$ mv new.log dir3/
[ec2-user@demohost ~]$ ls
abc  dir3  Maven.pem
[ec2-user@demohost ~]$ ls dir3
1  2  3  dir1  new.log
[ec2-user@demohost ~]$
```

Viewing files.

Command: **file**

Command file is to determine what file content that a file has.

```
[ec2-user@demohost ~]$ file Maven.pem
Maven.pem: PEM RSA private key
[ec2-user@demohost ~]$ file abc
abc: fifo (named pipe)
[ec2-user@demohost ~]$ file dir3/new.log
dir3/new.log: empty
[ec2-user@demohost ~]$ file /etc/passwd
/etc/passwd: ASCII text
[ec2-user@demohost ~]$ file /dev/xvda
/dev/xvda: block special
[ec2-user@demohost ~]$ file /dev/pts1
```

Command: **cat**

Command cat is used to concatenate the content of a file.

```
[ec2-user@demohost ~]$ ls
abc  file1.txt  Maven.pem
[ec2-user@demohost ~]$ cat file1.txt
Hello World!
[ec2-user@demohost ~]$
```

Command: **less**

Command less is used to view the content in a read-only viewer. Note that you cannot edit the content of a file using file editor.

```
[ec2-user@demohost ~]$ less file1.txt
[ec2-user@demohost ~]$
```

space	ahead one full screen
ENTER	ahead one line
b	back one full screen
k	back one line
g	top of the file
G	bottom of the file
/text	search forward for text (Regular Expressions can be used)
n	repeat last search
N	repeat backward last search
q	quit

Command: **vi/vim**

Command vi is the default editor in Linux OS, vim is the enhanced version of vi editor.

Insert mode

- i begins insert mode at the cursor
- A append to end of line
- I insert at beginning of line
- o insert new a line (below)
- O insert new line (above

Ex Mode

- :w writes (saves) the file to disk
- :wq writes and quits
- :q! quits, even if changes are lost

Manipulating Text (Command Mode)

Action followed by Target

Possible actions:

- change (c)
- cut (d)
- yank (y)
- paste (p) without target

Search and Replace (Command Mode)

- /, n, N Search
- <>/<>/<> Search/Replace (as in sed command)

```
:1,5s/cat/dog/g  # search in lines 1 to 5 and replace all words in any line
:%s/cat/dog/gi   # the whole file
```

- Goto Starting of the file (gg)
- Goto Ending of the file (G)
- Goto Specific line (:<line-no>) (Eg- :5)
- To search a word (/<word>)
- To search with case insensitivity
  :set ic
  /<word>
- To turn off case insensitivity (:set noic)
- To display line numbers (:set nu)
- To disable line numbers (:set nonu)

Command: **head**

Command head is used to display the content from top and it displays first 10 lines of a file. In case if you need specific lines then use -n option followed with number of lines.

```
[ec2-user@demohost ~]$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
[ec2-user@demohost ~]$
```

```
[ec2-user@demohost ~]$ head -n 3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
[ec2-user@demohost ~]$
```

Command: **tail**

Command **tail** is used to display the content from bottom and it displays bottom **10** lines by default. In case of you need any specific number of lines then you have to use **-n** option with number of lines.

```
[ec2-user@demohost ~]$ tail /etc/passwd
systemd-bus-proxy:x:999:997:systemd Bus Proxy:/:/sbin/nologin
systemd-network:x:998:996:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:997:995:User for polkitd:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:996:993::/var/lib/chrony:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
ec2-user:x:1000:1000:Cloud User:/home/ec2-user:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
[ec2-user@demohost ~]$
```

```
[ec2-user@demohost ~]$ tail -n 2 /etc/passwd
ec2-user:x:1000:1000:Cloud User:/home/ec2-user:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
[ec2-user@demohost ~]$
```

Command: **grep**

Command **grep** is used to search the content in file over command line.

```
[ec2-user@demohost ~]$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[ec2-user@demohost ~]$
```

Options:

-i ← To search with case insensitivity.

```
[ec2-user@demohost ~]$ grep -i root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[ec2-user@demohost ~]$
```

-c ← To count the number of lines from the output

```
[ec2-user@demohost ~]$ grep -c root /etc/passwd
2
[ec2-user@demohost ~]$
```

-e ← To search multiple words.

```
[ec2-user@demohost ~]$ grep -e root -e ec2-user /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
ec2-user:x:1000:1000:Cloud User:/home/ec2-user:/bin/bash
[ec2-user@demohost ~]$
```

Command: **cut**

Command **cut** is used to cut the output vertically, By default it uses **TAB** space as a default delimiter but you can specify which delimiter you would like to use and also you have to specify which fields you want by using -f option.

```
[ec2-user@demohost ~]$ cut -d : -f1 /etc/passwd
root
bin
daemon
adm
lp
```

Command: **find**

Command **find** is used to find the files across different directories based on the search criteria you provide.

Syntax: `find <dirs> [conditions] [–exec cmd {} \;]`

```
-atime n            File was last accessed n days ago
-ctime n            File was last changed n days ago
-user uname         File is owned by user uname (or user ID)
-group gname        File belongs to group gname (or group ID)
-size n[cwbkMG]     b 512-byte blocks (default), c in bytes, w two-byte words, k kilobyte
-iname              case -insensitive version of –name
-o                  logical operator between criteria (by default it is AND)
-not                negate (logical NOT)
-perm mode          permission bits are exactly mode (octal or symbolic).
-perm -mode         ALL of the permission bits mode are set for the file.
-perm +mode         Any of the permission bits mode are set for the file.
-regex pattern      Full path filename (not only filename) matches regular expression
pattern.
-mtime n            Files was last modified Exactly n*24 hours ago.
-mtime +n           Files was last modified >= n*24 hours ago.
-mtime -n           Files was last modified <= n*24 hours ago.
-mmin n             Files was last modified n minutes ago.
–daystart           measure time in the options above from the beginning of the current day
instead of 24 hours ago.
-newer <file>       Files newer than <file> modification date
```

You can find the examples in following.

```
find . –name "*.html"
find -iname snow.png
find -user peter -group peter
find -user joe -not -group joe
find -user joe -o -user jane
find -not \( -user joe -o -user jane \)
find -perm 755 # matches if mode is exactly 755
find -perm +222 # matches if anyone can write
find -perm -222 # matches if everyone can write
find -perm -002 # matches if other can write
find -size 1024k # excatly 1 MB
find -size +1024k # over 1 MB
find -size -1024k # less than 1 MB
find ~ –empty # find empty regular files or directories
find -size +102400k -ok gzip {} \; # OK prompte before acting
find . -regex '.*[124].*ms$'
find ~ –mtime 1 # files modified exactly 24 hours ago
find ~ –mtime 1 –daystart # modified yesterday
find ~ –mtime +356 # one year or longer ago
find ~ –mtime 2 –mtime –4 –daystart # two to four days ago
# files that were modified after May 4 of the current year
touch –t 05040000 /tmp/timestamp
find ~ –newer /tmp/timestamp
```

Command: **locate**

Command **locate** is used to find the files which are indexed in **mlocatedb**. It is fast but may not be right data. You can update the index and then search for data which gives you perfect file information.

You can find the files using following syntax.

```
[root@demohost ~]# locate vhosts
/usr/share/doc/httpd-2.4.6/httpd-vhosts.conf
[root@demohost ~]#
```

You can also use REGEX while searching.

```
[root@demohost ~]# locate *.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/broken.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/syntaxerror.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/test.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/foo/test.html
/usr/share/doc/Red_Hat_Enterprise_Linux-Release_Notes-7-en-US-7/index.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/advanced.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/appconvert.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/components.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/gssapi.html
```

You can enable case-insensitivity search.

```
[root@demohost ~]# locate -i *.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/broken.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/syntaxerror.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/test.html
/usr/lib/python2.7/site-packages/jinja2/testsuite/res/templates/foo/test.html
/usr/share/doc/Red_Hat_Enterprise_Linux-Release_Notes-7-en-US-7/index.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/advanced.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/appconvert.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/components.html
/usr/share/doc/cyrus-sasl-lib-2.1.26/gssapi.html
```

You can update the index / database.

```
[root@demohost ~]# updatedb
[root@demohost ~]#
```

Command: **ln**

Command **ln** is used to create a link for an existing file. Links are of two types one is Soft Link and another is Hard Link.

To create soft link.

```
[ec2-user@demohost ~]$ ls
abc  file1.txt  messages  new  newdir
[ec2-user@demohost ~]$ ln -s messages it_is_a_link
[ec2-user@demohost ~]$ ls -l it_is_a_link
lrwxrwxrwx 1 ec2-user ec2-user 8 Oct  5 07:05 it_is_a_link -> messages
[ec2-user@demohost ~]$ file it_is_a_link
it_is_a_link: symbolic link to `messages'
[ec2-user@demohost ~]$
```

To create hard link.

```
[ec2-user@demohost ~]$ ln messages messages.bkp
[ec2-user@demohost ~]$ ls -l messages*
-rw-rw-rw- 2 root root 326853 Oct  4 22:32 messages
-rw-rw-rw- 2 root root 326853 Oct  4 22:32 messages.bkp
[ec2-user@demohost ~]$ file messages.bkp
messages.bkp: UTF-8 Unicode text, with very long lines
[ec2-user@demohost ~]$ stat messages.bkp
  File: ‘messages.bkp’
  Size: 326853          Blocks: 640        IO Block: 4096   regular file
Device: ca02h/51714d    Inode: 207098      Links: 2
Access: (0666/-rw-rw-rw-)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-10-05 07:07:05.056044423 -0400
Modify: 2016-10-04 22:32:11.911836103 -0400
Change: 2016-10-05 07:06:53.167065923 -0400
 Birth: -
[ec2-user@demohost ~]$ stat messages
  File: ‘messages’
  Size: 326853          Blocks: 640        IO Block: 4096   regular file
Device: ca02h/51714d    Inode: 207098      Links: 2
Access: (0666/-rw-rw-rw-)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-10-05 07:07:05.056044423 -0400
Modify: 2016-10-04 22:32:11.911836103 -0400
Change: 2016-10-05 07:06:53.167065923 -0400
 Birth: -
[ec2-user@demohost ~]$
```

Command: **df**

Command **df** is used to show the information of mounted file systems on the server.

```
[root@demohost ~]# df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/xvda2      10473452 1341244   9132208  13% /
devtmpfs          489456       0    489456   0% /dev
tmpfs             507736       0    507736   0% /dev/shm
tmpfs             507736   12952    494784   3% /run
tmpfs             507736       0    507736   0% /sys/fs/cgroup
tmpfs             101548       0    101548   0% /run/user/1000
[root@demohost ~]#
```

You can get the human readable formats using -h option.

```
[root@demohost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda2       10G  1.3G  8.8G  13% /
devtmpfs        478M     0  478M   0% /dev
tmpfs           496M     0  496M   0% /dev/shm
tmpfs           496M   13M  484M   3% /run
tmpfs           496M     0  496M   0% /sys/fs/cgroup
tmpfs           100M     0  100M   0% /run/user/1000
[root@demohost ~]#
```

You can also check the type of mounted file system using -T option.

```
[root@demohost ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
/dev/xvda2     xfs        10G  1.3G  8.8G  13% /
devtmpfs       devtmpfs  478M     0  478M   0% /dev
tmpfs          tmpfs     496M     0  496M   0% /dev/shm
tmpfs          tmpfs     496M   13M  484M   3% /run
tmpfs          tmpfs     496M     0  496M   0% /sys/fs/cgroup
tmpfs          tmpfs     100M     0  100M   0% /run/user/1000
[root@demohost ~]#
```

Command: **du**

Command **du** is used to check the actual size of a directory or a file.

```
[root@demohost ~]# du -s /var/log/messages
372     /var/log/messages
[root@demohost ~]# du -s /etc
22680   /etc
[root@demohost ~]#
```

```
[root@demohost ~]# du -sh /etc
23M     /etc
[root@demohost ~]# du -sh /var/log/messages
372K    /var/log/messages
[root@demohost ~]#
```

Command: **gzip**

Command **gzip** is used to compress the file size to one fourth if it is text file.

To compress the file

```
[root@demohost ~]# ls -lh messages
-rw-r--r-- 1 root root 5.4M Oct  5 07:22 messages
[root@demohost ~]# gzip messages
[root@demohost ~]# ls -l messages.gz
-rw-r--r-- 1 root root 624371 Oct  5 07:22 messages.gz
[root@demohost ~]# ls -lh messages.gz
-rw-r--r-- 1 root root 610K Oct  5 07:22 messages.gz
[root@demohost ~]#
```

Command: **gunzip**

Command **gunzip** is used to uncompress the files which are compressed with **gzip** utility. Also you can use gzip -d option to uncompress it even.

```
[root@demohost ~]# cp messages.gz mess.gz
[root@demohost ~]# gunzip messages.gz
[root@demohost ~]# ls -l messages
-rw-r--r-- 1 root root 5609895 Oct  5 07:22 messages
[root@demohost ~]# ls -lh messages
-rw-r--r-- 1 root root 5.4M Oct  5 07:22 messages
[root@demohost ~]# gzip -d mess.gz
[root@demohost ~]# ls -lh
total 11M
-rw-r--r-- 1 root root 5.4M Oct  5 07:26 mess
-rw-r--r-- 1 root root 5.4M Oct  5 07:22 messages
[root@demohost ~]#
```

Command: **bzip2**

Command **bzip2** is another tool to compress the text file and it can even compress till one eighth of its total size.

To compress the file using bzip2 utility.

```
[root@demohost ~]# ls -lh messages
-rw-r--r-- 1 root root 5.4M Oct  5 07:22 messages
[root@demohost ~]# bzip2 messages
[root@demohost ~]# ls -lh messages.bz2
-rw-r--r-- 1 root root 227K Oct  5 07:22 messages.bz2
[root@demohost ~]#
```

Command: **bunzip2**

Command **bunzip2** is used to uncompress the **bzip2** compressed file. Also you can use **bzip2** utility to uncompress by using -d option.

```
[root@demohost ~]# ls -lh messages.bz2
-rw-r--r-- 1 root root 227K Oct  5 07:22 messages.bz2
[root@demohost ~]# bunzip2 messages.bz2
[root@demohost ~]# ls -lh messages
-rw-r--r-- 1 root root 5.4M Oct  5 07:22 messages
[root@demohost ~]# ls -l another.bz2
-rw-r--r-- 1 root root 232083 Oct  5 07:36 another.bz2
[root@demohost ~]# bzip2 -d another.bz2
[root@demohost ~]# ls -lh another
-rw-r--r-- 1 root root 5.4M Oct  5 07:36 another
[root@demohost ~]#
```

Command: **zip**

Command **zip** is to archive the directory and its contents into a single file.

```
[root@demohost ~]# du -sh /etc
23M     /etc
[root@demohost ~]# zip -r etc.zip /etc &>/dev/null
[root@demohost ~]# ls -lh etc.zip
-rw-r--r-- 1 root root 11M Oct  5 07:46 etc.zip
[root@demohost ~]#
```

Command: **unzip**

Command **unzip** is used to extract the file from .zip file created by **zip** utility.

```
[root@demohost ~]# unzip etc.zip &>/dev/null
[root@demohost ~]# ls
etc  etc.zip
[root@demohost ~]# ls etc |wc -l
178
[root@demohost ~]#
```

Command: **tar**

Command **tar** is used to take the archive of a directory and files inside it like zip utility.

c	To create archive
x	To extract archive
t	To test the archive
z	To enable gzip compression
j	To enable bzip2 compression.

To create an archive.

```
[root@demohost ~]# tar -cf etc.tar /etc
tar: Removing leading `/' from member names
[root@demohost ~]# ls -lh etc.tar
-rw-r--r-- 1 root root 21M Oct  5 07:52 etc.tar
[root@demohost ~]# gzip etc.tar
[root@demohost ~]# ls -lh etc.tar.gz
-rw-r--r-- 1 root root 6.9M Oct  5 07:52 etc.tar.gz
[root@demohost ~]#
```

You can directly create an archive with compression.

```
[root@demohost ~]# tar -czf etc.tar.gz /etc
tar: Removing leading `/' from member names
[root@demohost ~]# ls -lh etc.tar.gz
-rw-r--r-- 1 root root 6.9M Oct  5 07:54 etc.tar.gz
[root@demohost ~]#
```

To extract that archive.

```
[root@demohost ~]# cp etc.tar.gz new.tar.gz
[root@demohost ~]# gunzip etc.tar.gz
[root@demohost ~]# tar -xf etc.tar
[root@demohost ~]# du -sh etc
23M     etc
[root@demohost ~]# rm -rf etc
[root@demohost ~]# tar -xžf new.tar.gz
[root@demohost ~]# du -sh etc
23M     etc
[root@demohost ~]#
```

You can test the archive by using -t option.

Standard I/O & Pipes:

STDIN (<) : In order to take the input from a file instead of Keyboard we use STDIN Redirector.

```
[ec2-user@demohost ~]$ mail -s TestMail root@localhost <messages
[ec2-user@demohost ~]$ sudo su -
Last login: Wed Oct  5 08:53:59 EDT 2016 on pts/1
[root@demohost ~]# mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N  1 Cloud User            Wed Oct  5 19:30 3605/327478 "TestMail"
&
```

STDOUT (> or 1>) : In order to get the output of a command to a file then we use STDOUT Redirector.

```
[ec2-user@demohost ~]$ ls -l
total 648
prw-rw-r--. 1 ec2-user ec2-user      0 Oct  3 22:27 abc
-rw-rw-rw-. 1 ec2-user ec2-user   1675 Oct  4 02:16 file1.txt
-rw-rw-rw-  2 root     root     326853 Oct  4 22:32 messages
-rw-rw-rw-  2 root     root     326853 Oct  4 22:32 messages.bkp
-rw-rw-r--  1 ec2-user ec2-user     19 Oct  4 22:57 new
drwxrwxr-x  3 ec2-user ec2-user     37 Oct  4 22:08 newdir
[ec2-user@demohost ~]$ ls -l >out
[ec2-user@demohost ~]$ cat out
total 648
prw-rw-r--. 1 ec2-user ec2-user      0 Oct  3 22:27 abc
-rw-rw-rw-. 1 ec2-user ec2-user   1675 Oct  4 02:16 file1.txt
-rw-rw-rw-  2 root     root     326853 Oct  4 22:32 messages
-rw-rw-rw-  2 root     root     326853 Oct  4 22:32 messages.bkp
-rw-rw-r--  1 ec2-user ec2-user     19 Oct  4 22:57 new
drwxrwxr-x  3 ec2-user ec2-user     37 Oct  4 22:08 newdir
-rw-rw-r--  1 ec2-user ec2-user      0 Oct  5 19:34 out
[ec2-user@demohost ~]$
```

STDERR (2>) : In order to get the errors of any commands to file then we use STDERR redirector.

```
[ec2-user@demohost ~]$ ls -l nofile.txt
ls: cannot access nofile.txt: No such file or directory
[ec2-user@demohost ~]$ ls -l nofile.txt 2>err
[ec2-user@demohost ~]$ cat err
ls: cannot access nofile.txt: No such file or directory
[ec2-user@demohost ~]$
```

To redirect both STDOUT and STDERR to different files.

```
[ec2-user@demohost ~]$ ls -l file1.txt file2.txt  >out 2>err
[ec2-user@demohost ~]$ cat out
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
[ec2-user@demohost ~]$ cat err
ls: cannot access file2.txt: No such file or directory
[ec2-user@demohost ~]$
```

To redirect both STDOUT and STDERR to same file.

```
[ec2-user@demohost ~]$ ls -l file1.txt file2.txt  &>out
[ec2-user@demohost ~]$ cat out
ls: cannot access file2.txt: No such file or directory
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
[ec2-user@demohost ~]$
```

APPEND (>>): In order to append the content to file than overwriting then we use APPEND Redirector.

```
[ec2-user@demohost ~]$ cat out
total 0
drwxr-xr-x 14 root root 152 Oct  5 09:08 apache
[ec2-user@demohost ~]$ ls -l file1.txt >>out
[ec2-user@demohost ~]$ cat out
total 0
drwxr-xr-x 14 root root 152 Oct  5 09:08 apache
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
[ec2-user@demohost ~]$
```

Pipes (|): To send an output of command to another command as input then we use pipes.

```
[ec2-user@demohost ~]$ cat /etc/passwd | grep root
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[ec2-user@demohost ~]$
```

Command: **tee**

Command **tee** is use to redirect the output to a file and also same output will be printed on screen.

To redirect the output

```
[ec2-user@demohost ~]$ ls -l file1.txt |tee out
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
[ec2-user@demohost ~]$ cat out
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
[ec2-user@demohost ~]$
```

To append the output, use -a option

```
[ec2-user@demohost ~]$ ls -l file1.txt |tee -a out
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
[ec2-user@demohost ~]$ cat out
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
-rw-rw-rw-. 1 ec2-user ec2-user 1675 Oct  4 02:16 file1.txt
[ec2-user@demohost ~]$
```

Process Management:

Command: **top**

Command **top** is to see the process running on the server along with some more information.

```
top - 19:56:00 up 1 day, 10:33,  1 user,  load average: 0.00, 0.01, 0.05
Tasks: 116 total,   2 running, 114 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1015472 total,   178892 free,   106772 used,   729808 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   705688 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
    1 root      20   0   40948   3356   2288 S  0.0  0.3   0:05.43 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.00 ksoftirqd/0
```

Command: **ps**

Command **ps** is used to show all the process running, It is a snapshot of all the process running at the time you ran the command. The regular options which are used along with ps command is -e & -f combine.

To get all the process running on the system.

```
[ec2-user@demohost ~]$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Oct04 ?        00:00:05 /usr/lib/systemd/systemd --system --deserialize 20
root         2     0  0 Oct04 ?        00:00:00 [kthreadd]
root         3     2  0 Oct04 ?        00:00:00 [ksoftirqd/0]
root         6     2  0 Oct04 ?        00:00:00 [kworker/u30:0]
root         7     2  0 Oct04 ?        00:00:00 [migration/0]
```

Note : Above output is not complete, Just given few lines of output.

To check whether a specific process is running or not then.

```
[ec2-user@demohost ~]$ ps -ef |grep system
root         1     0  0 Oct04 ?        00:00:05 /usr/lib/systemd/systemd --system --deserialize 20
root       348     1  0 Oct04 ?        00:00:00 /usr/lib/systemd/systemd-journald
root       379     1  0 Oct04 ?        00:00:00 /usr/lib/systemd/systemd-udevd
dbus       472     1  0 Oct04 ?        00:00:01 /bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-ac
tivation
root       479     1  0 Oct04 ?        00:00:00 /usr/lib/systemd/systemd-logind
ec2-user 27540 27411  0 20:00 pts/0    00:00:00 grep --color=auto system
[ec2-user@demohost ~]$
```

Command: **kill**

Command **kill** is used to kill the running process. You should know the PID (Process ID) and parse the PID to kill command to kill the process.

To kill a process.

```
[ec2-user@demohost ~]$ ps -ef |grep sleep
ec2-user 27560 27411  0 20:02 pts/0    00:00:00 sleep 300
ec2-user 27562 27411  0 20:02 pts/0    00:00:00 grep --color=auto sleep
[ec2-user@demohost ~]$ kill 27560
[ec2-user@demohost ~]$ ps -ef  |grep sleep
ec2-user 27564 27411  0 20:03 pts/0    00:00:00 grep --color=auto sleep
[1]+  Terminated              sleep 300
[ec2-user@demohost ~]$
```

To kill a process forcefully.

```
[ec2-user@demohost ~]$ ps -ef |grep sleep
ec2-user 27566 27411  0 20:03 pts/0    00:00:00 sleep 300
ec2-user 27568 27411  0 20:03 pts/0    00:00:00 grep --color=auto sleep
[ec2-user@demohost ~]$ kill -9 27566
[ec2-user@demohost ~]$ ps -ef |grep sleep
ec2-user 27570 27411  0 20:04 pts/0    00:00:00 grep --color=auto sleep
[1]+  Killed                  sleep 300
[ec2-user@demohost ~]$
```

Command: **pkill**

Command **pkill** is used to kill the running process but with process name rather than with PID.

```
[ec2-user@demohost ~]$ ps -ef |grep sleep
ec2-user 27571 27411  0 20:05 pts/0    00:00:00 sleep 300
ec2-user 27572 27411  0 20:05 pts/0    00:00:00 sleep 200
ec2-user 27573 27411  0 20:05 pts/0    00:00:00 sleep 100
ec2-user 27575 27411  0 20:05 pts/0    00:00:00 grep --color=auto sleep
[ec2-user@demohost ~]$ pkill sleep
[1]   Terminated              sleep 300
[2]-  Terminated              sleep 200
[3]+  Terminated              sleep 100
[ec2-user@demohost ~]$ ps -ef |grep sleep
ec2-user 27578 27411  0 20:05 pts/0    00:00:00 grep --color=auto sleep
[ec2-user@demohost ~]$
```

Command: **jobs**

Command **jobs** is to list the background process.

To run a process in background then end the command with **&** symbol.

```
[ec2-user@demohost ~]$ sleep 300 &
[1] 27579
[ec2-user@demohost ~]$ jobs
[1]+  Running                 sleep 300 &
[ec2-user@demohost ~]$ sleep 200 &
[2] 27580
[ec2-user@demohost ~]$ jobs
[1]-  Running                 sleep 300 &
[2]+  Running                 sleep 200 &
[ec2-user@demohost ~]$
```

Package Management:

Command: **rpm**

Command **rpm** is used to install / update / remove / list rpm packages.

Option **-i** : To install a rpm.

```
[root@demohost ~]# ls
httpd-2.4.6-31.0.1.el7.x86_64.rpm  zip-3.0-10.el7.x86_64.rpm
[root@demohost ~]# rpm -ivh zip-3.0-10.el7.x86_64.rpm
warning: zip-3.0-10.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID ec551f03: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:zip-3.0-10.el7                   ################################# [100%]
[root@demohost ~]#
```

NOTE : To install a package with rpm command then the file should be there in server to install.

Dependency Issues with rpm.

```
[root@demohost ~]# ls
httpd-2.4.6-31.0.1.el7.x86_64.rpm  zip-3.0-10.el7.x86_64.rpm
[root@demohost ~]# rpm -ivh httpd-2.4.6-31.0.1.el7.x86_64.rpm
warning: httpd-2.4.6-31.0.1.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID ec551f03: NOKEY
error: Failed dependencies:
        /etc/mime.types is needed by httpd-2.4.6-31.0.1.el7.x86_64
        httpd-tools = 2.4.6-31.0.1.el7 is needed by httpd-2.4.6-31.0.1.el7.x86_64
        libapr-1.so.0()(64bit) is needed by httpd-2.4.6-31.0.1.el7.x86_64
        libaprutil-1.so.0()(64bit) is needed by httpd-2.4.6-31.0.1.el7.x86_64
[root@demohost ~]#
```

Get the remaining packages and then you install dependencies also along with main package then only it installs.

```
[root@demohost ~]# ls
apr-1.4.8-3.el7.x86_64.rpm          httpd-2.4.6-31.0.1.el7.x86_64.rpm       mailcap-2.1.41-2.el7.noarch.rpm
apr-util-1.5.2-6.0.1.el7.x86_64.rpm httpd-tools-2.4.6-31.0.1.el7.x86_64.rpm zip-3.0-10.el7.x86_64.rpm
[root@demohost ~]# rpm -ivh httpd-2.4.6-31.0.1.el7.x86_64.rpm apr-1.4.8-3.el7.x86_64.rpm apr-util-1.5.2-6.0.1.el7.x86_64.rpm h
ttpd-tools-2.4.6-31.0.1.el7.x86_64.rpm  mailcap-2.1.41-2.el7.noarch.rpm
warning: httpd-2.4.6-31.0.1.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID ec551f03: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:apr-1.4.8-3.el7                  ################################# [ 20%]
   2:apr-util-1.5.2-6.0.1.el7         ################################# [ 40%]
   3:httpd-tools-2.4.6-31.0.1.el7     ################################# [ 60%]
   4:mailcap-2.1.41-2.el7             ################################# [ 80%]
   5:httpd-2.4.6-31.0.1.el7           ################################# [100%]
[root@demohost ~]#
```

Option **-U** : To update a rpm package.

```
[root@demohost ~]# ls
httpd-2.4.6-40.0.1.el7_2.4.x86_64.rpm  httpd-tools-2.4.6-40.0.1.el7_2.4.x86_64.rpm
[root@demohost ~]# rpm -Uvh httpd-*
warning: httpd-2.4.6-40.0.1.el7_2.4.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID ec551f03: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:httpd-tools-2.4.6-40.0.1.el7_2.4 ################################# [ 25%]
   2:httpd-2.4.6-40.0.1.el7_2.4       ################################# [ 50%]
Cleaning up / removing...
   3:httpd-2.4.6-31.0.1.el7           ################################# [ 75%]
   4:httpd-tools-2.4.6-31.0.1.el7     ################################# [100%]
[root@demohost ~]#
```

Option **-e** : To erase the installed rpm. (To remove a package)

```
[root@demohost ~]# rpm -e zip
[root@demohost ~]# rpm -q zip
package zip is not installed
[root@demohost ~]#
```

Option **-q** : To check the installed packages. (To query the rpm database)

To check the installed package.

```
[root@demohost ~]# rpm -q httpd
httpd-2.4.6-40.0.1.el7_2.4.x86_64
[root@demohost ~]# rpm -qa |grep httpd
httpd-2.4.6-40.0.1.el7_2.4.x86_64
httpd-tools-2.4.6-40.0.1.el7_2.4.x86_64
[root@demohost ~]#
```

To check the information of a package.

```
[root@demohost ~]# rpm -qi httpd
Name        : httpd
Version     : 2.4.6
Release     : 40.0.1.el7_2.4
Architecture: x86_64
Install Date: Wed 05 Oct 2016 08:27:39 PM EDT
```

To check all the files associated with a rpm.

```
[root@demohost ~]# rpm -ql zip
/usr/bin/zip
/usr/bin/zipcloak
/usr/bin/zipnote
/usr/bin/zipsplit
/usr/share/doc/zip-3.0
/usr/share/doc/zip-3.0/CHANGES
```

Check a file, for which rpm it is associated with.

```
[root@demohost ~]# rpm -qf /usr/bin/zip
zip-3.0-10.el7.x86_64
[root@demohost ~]#
```

Command: **yum**

Command **yum** is used to install / update / remove / list rpm packages.

You should have a repository file in /etc/yum.repos.d directory to download and install the packages automatically.

You can download the repository file from the following URL.

```
## RHEL/CentOS 7 64-Bit ##
# wget http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-8.noarch.rpm
# rpm -ivh epel-release-7-8.noarch.rpm
```

General Yum Commands:

```
yum install httpd
yum install httpd -y
yum erase httpd -y
yum update httpd -y
yum update -y
yum list installed
yum list available
yum list all
yum provides "*bin/unzip"
yum repolist
```

Service Management:

Command: **systemctl**

Command **systemctl** is used to manage the installed services in RHEL 7 OS.

General systemctl commands

```
systemctl start httpd
systemctl stop httpd
systemctl restart httpd
systemctl status httpd
systemctl enable httpd
systemctl disable httpd
systemctl list-unit-files -a |grep httpd
```

Command: **service**

Command **service** is used to manage the installed services in RHEL 6 OS.

```
service httpd start
service httpd stop
service httpd status
service httpd restart
/etc/init.d/httpd start
/etc/init.d/httpd stop
/etc/init.d/httpd restart
/etc/init.d/httpd status
```

Command: **chkconfig**

Command **chkconfig** is used to manage the services during system boot.

```
chkconfig --list
chkconfig --list |grep httpd
chkconfig httpd on
chkconfig httpd off
```

User Management:

Command: **groupadd / groupmod / groupdel**

Command **groupadd** is used to add the group to the server.
Command **groupmod** is used to modify the existing group like name and ID.
Command **groupdel** is used to delete the existing group.

To add user.

```
[root@demohost ~]# groupadd admins
[root@demohost ~]# grep admins /etc/group
admins:x:1001:
[root@demohost ~]# getent group admins
admins:x:1001:
[root@demohost ~]#
```

To add group with specific Group ID (GID)

```
[root@demohost ~]# groupadd -g 2000 admins
[root@demohost ~]# getent group admins
admins:x:2000:
[root@demohost ~]# grep admins /etc/group
admins:x:2000:
[root@demohost ~]#
```

Associated commands.

```
groupdel admins
groupmod -n administrators admins
groupmod -g 3000 administrators
```

Command: **useradd / usermod / userdel**

Command **useradd** is used to add a user.
Command **usermod** is used to modify a user.
Command **userdel** is used to delete a user.

To add a user.

```
[root@demohost ~]# useradd john
[root@demohost ~]# id john
uid=1001(john) gid=1001(john) groups=1001(john)
[root@demohost ~]# getent passwd john
john:x:1001:1001::/home/john:/bin/bash
[root@demohost ~]# grep john /etc/passwd
john:x:1001:1001::/home/john:/bin/bash
[root@demohost ~]#
```

To add a user with specific group and specific UID.

```
[root@demohost ~]# useradd -u 2000 -g admins steve
[root@demohost ~]# id steve
uid=2000(steve) gid=2000(admins) groups=2000(admins)
[root@demohost ~]# getent passwd steve
steve:x:2000:2000::/home/steve:/bin/bash
[root@demohost ~]# grep steve /etc/passwd
steve:x:2000:2000::/home/steve:/bin/bash
[root@demohost ~]#
```

Other Associated commands.

```
userdel steve
userdel -rf steve
usermod -u 1050 steve
usermod -G dba steve
```

Command: **sudo**

Command **sudo** is to gain the root privileges as a normal user.

```
[ec2-user@demohost ~]$ sudo yum install gcc
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package gcc.x86_64 0:4.8.5-4.el7 will be installed
```

You can perform any task with root privilege but you should be granted privileges to do that.
To provide sudo privileges you have to be a root user and add the following lines to **`/etc/sudoers`**.

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
john    ALL=(ALL)       ALL
steve   ALL=(ALL)       NOPASSWD: ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Now john can perform all the privileges like root but with password every time, But steve can perform all the tasks as root but without password prompting.

Wheel group members also get the root privileges in the above example.

Command: **chmod**

Command **chmod** is used to change the permission of a file. Usually owner of the file can change the permissions of a file and also root user also can change the file permissions.

Permissions in Linux we have are **read, write, execute.** You can map these permissions to **Owner** of file, **Group Owner** of file and **Others** in the system.

You can see the permission of a file using ls -l command.

```
[ec2-user@demohost ~]$ ls -l
total 0
-rw-rw-rw- 1 ec2-user ec2-user 0 Oct  8 10:22 123
---------- 1 ec2-user ec2-user 0 Oct  8 10:22 abc
-r-------- 1 ec2-user ec2-user 0 Oct  8 10:22 file1.txt
[ec2-user@demohost ~]$
```

You can add read permission to group and other for a file file1.txt using the following command.

```
[ec2-user@demohost ~]$ ls -l file1.txt
-r-------- 1 ec2-user ec2-user 0 Oct  8 10:22 file1.txt
[ec2-user@demohost ~]$ chmod go+r file1.txt
[ec2-user@demohost ~]$ ls -l file1.txt
-r--r--r-- 1 ec2-user ec2-user 0 Oct  8 10:22 file1.txt
[ec2-user@demohost ~]$
```

+ operand is used to add the permission whereas **–** operand is used to remove the permission.
Also you can use number notation to add and remove the permissions.
Values for different permissions are.

Read = 4
Write = 2
Execute = 1

You can add the permission by numbers as shown.

```
[ec2-user@demohost ~]$ ls -l abc
---------- 1 ec2-user ec2-user 0 Oct  8 10:22 abc
[ec2-user@demohost ~]$ chmod 640 abc
[ec2-user@demohost ~]$ ls -l abc
-rw-r----- 1 ec2-user ec2-user 0 Oct  8 10:22 abc
[ec2-user@demohost ~]$
```

To change for a directory recursively same permissions then use -R option.

Command: **chown / chgrp**

Command **chown** is used to change the owner of a file. Command **chgrp** is used to change the group owner of the file Only root user can change the ownerships of a file or normal user with root privileges.

```
[root@demohost ec2-user]# ls -l file1.txt
-r--r--r-- 1 ec2-user ec2-user 0 Oct  8 10:22 file1.txt
[root@demohost ec2-user]# chown root file1.txt
[root@demohost ec2-user]# ls -l file1.txt
-r--r--r-- 1 root ec2-user 0 Oct  8 10:22 file1.txt
[root@demohost ec2-user]#
```

```
[root@demohost ec2-user]# ls -l file1.txt
-r--r--r-- 1 root ec2-user 0 Oct  8 10:22 file1.txt
[root@demohost ec2-user]# chgrp bin file1.txt
[root@demohost ec2-user]# ls -l file1.txt
-r--r--r-- 1 root bin 0 Oct  8 10:22 file1.txt
[root@demohost ec2-user]#
```

You can change both owner and group owner using **chown** command itself.

```
[root@demohost ec2-user]# ls -l file1.txt
-r--r--r-- 1 ec2-user ec2-user 0 Oct  8 10:22 file1.txt
[root@demohost ec2-user]# chown root:bin file1.txt
[root@demohost ec2-user]# ls -l file1.txt
-r--r--r-- 1 root bin 0 Oct  8 10:22 file1.txt
[root@demohost ec2-user]#
```

To Reboot Servers.

```
# init 6
# reboot
# shutdown -r
```

To Shutdown Servers.

```
# init 0
# halt
# shutdown -h
# poweroff
```

Login Scripts:

When user logs in there are some backend scripts which gets executed by default. If you would like to do that job every time you login then you can use such scripts.

These scripts are of two types, One is global which gets effected for all the users and other one is Local scripts which those task is limited to one particular user.

Global Scripts: `/etc/bashrc , /etc/profile , /etc/profile.d/*`

Local Scripts: `~/.bash_profile , ~/.bashrc`

For Example if you would like to add the JAVA_HOME as an environment variable to certain user then add the line to .bashrc file.

```
export JAVA_HOME=/opt/jdk1.8
```

What is Cloud Computing

- Cloud computing is a style of computing where scalable and elastic IT-enabled capabilities are delivered as a service to external customers using Internet technologies

Amazon in Cloud Computing

- Amazon has a long history of using a decentralized IT infrastructure.
- They decided to provide the idea/solution they has as a service.
- Amazon launched Amazon Web Services (AWS) so that other organizations could benefit from Amazon's experience and investment in running a large-scale distributed, transactional IT infrastructure.
- AWS has been operating since 2006, and today serves hundreds of thousands of customers worldwide. Today Amazon.com runs a global web platform serving millions of customers and managing billions of dollars' worth of commerce every year.

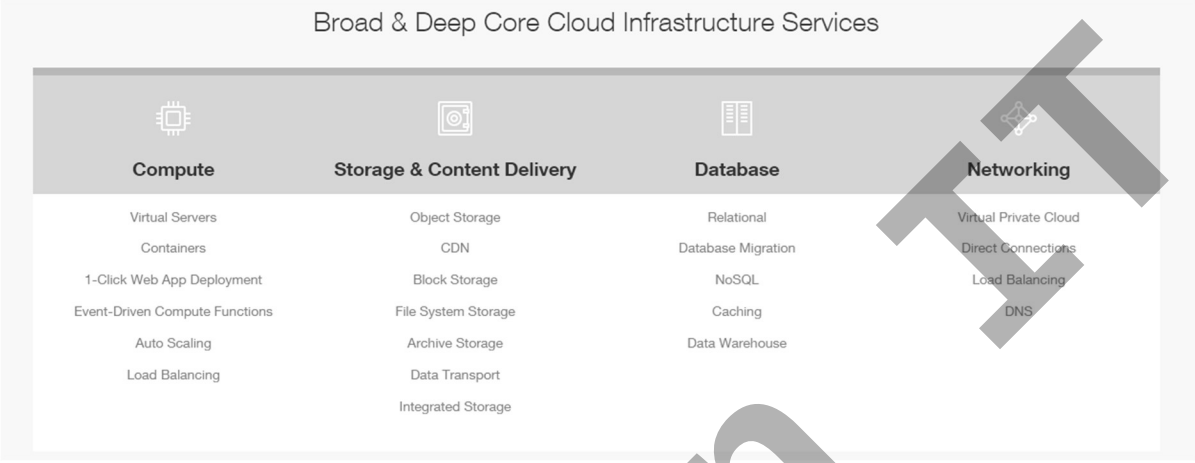
Benefits of AWS

- **Flexible**. AWS enables organizations to use the programming models, operating systems, databases, and architectures with which they are already familiar. In addition, this flexibility helps organizations mix and match architectures in order to serve their diverse business needs.
- **Cost-effective**. With AWS, organizations pay only for what they use, without up-front or long-term commitments.
- **Scalable and elastic**. Organizations can quickly add and subtract AWS resources to their applications in order to meet customer demand and manage costs.

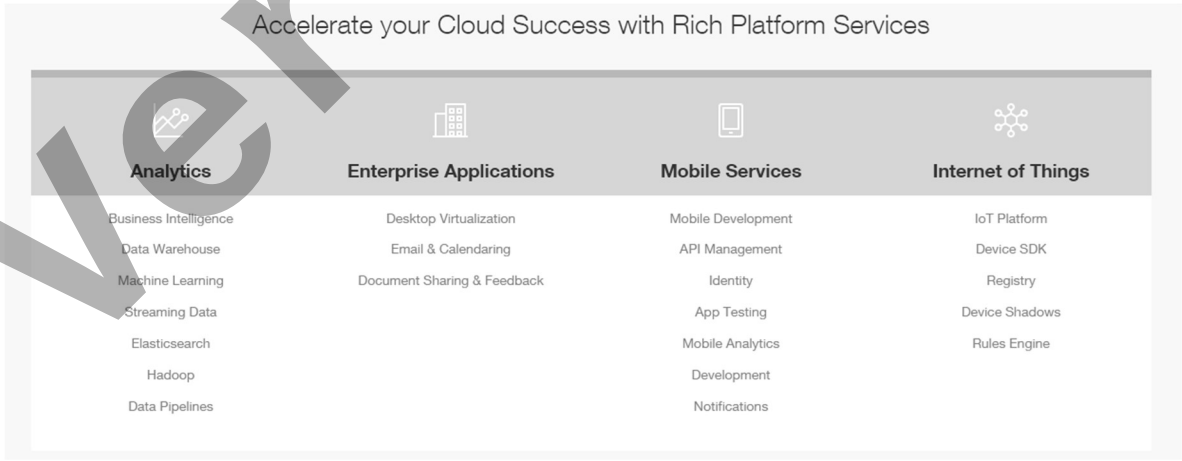
Benefits of AWS

- **Secure**. In order to provide end-to-end security and end-to-end privacy, AWS builds services in accordance with security best practices, provides the appropriate security features in those services, and documents how to use those features.
- **Experienced**. When using AWS, organizations can leverage Amazon's more than fifteen years of experience delivering large-scale, global infrastructure in a reliable, secure fashion. This is the major distinguish between the other providers with Amazon.

AWS : Cloud Services



AWS : Cloud Services

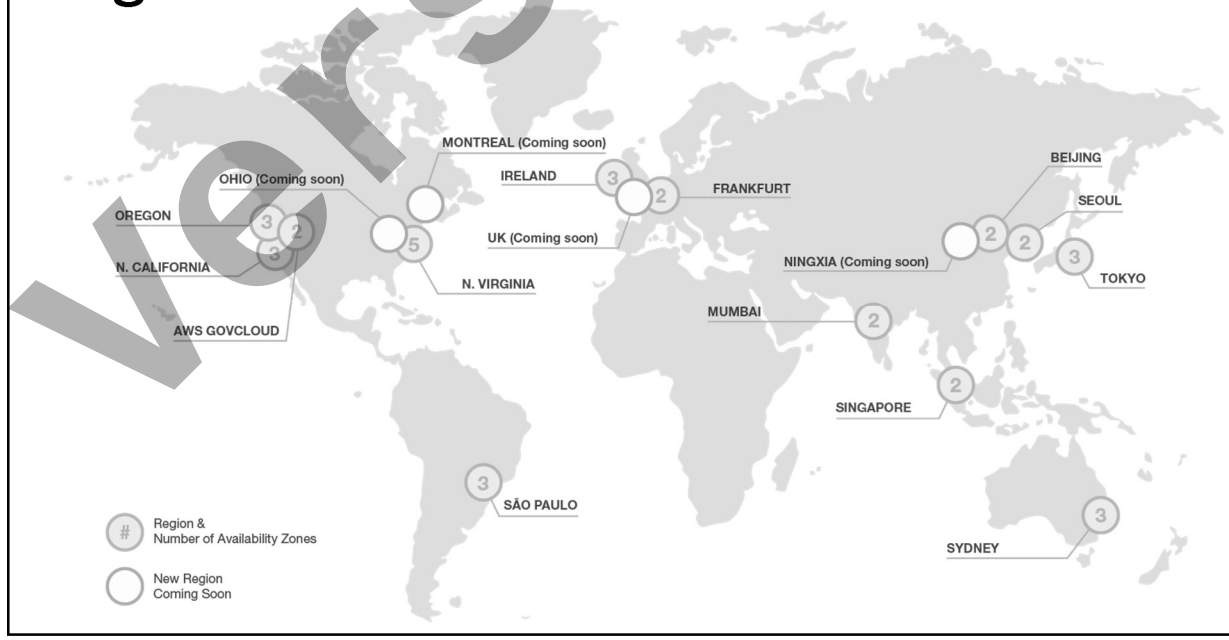


Create AWS Account

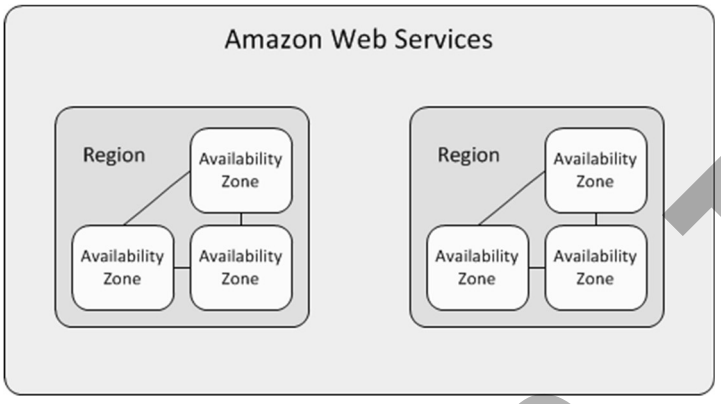
- Need Credit Card
- Provide valid email address
- Provide valid mobile number
- Provide PAN No if any
- Select Support Plan

- And there you go….

Regions



Availability Zone



EC2 instances: Families and Generations

- General-purpose : M1, M3 , T2
- Compute-optimized : C1, CC2, C3, C4
- Memory-optimized : M2, CR1, R3
- Dense-storage : HS1, D2
- I/O-optimized : HI1, I2
- GPU : CG1, G2
- Micro : T1, T2

EC2 instances: Types

Instance generation

c4.large

Instance family **Instance size**

Performance factors: CPU

Intel Xeon E5-2670 (Sandy Bridge) CPUs
- Available on M3, CC2, CR1, and G2 instance types

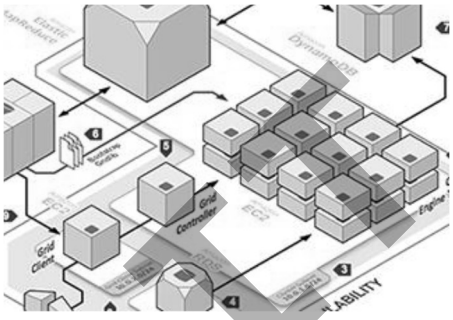
Intel Xeon E5-2680 v2 (Ivy Bridge) CPUs
- Available on C3, R3, and I2 instance types
- 2.8 GHz in C3, Turbo enabled up to 3.6 GHz
- Supports Enhanced Advanced Vector Extensions (AVX) instructions

Intel Xeon E5-2666 v3 (Haswell – AVX2) CPUs
- Available on C4 instance types
- 2.9 GHz in C4, Turbo enabled up to 3.5 GHz (with Intel Turbo Boost)
- Supports 256-bit integer vectors and can process 32 single-precision 16 or double-precision floating-point operations per cycle

Performance factors: Networks

AWS proprietary 10Gb networking

- Highest performance in .8xlarge instance sizes
- Full bi-section bandwidth in placement groups
- No network oversubscription



Enhanced Networking

- Available on D2, C3, C4, R3, I2 (in VPC with HVM)
- Over 1M PPS performance, reduced instance-to-instance latencies, more consistent performance

Performance factors: Storage

- Locally attached or “instance storage”
- Amazon EBS General Purpose (SSD) volumes
- Amazon EBS Provisioned IOPS (SSD) volumes
- Amazon EBS Magnetic volumes
- S3/Glacier

T2 and EC2 Free Tier instances

T2 is a Burstable Performance Instance

- Burstable Performance Instances provide a baseline level of CPU performance with the ability to burst above the baseline
- T2 instances are for workloads that don't use the full CPU often or consistently, but occasionally need to burst
- Great for getting started on EC2

EC2 Free Tier

- 750 hours of EC2 Linux t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support)

Instances

- EC2 Compute Instances.
- Spot Instances.
- Reserved Instances.
- Scheduled Instances.
- Dedicated Hosts.

Images

- AMI (Amazon Machine Image)
  - AWS Market Place
  - Community AMI's
  - My AMI's
- Bundle Tasks

Elastic Block Store

- Volumes
  - General Purpose SSD
  - Provisioned IOPS SSD
  - Throughput Optimized HDD
  - Cold HDD
  - EBS Magnetic
- Snapshots

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

Load Balancing

- Load Balancers.
- Target Groups.

Auto Scaling

- Launch Configurations
- Auto Scaling Groups

EC2

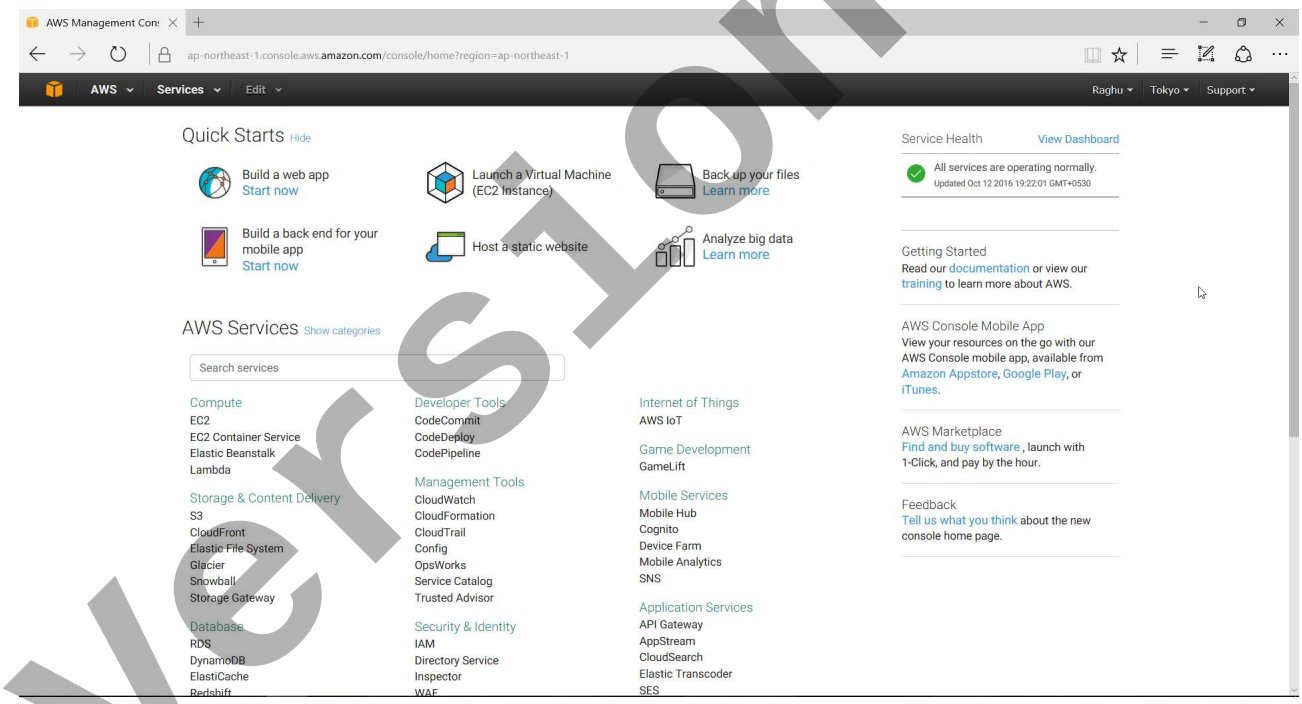
Creating a Instance:

Register your account using your credit card in the following URL.

https://aws.amazon.com/console/

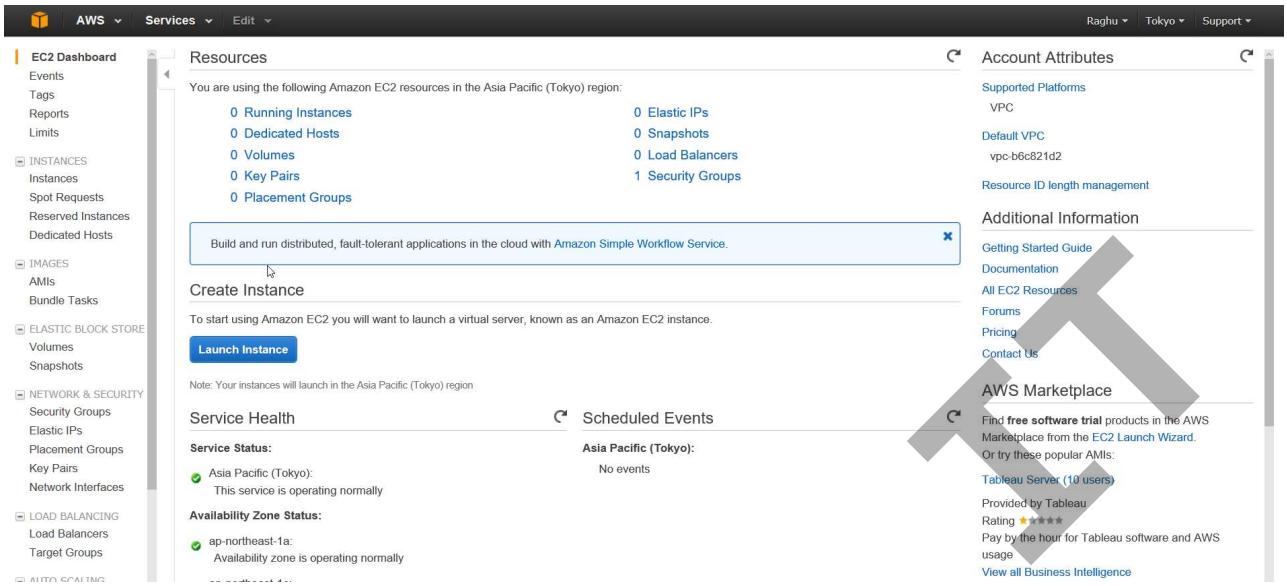
After registration login with the credentials in the above URL.

After you login you will see the dashboard as follows.



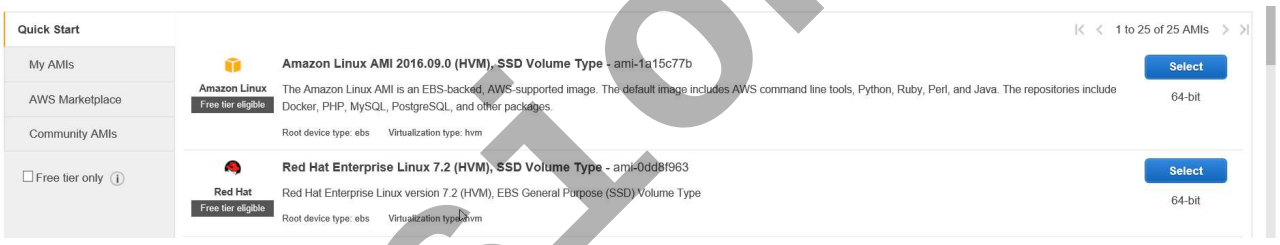
After that navigate to “**Compute**” and then click on “**EC2**”.

The it will redirect you to EC2 dashboard which looks like follows.



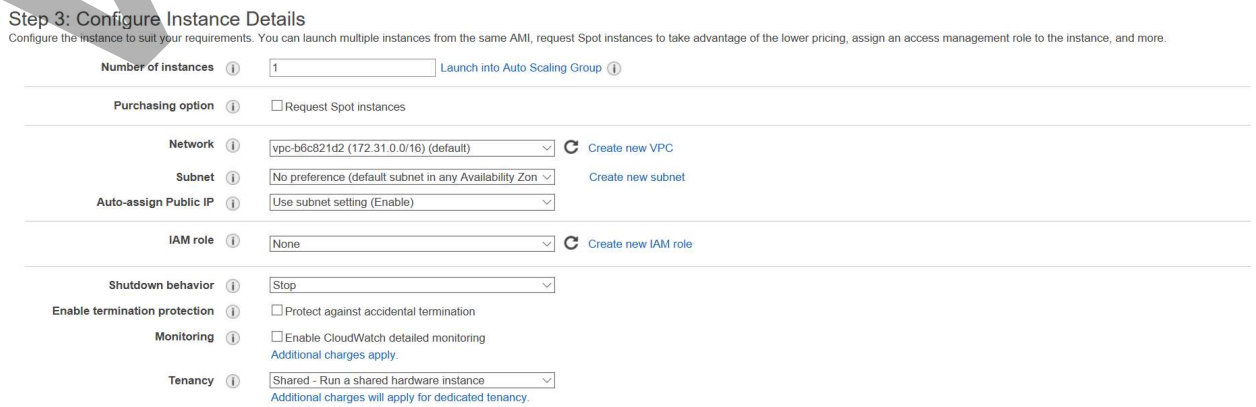
Click on **Launch Instance** which will take you to create an instance.

Select the required version of RedHat OS, Here I am selecting 7.2 instance.



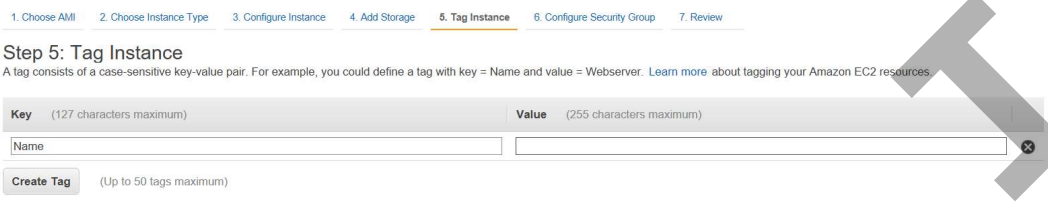
Select the Instance Type and then proceed.

Select the default values in this page and then proceed.

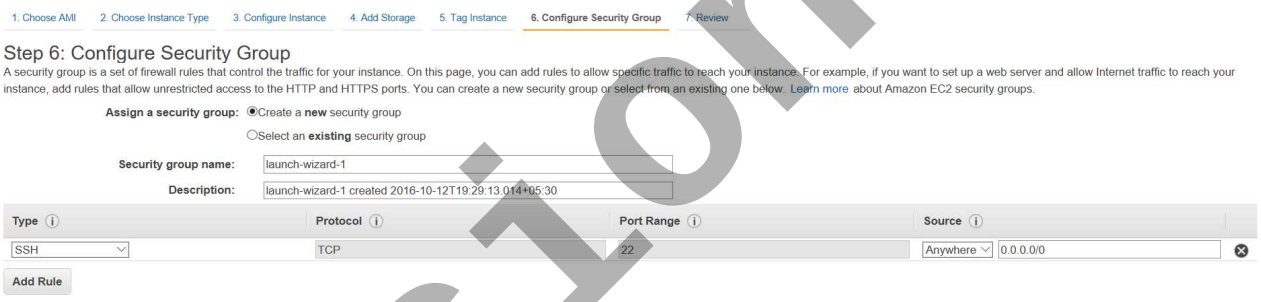


Select the required Storage and then proceed.

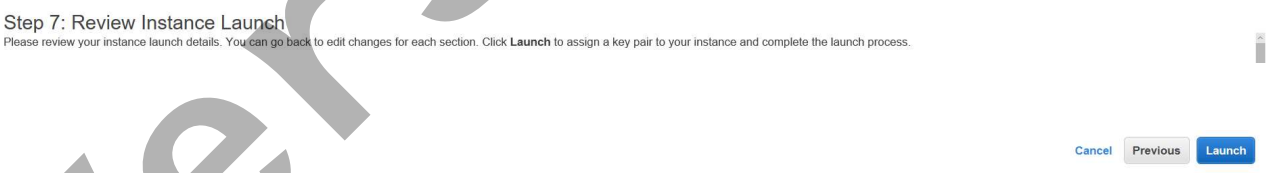
You can tag an instance or you can proceed without any value.



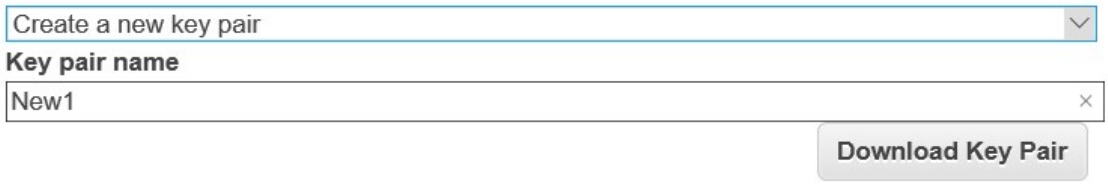
Select an existing Security Group or Select the default one to create a new for this.



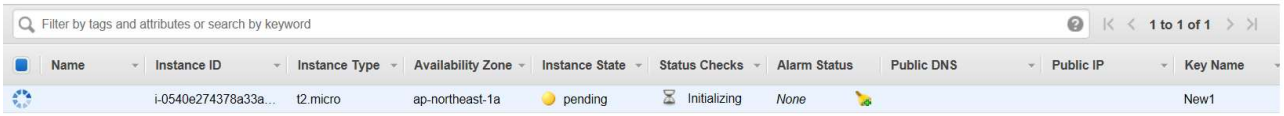
Review your selection and click on Launch.



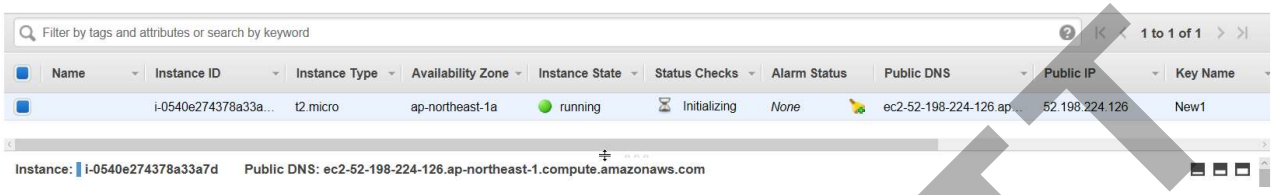
You can use a create a new Keypair and Download it to connect to server.



Now you can see a server getting launched.



You can connect to this instance using the downloaded .pem file to the public IP address assigned to instance.



You cannot use PEM file directly in windows to connect, But you can use that to connect from Linux machine directly.

Following is the command can be used to connect to the server from Linux CLI.

```
# ssh -i Desktop/New1.pem -l ec2-user ec2-52-198-224-126.ap-northeast-1.compute.amazonaws.com
The authenticity of host 'ec2-52-198-224-126.ap-northeast-1.compute.amazonaws.com (52.198.224.
126)' can't be established.
ECDSA key fingerprint is SHA256:iZdzMNcQDQiAFL0nIZf+lDN3U9gIQmw15q519YYSSNQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-52-198-224-126.ap-northeast-1.compute.amazonaws.com,52.198.224
.126' (ECDSA) to the list of known hosts.
[ec2-user@ip-172-31-20-41 ~]$ id
uid=1000(ec2-user) gid=1000(ec2-user) groups=1000(ec2-user),4(adm),10(wheel),190(systemd-journ
```

There you are.. ☺

What is Web Server?

A **web server** is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web.

HTTP: every web server program operates by accepting HTTP requests from the client, and providing an HTTP response to the client. The HTTP response usually consists of an HTML document, but can also be a raw file, an image, or some other type of document; if some error is found in client request or while trying to serve the request, a web server has to send an ***error response*** which may include some custom HTML or text messages to better explain the problem to end users.

General Features of Web Server

- **Logging**, usually web servers have also the capability of logging some detailed information, about client requests and server responses, to log files; this allows the webmaster to collect statistics by running log analyzers on log files.
- **Authentication**, optional authorization request (request of user name and password) before allowing access to some or all kind of resources.
- Handling of not only **static content** (file content recorded in server's filesystem(s)) but of **dynamic content** too by supporting one or more related interfaces (SSI, CGI, SCGI, FastCGI, JSP, PHP, ASP, ASP .NET, Server API such as NSAPI, ISAPI, etc.).

General Features of Web Server

- **HTTPS** support (by SSL or TLS) to allow secure (encrypted) connections to the server on the standard port 443 instead of usual port 80.
- **Content compression** (i.e. by gzip encoding) to reduce the size of the responses (to lower bandwidth usage, etc.).
- **Virtual hosting** to serve many web sites using one IP address.
- Large file support to be able to serve files whose **size is greater than** 2 GB on 32 bit OS.
- **Bandwidth throttling** to limit the speed of responses in order to not saturate the network and to be able to serve more clients.

Apache Web Server

- Apache Web Server is the most widely used Web Server application in the world with more than 50% share in the commercial web server market.
- Apache is the most widely used Web Server application in Unix-like operating systems but can be used on almost all platforms such as Windows, OS X, OS/2, etc.
- The word, Apache, has been taken from the name of the Native American tribe 'Apache', famous for its skills in warfare and strategy making.

Apache Web Server :: Installation

- From rpm packages.
- From Source.
- Difference between Installations.
- Different situations to use what.
- Apache HTTPD Directory Structure.

HTTPD Configuration file

- ServerRoot
- User
- Group
- DocumentRoot
- Listen
- Include
- VirtualHost

HTTP Request Methods

- **GET** : requests a resource
- **HEAD** : like GET request but without the response body
- **POST** : requests that the server accept the entity enclose in the request.
  - Might be a form
- **PUT** : requests that the enclose entity be store
- **DELETE** : deletes the resource

- **POST vs PUT**

HTTP Request Methods

- **TRACE** : echoes back the received request.
  - Usually for DEBUG.
- **OPTIONS** : returns the HTTP methods that the server supports
- **CONNECT** : Uses a proxy like communication tunnel.
  - for SSL
- **PATCH** : uses to apply a partial modification to resource

Tomcat Directory Structure

- /etc/tomcat
- /var/log/tomcat
- /var/lib/tomcat/webapps

- bin
- conf
- lib
- logs
- temp
- webapps
- work

Major Components of Tomcat

- **Catalina** - Servlet container name
- **Jasper** - JSP engine
- **Coyote** - HTTP connector
- **Cluster** - is load balancer to manage large scale application.

Tomcat Log Files

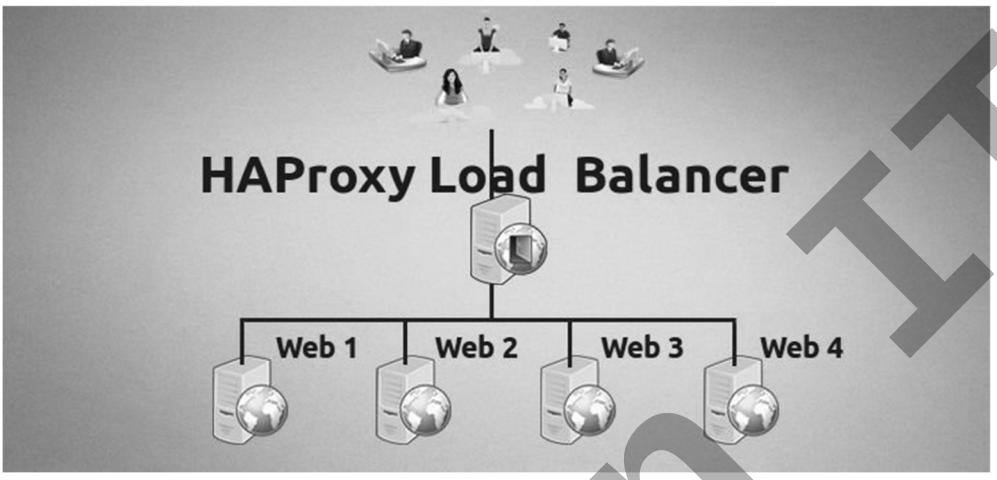
- **catalina.log** : It contains all server related information.
- **catalina.out**: It contains application specific log and server start-up and shutdown information.
- **localhost_access_log**: It contains server traffic or access info.
- **manager.log, host-manager.log** : Application related logs.

JAVA Parameters

-Xms and -Xmx

- These settings are used to define the size of the heap used by the JVM. -Xms defines the initial size of the heap and -Xmx defines the maximum size of the heap. Specific values for these options will depend on the number of applications and the requirements of each application deployed to a Tomcat instance.
- With regard to Tomcat, it is recommended that the initial and maximum values for heap size be set to the same value. This is often referred to as a fully committed heap and this will instruct the JVM to create a heap that is initially at its maximum size and prevent several full garbage collections from occurring as the heap expands to its maximum size.

HAProxy



Apache Web Server

Basic Installation

Installation of apache web server in EC2 instance is simple using yum command.

RedHat Linux Version using across this is RHEL7

Install the package using following command.

```
# yum install httpd -y
```

Verify whether the installation is happened properly or not using

```
# rpm -qa |grep httpd
```

Start the httpd service.

```
# systemctl start httpd
```

Enable the httpd service to start automatically while reboot.

```
# systemctl enable httpd  (RHEL 6 : chkconfig httpd on)
```

Stop the httpd service

```
# systemctl stop httpd
```

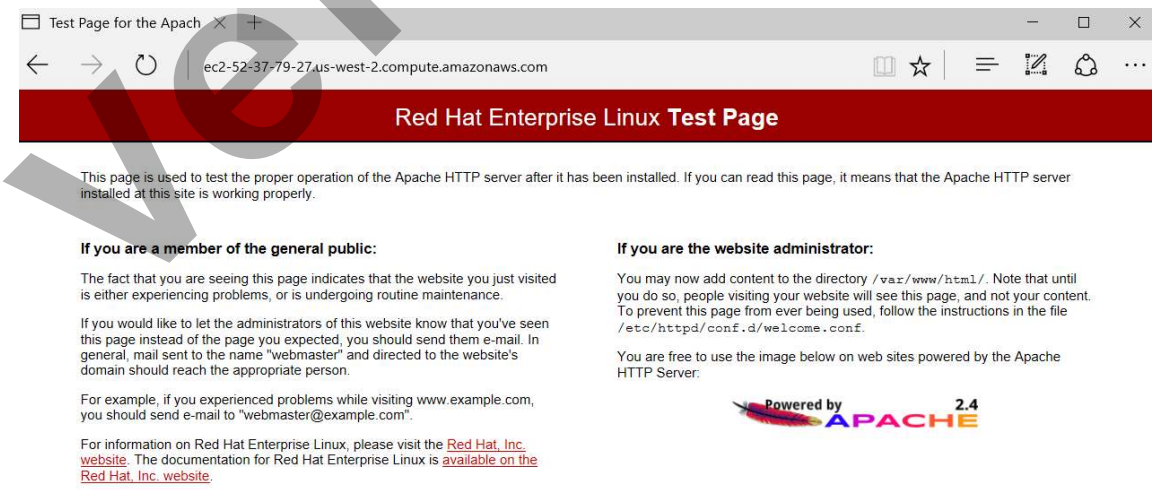
Disable the httpd service to not start during reboot.

```
# systemctl disable httpd
```

Test the connection.

```
# curl http://localhost
```

You can test from browser remotely.



Create a sample html file.

```
# cat /var/www/html/index.html
<h1>This is a Test Page</h1>
```

Access this page over browser.

Deploy sample website in Web Server:

```
# wget https://codeload.github.com/versionit/webcontent/zip/master -O /tmp/master.zip
# cd /tmp ; unzip master.zip
# cd /var/www/html
# tar xzf /tmp/webcontent-master/ecommerce.tgz
```

Access the page over browser.

Deploy one more project:

```
# cd /var/www/html
# tar xzf /tmp/webcontent-master/ers.tgz
```

Access the page over browser.

Create Name Virtual Hosting.

```
# cat /etc/httpd/conf.d/virt.conf

<VirtualHost *:80>
    DocumentRoot "/var/www/html/ERS/"
    ServerName ers.awstrainings.com

    # Other directives here
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "/var/www/html/ecommerce/"
    ServerName ecommerce.awstrainings.com

    # Other directives here
</VirtualHost>
```

Restart the service.

```
# systemctl restart httpd
```

Add the hosts entry in your Windows/Linux desktop and then check the URL's over browser.

```
52.43.183.209 ers.awstrainings.com
52.43.183.209 ecommerce.awstrainings.com
```

SSL Configuration on Apache:

Install the required packages.

```
# yum install mod_ssl
```

Create a New Certificate

```
# cd /etc/httpd/conf.d
# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/httpd/conf.d/apache.key -out /etc/httpd/conf.d/apache.crt
```

Just fill in the details or else you can press <ENTER> for all to continue.

Add / Modify the following lines in **/etc/httpd/conf.d/ssl.conf**

```
ServerName ip-172-31-43-159.us-west-2.compute.internal
SSLCertificateFile /etc/httpd/conf.d/apache.crt
SSLCertificateKeyFile /etc/httpd/conf.d/apache.key
```

After that restart the service.

```
# systemctl restart httpd
```

Confirm the port was opened or not.

```
# netstat -antp |grep http |grep LISTEN
```

Check the SSL Connection over browser.

Apache Tomcat Server

Installation using Source:

Download the tomcat server from Internet.

```
# cd /opt
# wget http://mirror.fibergrid.in/apache/tomcat/tomcat-8/v8.5.5/bin/apache-tomcat-8.5.5.tar.gz
```

Extract the package.

```
# tar -xzf apache-tomcat-8.5.5.tar.gz
```

Start the apache tomcat.

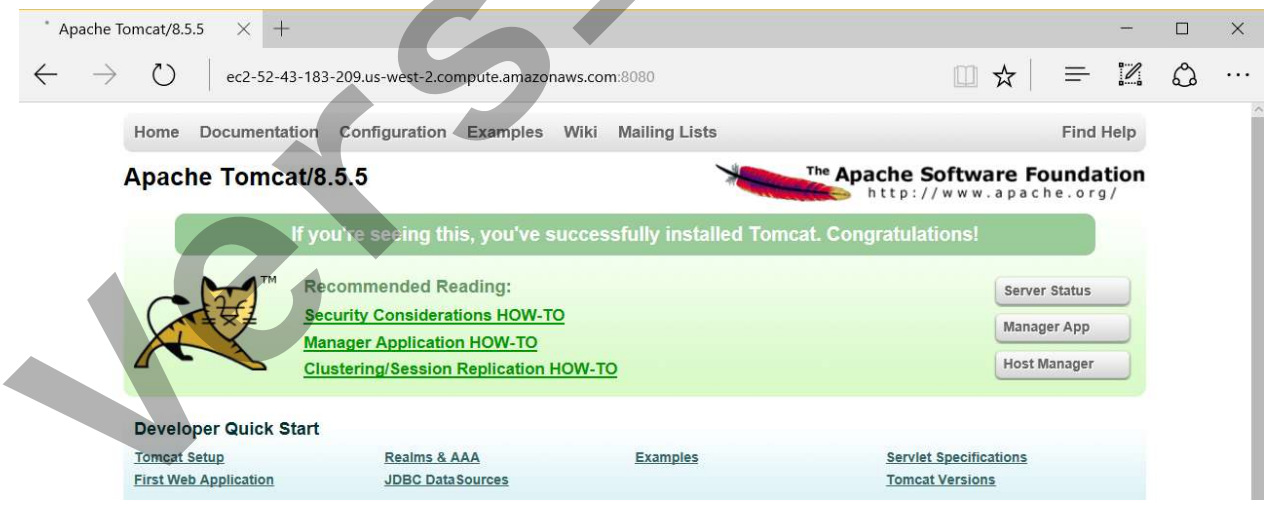
Unless you have java installed, Tomcat will not run. So install java and then start the tomcat. In case if Java is already installed then start it directly

```
# yum install java -y
# /opt/apache-tomcat-8.5.5/bin/startup.sh
```

Check whether it has started properly or not.

```
# netstat -antp |grep java
# ps -ef |grep java
```

It listens on port 8080 by default, so check from browser.



Installation using yum repo:

Install Java, Before tomcat as a pre-requisite.

Using this command, you are going to install java 1.8 version.

```
# yum install java -y
```

Now you can install tomcat

```
# yum install -y tomcat
```

Now we can start tomcat service as

```
# systemctl start tomcat
```

Once you started your service please check with **ps** command.

```
# ps -ef | grep -i tomcat
```

Check for the port number, on default tomcat service would be 8080.

```
# netstat -tulnp | grep -i java
```

Deploying application:

For default path for deploy application on tomcat would be :

```
# cd /var/lib/tomcat/webapps
```

So here we need to copy war/ear or any j2ee related app.

Please install **wget** first *(if not installed)* using yum, it is useful to download content from internet.

```
# yum install wget -y
```

Now please run below command to download war.

```
# wget https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/sample.war
```

Now change ownership to **sample.war**

```
# chown -R tomcat.tomcat sample.war
```

Please stop tomcat service.

```
# systemctl stop tomcat
```

Now start tomcat service

```
# systemctl start tomcat
```

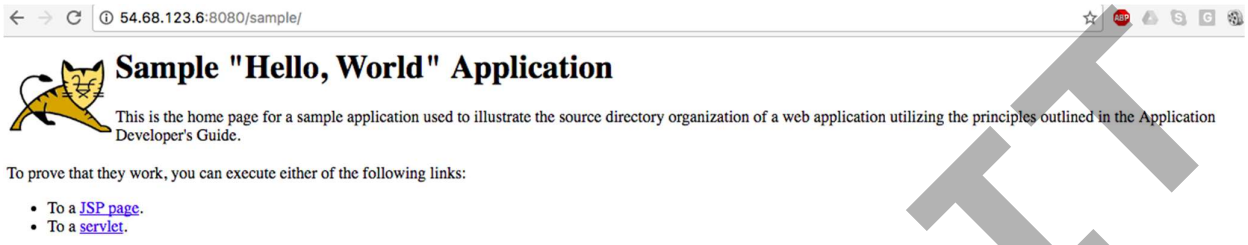
Now you can see **sample.war** is going to exploded to sample directory.

```
# cd /var/lib/tomcat/webapps
# ls
```

Now you can check is deployed app working or not.

```
# curl http://localhost:8080/sample/index.html
```

Also you can check the deployed application via browser.



Now if you want to deploy app with root access (without providing “sample” as a context path).

In that case first you need to rename exploded sample directory to ROOT.

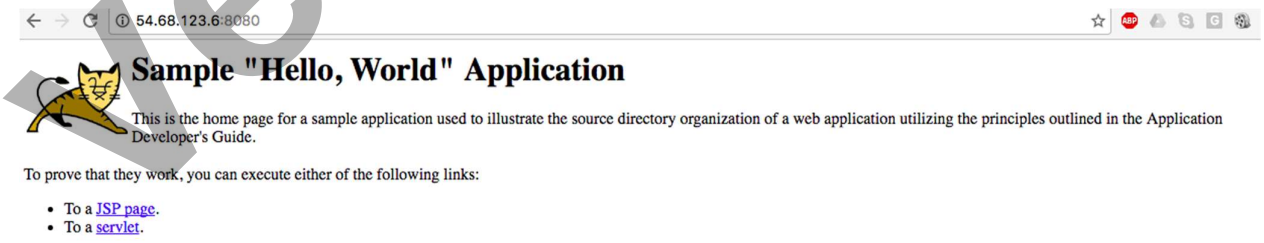
And restart the tomcat service.

```
# cd /var/lib/tomcat/webapps
# mv sample ROOT
```

Restart the Tomcat Service.

```
# systemctl stop tomcat
# systemctl start tomcat
```

Now check the same via browser, Without providing /sample in URL it got worked.



To start tomcat service automatically during reboot.

```
# systemctl enable tomcat
```

To disable automatically while reboot.

```
# systemctl disable tomcat
```

Change JAVA Arguments:

Open file **/etc/tomcat/tomcat.conf**

Add the following content to end of file.

```
JAVA_OPTS="-Xms512m -Xmx512m"
```

Then restart the service to load the changes you have made.

```
# systemctl restart tomcat
```

Check the changes you have made

```
# ps -ef | grep tomcat
```

Change Default Port:

Open the file **/etc/tomcat/server.xml** and change the required port number in the block.

Change From:

```
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

Change TO

```
    <Connector port="8081" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8081" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

After that restart the service.

```
# systemctl restart tomcat
```

Verify the port number working with **ps** command.

```
# ps -ef |grep tomcat
```

Verify theport number using **netstat** command.

```
# netstat -antp |grep LISTEN |grep java
```

HAProxy as Proxy Server:

Install haproxy package.

```
# yum install haproxy -y
```

First move the existing

configuration file and take backup

```
# cd /etc/haproxy
```

```
# mv haproxy.cfg haproxy.cfg.bkp
```

Now create the haproxy.cfg with following content.

```
# cat /etc/haproxy/haproxy.cfg
global
        daemon
        maxconn 256

    defaults
        mode http
        timeout connect 5000ms
        timeout client 50000ms
        timeout server 50000ms

    frontend http-in
        bind *:80
        default_backend servers

    backend servers
        server server1 127.0.0.1:8080 maxconn 32
```

Then start the service.

systemctl start haproxy

Test this over browser.

HAProxy as Load Balancer:

Install haproxy package.

```
# yum install haproxy -y
```

First move the existing configuration file and take backup

```
# cd /etc/haproxy
```

```
# mv haproxy.cfg haproxy.cfg.bkp
```

Now create the haproxy.cfg with following content.

```
# cat /etc/haproxy/haproxy.cfg
global
        daemon
        maxconn 256

    defaults
        mode http
        timeout connect 5000ms
        timeout client 50000ms
        timeout server 50000ms

    frontend http-in
        bind *:80
        balance roundrobin
        default_backend servers

    backend servers
        server server1 54.70.16.58:8080 check maxconn 32
        server server2 52.43.192.190:8080 check maxconn 32
```

No restart the service

```
# systemctl restart haproxy
```

Check the service working or not over browser.

So you can test whether haproxy working as load balancer or not by bringing down one server and check still the URL is able to be accessed or not.

What is Version Control System?

Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.

Following are the goals of a Version Control System.

- Allow developers to work simultaneously.
- Do not overwrite each other's changes.
- Maintain history of every version of everything.

A VCS is divided into two categories.

- Centralized Version Control System (CVCS), and
- Distributed/Decentralized Version Control System (DVCS).

Apache SubVersion (SVN)

- Open Source
- Simplicity
- Centralized
- Best IDE integration with SVN
- Full version history
- Locking support
- Built-in Authentication

Version Control Terminologies

- **Repository:** A repository is the heart of any version control system. It is the central place where developers store all their work. Repository not only stores files but also the history. Repository is accessed over a network, acting as a server and version control tool acting as a client. Clients can connect to the repository, and then they can store/retrieve their changes to/from repository. By storing changes, a client makes these changes available to other people and by retrieving changes, a client takes other people's changes as a working copy.

- **Trunk:** The trunk is a directory where all the main development happens and is usually checked out by developers to work on the project.

Version Control Terminologies

- **Tags** : The tags directory is used to store named snapshots of the project. Tag operation allows to give descriptive and memorable names to specific version in the repository.

For example, LAST_STABLE_CODE_BEFORE_EMAIL_SUPPORT is more memorable than

Repository UUID: 7ceef8cb-3799-40dd-a067-c216ec2e5247 and

Revision: 13

Version Control Terminologies

- **Branches:** Branch operation is used to create another line of development. It is useful when you want your development process to fork off into two different directions. For example, when you release version 5.0, you might want to create a branch so that development of 6.0 features can be kept separate from 5.0 bug-fixes.

- **Working copy:** Working copy is a snapshot of the repository. The repository is shared by all the teams, but people do not modify it directly. Instead each developer checks out the working copy. The working copy is a private workplace where developers can do their work remaining isolated from the rest of the team.

Version Control Terminologies

- **Commit changes:** Commit is a process of storing changes from private workplace to central server. After commit, changes are made available to all the team. Other developers can retrieve these changes by updating their working copy. Commit is an atomic operation. Either the whole commit succeeds or is rolled back. Users never see half finished commit.

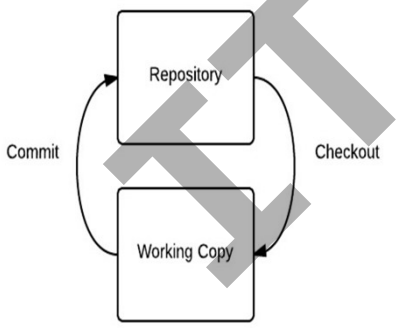
2 – Tree Architecture (SVN)

Working copy is the place where you make your changes. Whenever you edit something, it is saved in working copy and it is a physically stored in a disk.

Repository is the place where all the version of the files or commits, logs etc is stored. It is also saved in a disk and has its own set of files.

Checking-out is the process of getting files from repository to your working copy. This is because you can only edit files when it is on your working copy. When you are done editing the file, you will save it back to the repository by ***committing*** it.

Committing is the process of putting back the files from working copy to repository.



SVN Status Codes

U	Working file was updated
G	Changes on the repo were automatically merged into the working copy
M	Working copy is modified
C	This file conflicts with the version in the repo
?	This file is not under version control
!	This file is under version control but is missing or incomplete
A	This file will be added to version control (after commit)
A+	This file will be moved (after commit)
D	This file will be deleted (after commit)
S	This signifies that the file or directory has been switched from the path of the rest of the working copy (using svn switch) to a branch
I	Ignored
X	External definition
~	Type changed
R	Item has been replaced in your working copy. This means the file was scheduled for deletion, and then a new file with the same name was scheduled for addition in its place.
L	Item is locked
E	Item existed, as it would have been created, by an svn update.

Apache Subversion

Installing Sub-Version:

Install the package using yum command.

```
# yum install subversion -y
```

Check the installed subversion.

```
# svn --version
```

SVN needs web server to serve the repository. So install the webserver & svn web server module.

```
# yum install mod_dav_svn -y
```

Check the HTTPD SVN Module configuration files and it should look like follows.

```
# cat /etc/httpd/conf.modules.d/10-subversion.conf
```

```
LoadModule dav_svn_module     modules/mod_dav_svn.so
LoadModule authz_svn_module   modules/mod_authz_svn.so
LoadModule dontdothat_module  modules/mod_dontdothat.so
```

Create the SVN configuration file in HTTPD.

```
# vim /etc/httpd/conf.d/subversion.conf
```

```
<Location /svn>
   DAV svn
   SVNParentPath /var/www/svn
   AuthType Basic
   AuthName "Authorization Realm"
   AuthUserFile /etc/svn-users
   Require valid-user
</Location>
```

Create httpd users.

```
# htpasswd -cm /etc/svn-users tom
```

```
# htpasswd -m /etc/svn-users jerry
```

Create required directories.

```
# mkdir /var/www/svn
```

Start / Restart the webserver service.

```
# systemctl restart httpd
```

Enable the httpd service.

```
# systemctl enable httpd
```

Create a new project in SVN.

Start creating a project in SVN.

```
# cd /var/www/svn/

# svnadmin create project_repo
```

Change the ownership of the files in that project with apache user as we are going to access these files over http server.

```
# chown -R apache.apache project_repo/
```

Restart the web service now.

```
# systemctl restart httpd
```

To restrict the SVN repository to only the authorized users then modify the repository configuration.

```
# vim /var/www/svn/project_repo/conf/svnserve.conf

anon-access = none
authz-db = authz
```

You can create the SVN directory structure and import it.

```
# mkdir -p /tmp/svn-template/{trunk,branches,tags}
```

Import the directory structure in to repository.

```
# svn import /tmp/svn-template/ file:///var/www/svn/project_repo/ -m 'create
trunk,branches and tags directory structure'
```

Start using the SVN Repository from Client side.

Checkout the code from client side.

```
# svn checkout http://svn.server.com/svn/project_repo --username=tom
```

You can see the following output.

```
A    project_repo/trunk
A    project_repo/branches
A    project_repo/tags
Checked out revision 1.
```

Goto trunk directory and run the following command to see the project information.

```
# svn info
```

Add new files from Client.

Checkout the latest repository from SVN server.

```
# svn checkout http://svn.server.com/svn/project_repo --username=tom
```

Check the folder that created and repository under it.

```
[tom@ip-172-31-28-154 ~]$ ls
project_repo
[tom@ip-172-31-28-154 ~]$
```

Check the files created under trunk directory.

```
[tom@ip-172-31-28-154 ~]$ cd project_repo/trunk/
[tom@ip-172-31-28-154 trunk]$ ls
[tom@ip-172-31-28-154 trunk]$
```

Create the following file under it.

```
[tom@ip-172-31-28-154 trunk]$ vim index.html
<h1>
Hello
</h1>
```

Check the SVN Status.

```
[tom@ip-172-31-28-154 trunk]$ svn status
?       index.html
```

? mean that file is unknown to SVN. You can add that file to SVN using the following command.

```
[tom@ip-172-31-28-154 trunk]$ svn add index.html
A         index.html
[tom@ip-172-31-28-154 trunk]$ svn status
A       index.html
[tom@ip-172-31-28-154 trunk]$
```

A means that file is added.

You can commit the changes / additions to repository using the following command.

```
[tom@ip-172-31-28-154 trunk]$ svn commit -m 'Adding my first file'
Adding         index.html
Transmitting file data .
Committed revision 2.

Warning: post commit FS processing had error:
sqlite: attempt to write a readonly database
[tom@ip-172-31-28-154 trunk]$
```

You can see some error popped up due to permission issue, You can fix that by running the following command on repository.

```
# chown apache:apache /var/www/svn -R
```

Dealing with Conflicts.

Run the following commands as jerry user.

```
[jerry@ip-172-31-28-154 ~]$ svn checkout http://localhost/svn/project_repo --
username=jerry
A    project_repo/tags
A    project_repo/trunk
A    project_repo/trunk/index.html
A    project_repo/branches
Checked out revision 2.
[jerry@ip-172-31-28-154 ~]$
```

Try to modify the newly created file which was added by tom user.

```
[jerry@ip-172-31-28-154 trunk]$ cat index.html
<h1>
Hello World
</h1>
[jerry@ip-172-31-28-154 trunk]$
Check the SVN status now.
[jerry@ip-172-31-28-154 trunk]$ svn status
M       index.html
[jerry@ip-172-31-28-154 trunk]$
```

M means modified, Means that file is modified and try to commit the changes.

```
[jerry@ip-172-31-28-154 trunk]$ svn commit -m 'Adding new file'
Sending        index.html
Transmitting file data .
Committed revision 3.
```

Now go back to tom user and try to modify the file.

```
[tom@ip-172-31-28-154 trunk]$ cat index.html
<h1>
Hello
<h2> New World </h2>
</h1>
```

Then try to commit the code for which you recives and error.

```
[tom@ip-172-31-28-154 trunk]$ svn commit -m 'Added one more line'
Sending        index.html
Transmitting file data .svn: E155011: Commit failed (details follow):
svn: E155011: File '/home/tom/project_repo/trunk/index.html' is out of date
svn: E170004: Item '/trunk/index.html' is out of date
[tom@ip-172-31-28-154 trunk]$
```

You can pull the new code before you make change.

```
[tom@ip-172-31-28-154 trunk]$ svn update
Updating '.':
G    index.html
Updated to revision 3.
[tom@ip-172-31-28-154 trunk]$
```

G: Changes on the repo were automatically merged into the working copy

Now if you check the file both the lines are merged into your code.

```
[tom@ip-172-31-28-154 trunk]$ cat index.html
<h1>
Hello World
<h2> New World </h2>
</h1>
```

Now try to commit them.

```
[tom@ip-172-31-28-154 trunk]$ svn commit -m 'Adding the merged code'
Sending        index.html
Transmitting file data .
Committed revision 4.
```

Now as a jerry user, try to change the code again.

```
[jerry@ip-172-31-28-154 trunk]$ cat index.html
<h1>
Hello World
Hello Universe
</h1>
```

Update the code and you can see the conflicts.

```
[jerry@ip-172-31-28-154 trunk]$ svn update
Updating '.':
Conflict discovered in '/home/jerry/project_repo/trunk/index.html'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```

Create Tags.

Creating tags can be done by the following command.

```
[tom@ip-172-31-28-154 project_repo]$ svn copy --revision=4 trunk/
tags/My_First_WebSite
Updating 'tags/My_First_WebSite':
A    tags/My_First_WebSite/index.html
Updated to revision 4.
A         tags/My_First_WebSite
[tom@ip-172-31-28-154 project_repo]$
```

Check the list of files that got created as tag.

```
[tom@ip-172-31-28-154 project_repo]$ ls -l tags/
total 0
drwxrwxr-x 2 tom tom 23 Sep 26 14:04 My_First_WebSite
[tom@ip-172-31-28-154 project_repo]$ ls -l tags/My_First_WebSite/
total 4
-rw-rw-r-- 1 tom tom 44 Sep 26 14:04 index.html
[tom@ip-172-31-28-154 project_repo]$
```

Check the SVN status and commit the changes.

```
[tom@ip-172-31-28-154 project_repo]$ svn status
A  +    tags/My_First_WebSite
[tom@ip-172-31-28-154 project_repo]$ svn commit -m 'Adding new tag'
Adding         tags/My_First_WebSite

Committed revision 5.
[tom@ip-172-31-28-154 project_repo]$
```

Create Branches.

Creating branches is same as creating tags using the following command.

```
[jerry@ip-172-31-28-154 project_repo]$ svn copy trunk branches/jerry_branch
A         branches/jerry_branch
[jerry@ip-172-31-28-154 project_repo]$
```

Check the status.

```
[jerry@ip-172-31-28-154 project_repo]$ svn status
A  +    branches/jerry_branch
```

Commit the new changes to repository.

```
[jerry@ip-172-31-28-154 project_repo]$ svn commit -m "Jerry's private branch"
Adding       branches/jerry_branch

Committed revision 7.
```

Verify the branch file.

```
[jerry@ip-172-31-28-154 project_repo]$ cat branches/jerry_branch/index.html
<h1>
Hello World
Hello Universe
</h1>
```

What is Version Control System?

Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.

Following are the goals of a Version Control System.

- Allow developers to work simultaneously.
- Do not overwrite each other's changes.
- Maintain history of every version of everything.

A VCS is divided into two categories.

- Centralized Version Control System (CVCS), and
- Distributed/Decentralized Version Control System (DVCS).

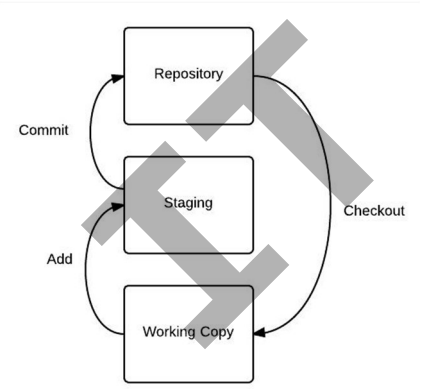
GIT

- Branching and Merging
  - Frictionless Context Switching
  - Role-Based Code lines
  - Feature Based Workflow
  - Disposable Experimentation
- Small and Fast
- Distributed
- Data Assurance
- Staging Area
- Free and Open Source

3 – Tree Architecture (GIT)

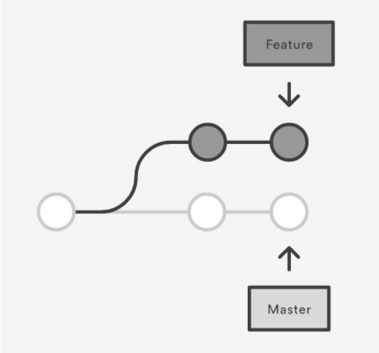
This is one of the fundamental difference of Git that sets it apart from other VCS, this **Staging Area** is a place where you prepare all the things that you are going to commit.

In Git, you don't move things directly from your working copy to the repository, you have to stage them first.



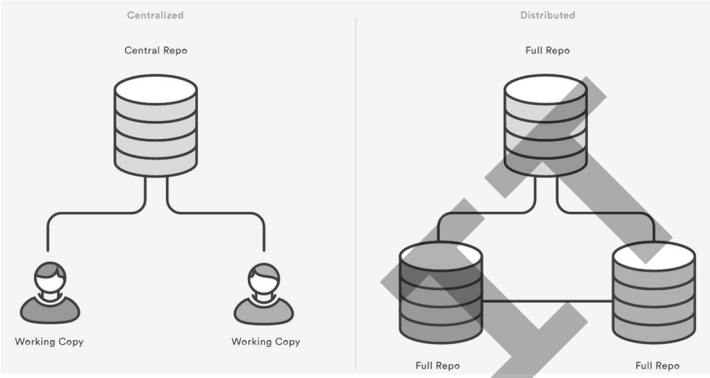
Feature Branch Workflow

- One of the biggest advantages of Git is its branching capabilities. Unlike centralized version control systems, Git branches are cheap and easy to merge. This facilitates the feature branch workflow popular with many Git users.
- Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something—no matter how big or small—they create a new branch. This ensures that the master branch always contains production-quality code.



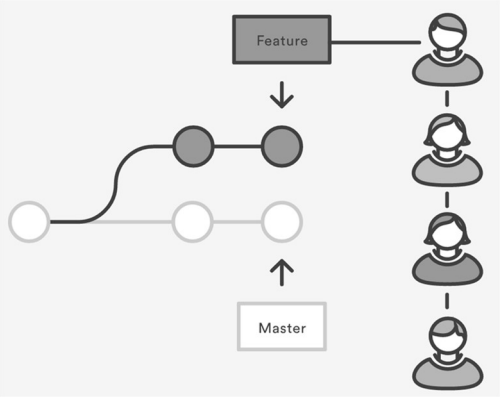
Distributed Development

- Git fast, since it means you don't need a network connection to create commits, inspect previous versions of a file, or perform diffs between commits.
- And, similar to feature branches, distributed development creates a more reliable environment. Even if a developer obliterates their own repository, they can simply clone someone else's and start anew.



Pull Requests

- A pull request is a way to ask another developer to merge one of your branches into their repository. This not only makes it easier for project leads to keep track of changes, but also lets developers initiate discussions around their work before integrating it with the rest of the codebase.



GIT

Installing GIT:

Install the package using yum command.

```
# yum install git -y
```

Check the installed git version.

```
# git --version
```

Setting up username and email address, GIT is going to use this information while committing the changes to repository.

```
# git config --global user.name rwnair0916
# git config --global user.email rwnair0916@gmail.com
```

These details you entered are stored under user home directory.

```
# cat .gitconfig
[user]
        name = raghu0916
        email = rwnair0916@gmail.com
```

Create a directory and create some files under it.

```
# mkdir -p test
# touch test/{file1.txt,mydoc.txt}
```

Now you can inititate a git repository using the following command.

```
# git init
Initialized empty Git repository in /root/test/.git/
```

You can see .git directory has been created under this directory which has the

```
# pwd
/root/test
[root@ip-172-31-6-83 test]# ls -ld .git/
drwxr-xr-x 7 root root 111 Sep 29 08:29 .git/
```

After that you need to add the files created to git repository using the following command.

```
# git add file1.txt
# git add mydoc.txt
```

You can check the git status by using following command.

```
# git status
```

You can permanenty add the above directories to the repository using the following command.

```
# git commit -m "Adding new files"
```

Now check the status again.

```
# git status
```

Now add some content to one of the file.

```
# cat file1.txt
Hello
#
```

Check the status again.

```
# git status
```

Commit the changes to staging are using following command.

```
# git commit -am 'Added new line'
```

Now modify the same file again by adding some other content.

```
# cat file1.txt
Hello
Hello World
#
```

You can now try to see the differences made from previous commit to the latest changes you made using the following command.

```
# git diff
diff --git a/file1.txt b/file1.txt
index e965047..727b041 100644
--- a/file1.txt
+++ b/file1.txt
@@ -1 +1,2 @@
 Hello
+Hello World
```

Check the previous commits log by using the following command.

```
# git log
commit c2e68a2848a7baf05fcfc9c360912800637b507e
Author: raghu0916 <rwnair0916@gmail.com>
Date:   Thu Sep 29 08:51:32 2016 -0400

    Added new line

commit 02efb5f6800747a18f831844900dd0ce8ea813dd
Author: raghu0916 <rwnair0916@gmail.com>
Date:   Thu Sep 29 08:48:49 2016 -0400

    Adding new files
```

You can use -p option to the above command to see more description too.

Also run the following command for complete summary of the commit log.

```
# git log --stat --summary
```

Branches in GIT:

List the total branches of the repository using the following command.

```
# git branch
```

Create a branch **'test'** using following command.

```
# git branch test
```

You can check the branches again.

```
# git branch
```

You can switch to branch from **master** using the following link.

```
# git checkout test
```

You can check the current branch which is pointed using following command.

```
# git branch
```

Make some changes in branch and commit them.

```
# cat file1.txt
Hello
Hello World
New line created in branch
```

Commit the changes made in branch.

```
# git commit -a -m 'new line from branch'
```

Now you can switch to **master** branch and then check the content of the file which it will not have the changes made it in branch.

```
# git branch master
# git branch
# cat file1.txt
Hello
Hello World
```

You can merge the changes made in branch to **master** branch using the following command.

```
# git merge test
```

You can delete a branch using the following command.

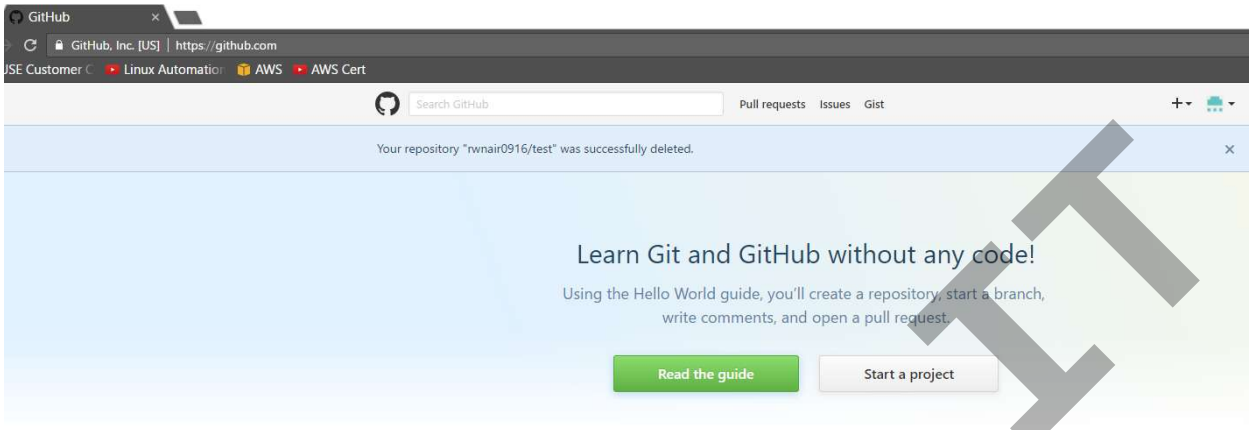
```
# git branch -d test
```

Also if you want to discard the changes of branch and delete it forcefully then you can use the following command.

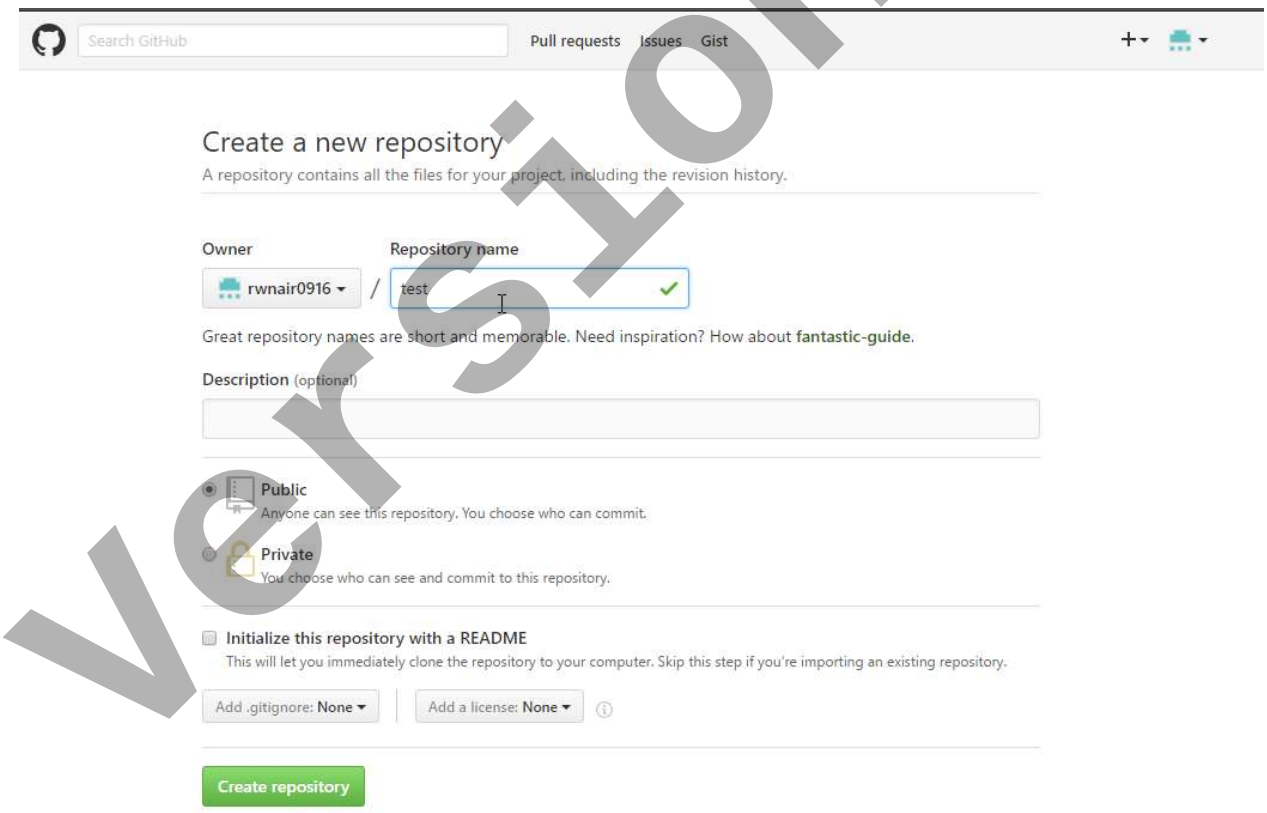
```
# git branch -D test
```

GIT Repositories using GitHub:

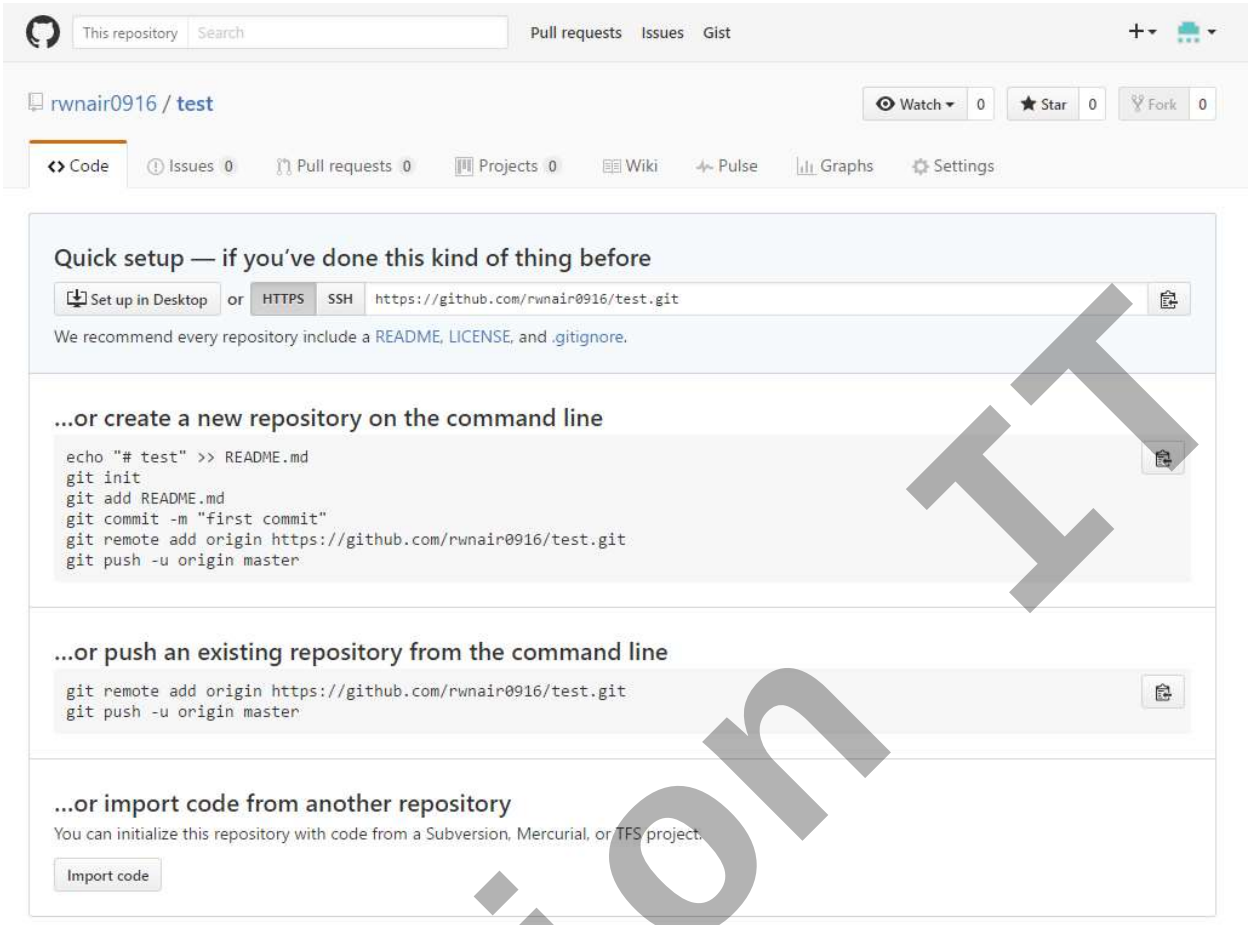
Create an account on GitHub which is free and after registration you can create a project as shown.



Click on “Start a Project”



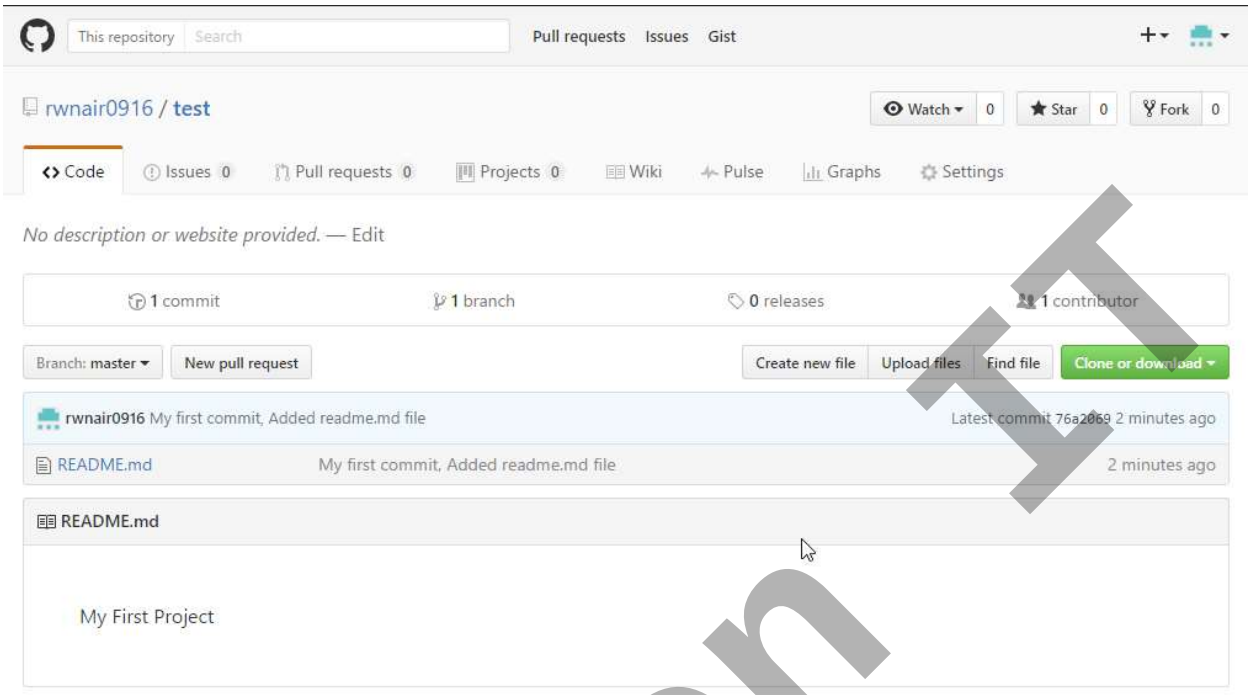
Provide the project name and click on “Create repository”.



Run the above commands from your workstation.

```
[screen 0: root@ip-172-31-6-83:~/test/test]
# mkdir test
# cd test/
# echo "My First Project" >> README.md
# git init
Initialized empty Git repository in /root/test/test/.git/
# git add README.md
# git commit -m 'My first commit, Added readme.md file'
[master (root-commit) 76a2069] My first commit, Added readme.md file
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
# git remote add origin https://github.com/rwnair0916/test.git
You have new mail in /var/spool/mail/root
# git push -u origin master
Username for 'https://github.com': rwnair0916@gmail.com
Password for 'https://rwnair0916@gmail.com@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 246 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/rwnair0916/test.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
#
```

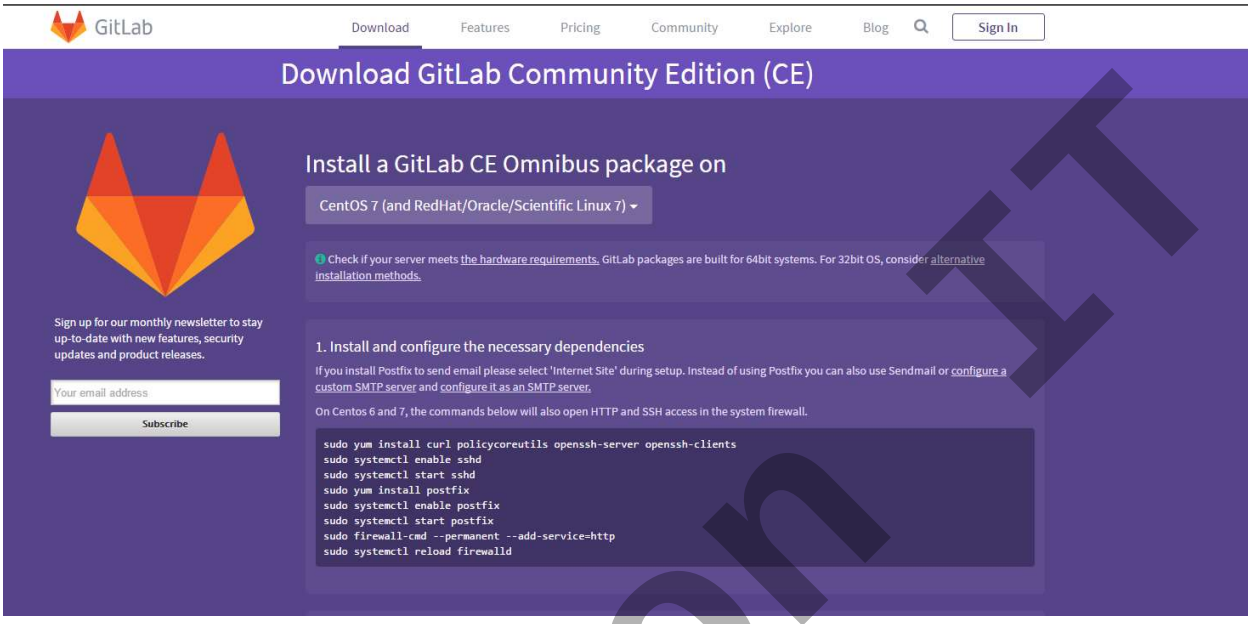
Now you can see the file gets updated in the GitHub repository.



GitLab Installation and Configuration:

You can download the GitLab Community edition from its website directly.

URL : https://about.gitlab.com/downloads/



Run the list of commands given by them in your server.

1. Install and configure the necessary dependencies

```
# yum install curl policycoreutils openssh-server openssh-clients
# systemctl enable sshd
# systemctl start sshd
# yum install postfix
# systemctl enable postfix
# systemctl start postfix
# firewall-cmd --permanent --add-service=http
# systemctl reload firewalld
```

2. Add the GitLab package server and install the package

```
# curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash
# yum install gitlab-ce
```

3. Configure and start GitLab

```
# gitlab-ctl reconfigure
```

4. Browse to the hostname and login

Open the browser and give the IP-Address and then provide root password and create a project and then continue.

ANT

- ANT evolved from the UNIX based build utility called “MAKE”.
- ANT was initially created by James Duncan Davidson as a part of TOMCAT and contributed to the Apache Jakarta Project.
- It later was promoted to an individual project under Apache
- ANT stands for “**Another Neat Tool**”
- Ant is a Java-based build tool.
- Simply put, ANT executes tasks configured in a build XML file.

Advantages of using ANT

- ANT is not OS specific unlike MAKE which is UNIX specific
- ANT executes it’s tasks as JAVA programs which makes it platform independent
- ANT tasks support JAVA project structures which eases it’s use with JAVA/J2EE applications
  - For eg:
    - Manifest in a jar file
    - Web-inf in a war file
- ANT is extensible. ANT tasks can be extended to create customized tasks.
- Using ANT to build an application involves writing a simple configuration XML file. ANT will do the rest.

Advantages of using ANT

- ANT integrates well with editors such as Eclipse, IntelliJ IDEA. You may almost never need to go the command-prompt to deal with ANT.
- ANT can invoke third party plug-in tasks
  - *For eg : PMD which is a tool to review code can be invoked as a third party plug-in in an ANT build file.*
- ANT has support from most application servers
  - *For eg: WebLogic app server provides a "wlserver" Ant task that enables you to start, reboot, shutdown, or connect to a WebLogic Server instance*
- There is NAnt for .NET projects which works pretty much like ANT. If you know one, you know the other.

ANT Basics

- Configurations for a project build are written in XML files in ANT
- To build an application, an ANT user has to write a simple configuration XML file. ANT will do the rest.
- ANT parses the build XML files using it's own JAXP compliant parser
- ANT executes tasks referred in the build XML to build an application
- Tasks are wrapped up by targets.
- The tasks are sequenced by creating dependencies amongst targets. If target B is dependent on target A, target B cannot start until and unless target A is complete.

- There are an enormous number of core and optional tasks built into ANT for a variety of purposes.
- Moreover, third party tools or application servers provide their own ANT tasks as well.
- A custom task can be written and used in a build XML to be invoked by ANT although you may almost never need to write one.

What is XML

- e**X**tensible **M**arkup **L**anguage, is a specification for creating custom markup languages
- W3C Recommendation
- Primary purpose is to help computers to share data
- XML is *meta-language.* This means that you use it for creating languages.
- XML is an extensive concept.

Simple Generic XML Example

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<presentation>
  <slide number="1">
    <name>Introduction to XML</name>
    <contents>XML is ...</contents>
  </slide>
</presentation>
```

Element vs. Tag vs. Attribute

- **Element** consists of *start tag, optional content* and an *end tag:*
  - <name>Introduction to XML</name>
- **Start tag**
  - <name>
- **Content**
  - Introduction to XML
- **End tag**
  - </name>
- **Start tag** may have **attribute**
  - <slide **number="1"**>

Rules about Elements

- **Only one root - element**
- Every element contains starting tag and an ending tag
- Content is optional: **Empty element**
  - <x></x> <!-- same as -->
  - <x/>
- Tag – names are case-sensitive:
  - <X></x> <!-- Error -->
- Elements must be ended with the end tag *in correct order*:
  - problem here <!– Error →

Rules about Attributes

- XML elements can have attributes in the start tag.
- Attributes must be quoted:
  - <person sex="female">
  - <person sex='female'>
  - <gangster name='George "Shotgun" Ziegler'>
  - <gangster name="George "Shotgun" Ziegler">

Naming Tags

- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation character
- Names must not start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces

Build XML - Project element

- A build XML file starts with the “Project” root element.
  - Sample:

    <project name="test" default="compile" basedir=".">
  - “name” is an optional attribute used to specify the name of the project for which the build file is being written.
  - “default” is an optional attribute used to define a target that will be executed by default. If this attribute is not defined, ANT will use an implicit target to execute all top level tasks.
  - “basedir” attribute is used to define a base directory which will be a start point for all paths in the XML. In case this attribute is not defined, the parent directory of the build XML will be taken as the “basedir”.
- A “project” has one or more “target” elements within it.

Build XML – Target element

- A "target" is a bunch of "tasks" that need to be executed.
  - Sample:

    <target name="A"/>
    - "name" attribute is mandatory and indicates the name of the target

- This target has tasks to compile web components.
  - javac
  - copy

```
<target name="compile-web">
   <javac srcdir="${src}"
destdir="${build}">
       <include name="**/web/**"/>
   </javac>
   <copy todir="${build}">
       <fileset dir="web">
       <include name="**/*.properties" />
       </fileset>
   </copy>
</target>
```

Build XML– Target element

- A " target" can depend on one or more "targets".
- The dependencies on other target(s) for a target is defined using the "depends" attribute.
- ANT will decide the order of processing of targets in a build XML using these dependencies between targets.
- In this example,
  - "compile" depends on "init"
  - "package" depends on "compile" and "copyFiles"

```
<target name="init">
   …
</target>
<target name="copyFiles">
    …
</target>
<target name="compile" depends="init">
        …
</target>
<target name="package" depends="compile,
copyFiles">
        …
</target>
```

Build XML– Target element

- Some rules to follow while defining targets
  - ANT will execute the targets defined in the “depends” attribute from left to right
    - Sample

```
<target name=“package” depends=“compile, copyFiles”>
        <jar> …</jar>
</target>
```

    - Here “compile” will get executed first and then “copyFiles”

Build XML– Target element

- “if” and “unless” rules on a target
  - The execution of a target can be forced to depend on a property using the “if” attribute “ or the “unless” attribute
  - A target will execute if the property in it’s “if” attribute is set.
    - Sample:

```
<target name=“compileEJB” if=“EJBModule”/>
```

      - The target “compileEJB” will be executed only if the property “EJBModule” has some value in the build XML. The value could be even an empty string.

Build XML– Target element

- A target will not execute if the property in it's "unless" attribute is set.
  - Sample:

    <target name="compileAllModules" unless="ModuleList"/>
    - The target "compileAllModules" will be executed only if the property "ModuleList" is not set.
- If "depends" and "if/unless" are set on a target, the "depends" will executed first

Build XML– Task element

- A task is a program that can be executed.
- A task will look like this :

  <taskname attribute1="value1" attribute2="value2".....attributeN="valueN"/>
- Sample

```
<javac srcdir="${src}" destdir="${build}" source="${source.version}" target="${target.version}">
  <include name="**/appclient/**"/>
   <exclude name="**/*.properties" />
  <classpath refid="classpath" />
</javac>
```

  - Javac is a task with attributes such as "srcdir".
  - The task supports elements within it such as "include" and "exclude".
  - This task is a JAVA class in ant.jar under "lib" directory of ANT under the package "org.apache.tools.ant.taskdefs".

Build XML– Properties

- There is a need to refer to the code directory structure in the build XML
- Environment specific data such as JDK_HOME, application server installation location, database URL, server IP, port may be referenced in the build tasks.
- External property files may need to be referenced in the build XML.
- System properties such as java.home, os.name may be required in the build XML.
- ANT properties cater to all these needs and much more.

Build XML– Properties

- Properties are name-value pairs.
- Properties can be set in an ANT build file by
  - using a task called “property” under the “project” element
    - Sample:

```
<property name="libDirectory" value=“./lib"/>
```

  - using a separate properties file with name value pairs and referencing the file in the build file.

```
<property file="../common/build.properties"/>
```

APACHE ANT

Installing ANT using YUM:

Install the package using yum command.

```
# yum install ant -y
```

Check the installed ant version.

```
# ant –version
```

Installing ANT using SOURCE:

Download ANT from ant.apache.org URL.

```
# wget http://mirror.fibergrid.in/apache//ant/binaries/apache-ant-1.9.7-bin.tar.gz
```

```
# tar xzf apache-ant-1.9.7-bin.tar.gz
```

Check the version by running the binary came in package.

```
# ./apache-ant-1.9.7/bin/ant -version
```

Download the sample code and it already has **build.xml** included.

```
# git clone https://github.com/versionit/antproj.git
```

Run **ant** command to start building, It automatically picks **build.xml** and proceed with building.

```
[root@ip-172-31-54-155 antproj]# /opt/apache-ant-1.9.7/bin/ant
Buildfile: /root/antproj/build.xml

init:
    [mkdir] Created dir: /root/antproj/build/classes
    [mkdir] Created dir: /root/antproj/dist

compile:
    [javac] /root/antproj/build.xml:16: warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=las
t; set to false for repeatable builds
    [javac] Compiling 4 source files to /root/antproj/build/classes
    [javac] Note: /root/antproj/src/com/vaannila/web/UserController.java uses or overrides a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.

war:
      [war] Building war: /root/antproj/dist/AntExample.war

BUILD SUCCESSFUL
Total time: 1 second
[root@ip-172-31-54-155 antproj]#
```

Now check the new build file.

Deploy the file in apache tomcat and test it.

What is Maven

- **Maven** is a **build automation tool** used primarily for Java projects.
- *Maven addresses two aspects of building software:*

1. *it describes how software is built*
2. *it describes its dependencies*

Maven Project Object Model (POM)

- **Describes a project**
  - Name and Version
  - Artifact Type
  - Source Code Locations
  - Dependencies
  - Plugins
  - Profiles (Alternate build config.)
- **Uses XML by default**

Project Name (GAV)

- Maven uniquely identifies a project using:
  - **g**roupID: project grouping identifier (no spaces or colons)
    - Usually loosely based on Java package
  - **a**rtfiactId: name of project (no spaces or colons)
  - **v**ersion: Version of project
    - Format {Major}.{Minor}.{Maintanence}
    - Add '-SNAPSHOT ' to identify in development

Project Name (GAV)

groupId	This is an Id of project's group. This is generally unique amongst an organization or a project. For example, a banking group com.company.bank has all bank related projects.
artifactId	This is an Id of the project.This is generally name of the project. For example, consumer-banking. Along with the groupId, the artifactId defines the artifact's location within the repository.
version	This is the version of the project.Along with the groupId, It is used within an artifact's repository to separate versions from each other. For example: *com.company.bank:consumer-banking:1.0* *com.company.bank:consumer-banking:1.1.*

Project Name (GAV)

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
      <modelVersion>4.0.0</modelVersion>
      <groupId>org.lds.training</groupId>
      <artifactId>maven-training</artifactId>
      <version>1.0</version>
      <name>Windows 8</name>
      <description>The best OS ever!</description>
      <packaging>jar</packaging>
            <properties>
                  <java.version>1.6</java.version>
            <properties>
</project>
```

Maven LifeCycle

Phase	Handles	Description
prepare-resources	resource copying	Resource copying can be customized in this phase.
compile	compilation	Source code compilation is done in this phase.
package	packaging	This phase creates the JAR / WAR package as mentioned in packaging in POM.xml.
Install	installation	This phase installs the package in local maven repository.
deploy	Deploy	This phase deploys the package in remote maven repository.

APACHE MAVEN

Installing MAVEN from Source:

Download the Maven from Apache website and use it.

```
# cd /opt
# wget http://mirror.fibergrid.in/apache/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz
# tar -xzf apache-maven-3.3.9-bin.tar.gz
# mv apache-maven-3.3.9 maven
```

Check the installed maven version.

```
# cd /opt/maven/bin
# ./mvn –version
```

Add installed maven to PATH so that it can be accessed as a command.

```
# echo “export PATH=$PATH:/opt/maven/bin” >>~/.bashrc
```

Logout and Login again and then you can use maven commands now.

```
# mvn -version
```

Create a Maven Project:

You can create a project directory structure using maven. There are lot types of archetypes and you have to choose one of them.

```
# mvn archetype:generate
```

For these you have to provide following values to create a project.

```
'groupId': : com.mycompany
'artifactId': : sample
'version':  1.0-SNAPSHOT: :
'package':  com.mycompany: :
```

Instead you can provide these values automatically while generating the project directory structure.

```
# mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Check the directory and it has **pom.xml** created automatically.

Build the project using

```
# mvn package
```

You may test the newly compiled and packaged JAR with the following command:

```
# java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App
```

Download the sample project and build the project.

```
# git clone --depth=1 https://github.com/versionit/mvnrepo.git
# cd mvnrepo
# mvn package
```

```
[INFO] Packaging webapp
[INFO] Assembling webapp [javaee7-simple-sample] in [/root/mvnrepo/target/javaee7-simple-sample]
[INFO] Processing war project
[INFO] Copying webapp resources [/root/mvnrepo/src/main/webapp]
[INFO] Webapp assembled in [43 msecs]
[INFO] Building war: /root/mvnrepo/target/javaee7-simple-sample.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.035 s
[INFO] Finished at: 2016-12-07T18:11:15-05:00
[INFO] Final Memory: 15M/37M
[INFO] ------------------------------------------------------------------------
[root@ip-172-31-54-155 mvnrepo]#
```

Deploy the war file in tomcat instance and test it.

You can install into home directory repository using the following task in maven.

```
# mvn install
```

```
[INFO] Building war: /root/mvnrepo/target/javaee7-simple-sample.war
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ javaee7-simple-sample ---
[INFO] Installing /root/mvnrepo/target/javaee7-simple-sample.war to /root/.m2/repository/org/javaee7/sample/javaee7-si
mple-sample/3.13-SNAPSHOT/javaee7-simple-sample-3.13-SNAPSHOT.war
[INFO] Installing /root/mvnrepo/pom.xml to /root/.m2/repository/org/javaee7/sample/javaee7-simple-sample/3.13-SNAPSHOT
/javaee7-simple-sample-3.13-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.320 s
[INFO] Finished at: 2016-12-07T18:11:42-05:00
[INFO] Final Memory: 9M/22M
[INFO] ------------------------------------------------------------------------
[root@ip-172-31-54-155 mvnrepo]#
```

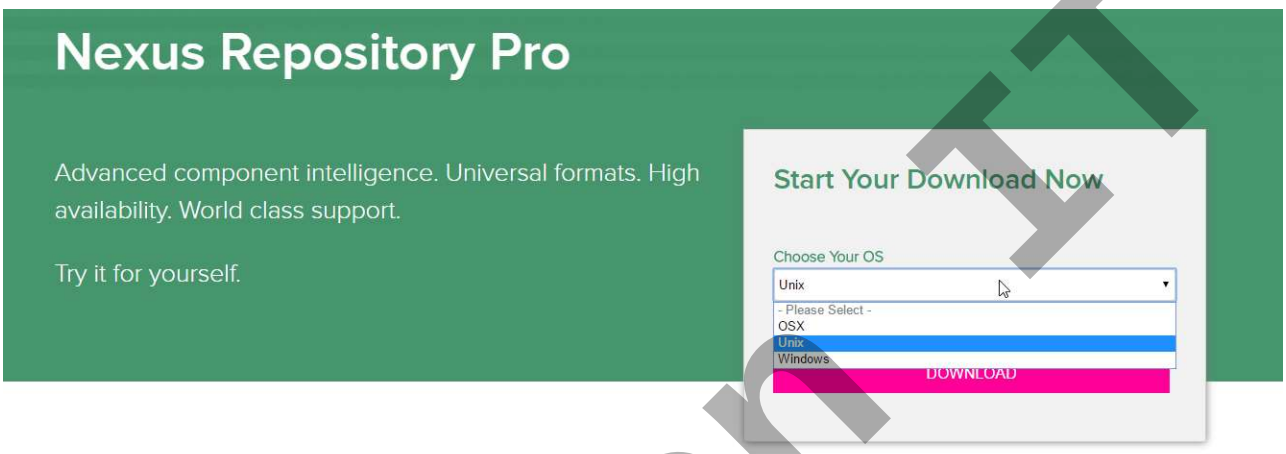
SONATYPE NEXUS

Installing NEXUS on RedHat Linux:

Go to the following URL and Click on "**FREE TRAIL**".

https://www.sonatype.com/nexus-repository-sonatype

Then select on "UNIX" as the required version, Then click on "Download"



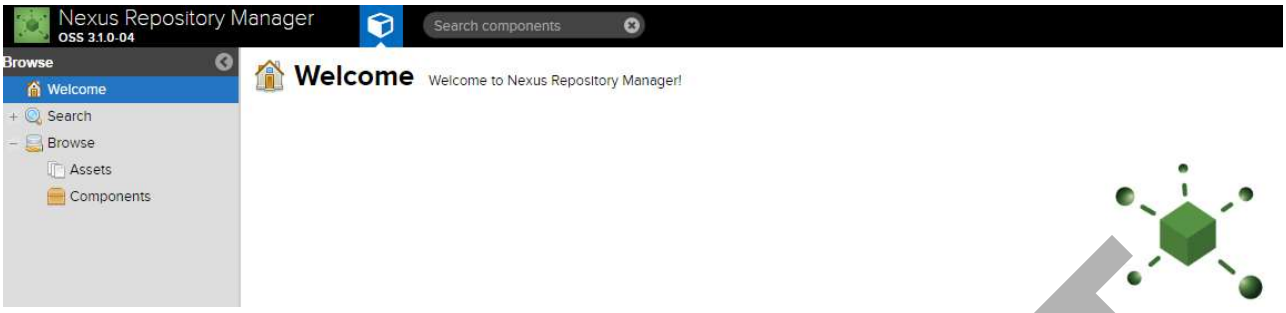
After that you get a download link and use that download link and download using wget command in Linux.

```
[root@ip-172-31-51-136 ~]# wget https://sonatype-download.global.ssl.fastly.net/nexus/3/nexus-3.1.0-04-unix.tar.gz
--2016-12-06 21:42:30--  https://sonatype-download.global.ssl.fastly.net/nexus/3/nexus-3.1.0-04-unix.tar.gz
Resolving sonatype-download.global.ssl.fastly.net (sonatype-download.global.ssl.fastly.net)... 151.101.32.249
Connecting to sonatype-download.global.ssl.fastly.net (sonatype-download.global.ssl.fastly.net)|151.101.32.249|:443...
 connected.
HTTP request sent, awaiting response... 200 OK
Length: 98475782 (94M) [application/octet-stream]
Saving to: 'nexus-3.1.0-04-unix.tar.gz'

73% [=================================================>                ] 72,012,589  42.8MB/s
```

Extract the tar file and run the following command.

```
[root@ip-172-31-51-136 ~]# tar xzf nexus-3.1.0-04-unix.tar.gz
[root@ip-172-31-51-136 ~]# cd nexus-3.1.0-04/bin/
[root@ip-172-31-51-136 bin]# ./nexus start
WARNING: ************************************************************
WARNING: Detected execution as "root" user.  This is NOT recommended!
WARNING: ************************************************************
Starting nexus
[root@ip-172-31-51-136 bin]#
```

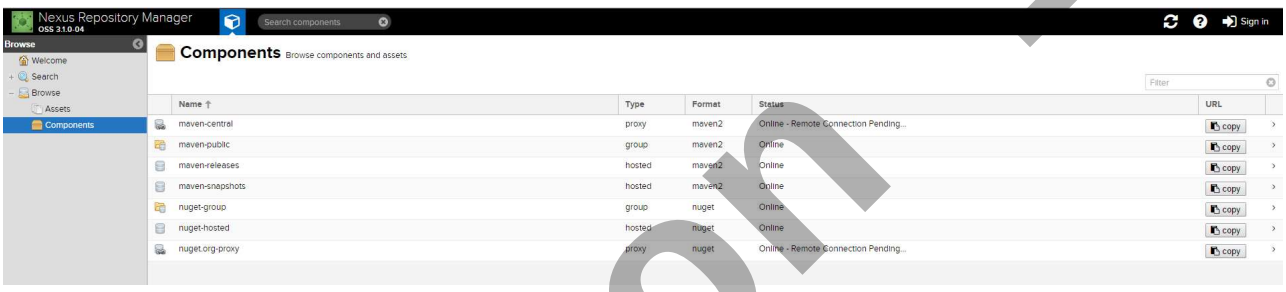
Installation is now completed. Nexus listen on 8081 port by default. Now open the Nexus Manager from browser.



Our repository is ready to keep the files.

You can use user "**admin**" and password "**admin123**" to login and check the repositories.

On left pane you can find "Components and if you click on that you can find all the repositories.



Now are ready with Nexus Repository. All we need to do is configure maven to put the builds in repository.

Configure Maven with Nexus:

Assume you have the code on server and proper pom.xml file. Then you can make a package using maven commands.

```
[root@maven ~]# ls
anaconda-ks.cfg  javaee7-simple-sample-master  master.zip
[root@maven ~]# cd javaee7-simple-sample-master/
[root@maven javaee7-simple-sample-master]# /opt/maven/bin/mvn package

[INFO] Building war: /root/javaee7-simple-sample-master/target/javaee7-simple-sample.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.111 s
[INFO] Finished at: 2016-10-12T12:10:34-04:00
[INFO] Final Memory: 15M/36M
[INFO] ------------------------------------------------------------------------
[root@maven javaee7-simple-sample-master]#
```

Now open the pom.xml and you can see some repository entries pointing to localhost, In case if your Maven and Nexus is on same server then you can give localhost in URL. If not change that to proper URL.

```
[root@ip-172-31-54-155 maven]# grep url pom.xml
        <url>https://github.com/javaee-samples/javaee7-simple-sample.git</url>
            <url>http://172.31.51.136:8081/repository/maven-releases/</url>
            <url>http://172.31.51.136:8081/repository/maven-releases//</url>
[root@ip-172-31-54-155 maven]#
```

After changing to proper URL's then execute the following command.

```
# mvn deploy
```

You can see that the deployment has failed.

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy (default-dep
loy) on project javaee7-simple-sample: Failed to deploy artifacts: Could not transfer artifact org.
javaee7.sample:javaee7-simple-sample:war:1.11-20161012.161523-1 from/to deployment (http://52.66.27
.221:8081/nexus/content/repositories/snapshots/): Failed to transfer file: http://52.66.27.221:8081
/nexus/content/repositories/snapshots/org/javaee7/sample/javaee7-simple-sample/1.11-SNAPSHOT/javaee
7-simple-sample-1.11-20161012.161523-1.war. Return code is: 401, ReasonPhrase: Unauthorized. -> [He
lp 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following art
icles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
[root@maven javaee7-simple-sample-master]#
```

It is due to the nexus credentials are missing.

You can provide the credentials either globally which is going to be same for all the users using this maven or you can go with specific user by providing the credentials in his local directory.

Let us use those credentials by local user, for that you need to create a file in your home directory maven configuration.

File Name : /root/.m2/settings.xml (Assuming root as the user)

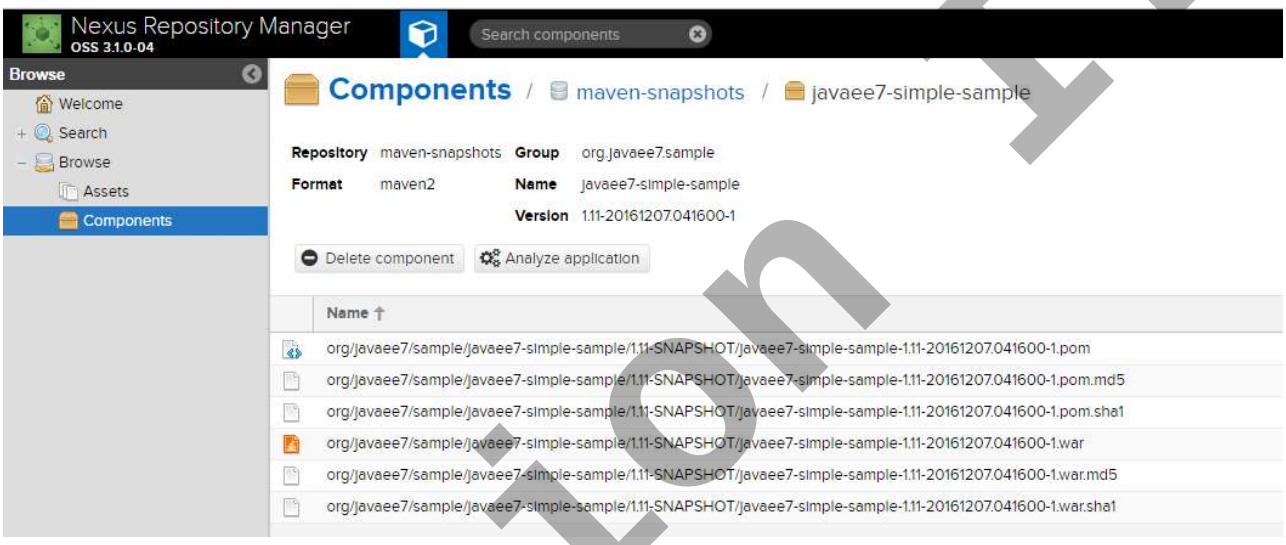
```
[root@maven javaee7-simple-sample-master]# cat /root/.m2/settings.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0
http://maven.apache.org/xsd/settings-1.1.0.xsd">
   <servers>
     <server>
       <id>deployment</id>
       <username>admin</username>
       <password>admin123</password>
     </server>
  </servers>
</settings>
[root@maven javaee7-simple-sample-master]#
```

You can get this template file from MAVEN_HOME/conf directory.

Now try to deploy and it will be successful.

```
Uploading: http://52.66.27.221:8081/nexus/content/repositories/snapshots/org/javaee7/sample/javaee7
-simple-sample/maven-metadata.xml
Uploaded: http://52.66.27.221:8081/nexus/content/repositories/snapshots/org/javaee7/sample/javaee7-
simple-sample/maven-metadata.xml (298 B at 15.3 KB/sec)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.214 s
[INFO] Finished at: 2016-10-12T12:21:03-04:00
[INFO] Final Memory: 11M/28M
[INFO] ------------------------------------------------------------------------
[root@maven javaee7-simple-sample-master]#
```

You can verify the same from Nexus Repository Manager.



That is all ☺ …

You can download the maven sample project from the following URL.

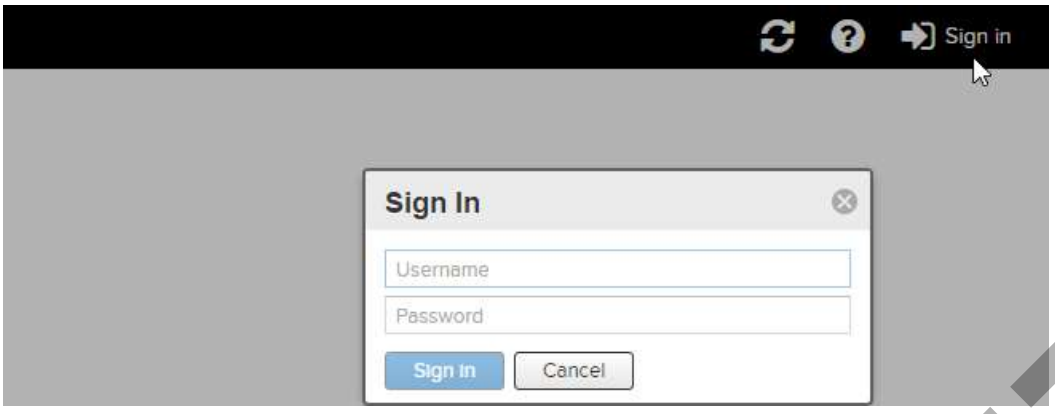
```
# git clone https://github.com/versionit/mvnrepo.git
```

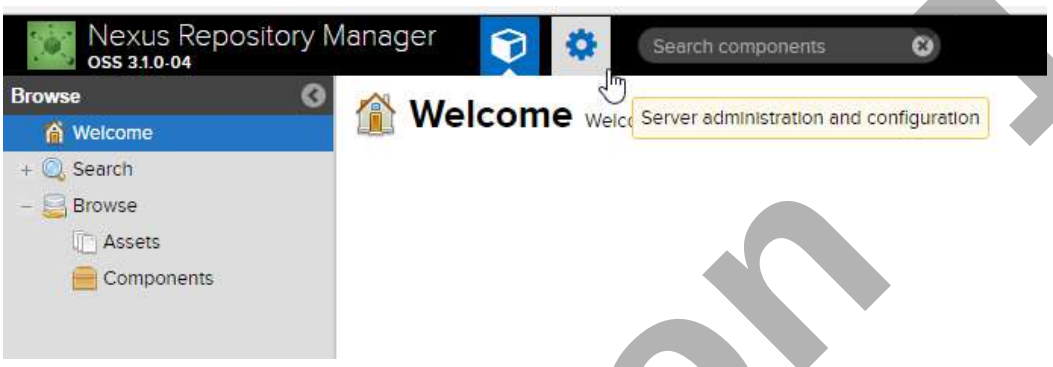
You can create a project and associated user for that project in Nexus and use that username and password in maven integration.

Firstly, let us create project. Sign In to Nexus URL with default username and password.

Username: **admin**

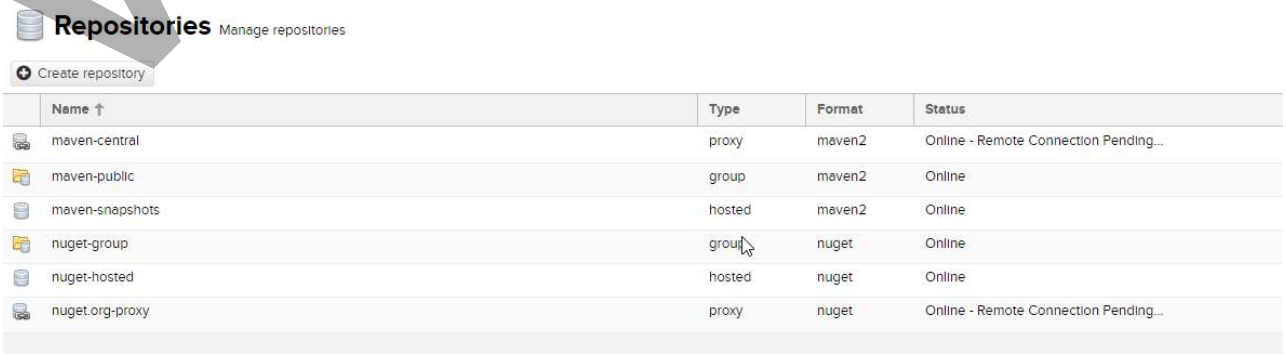
Password: **admin123**



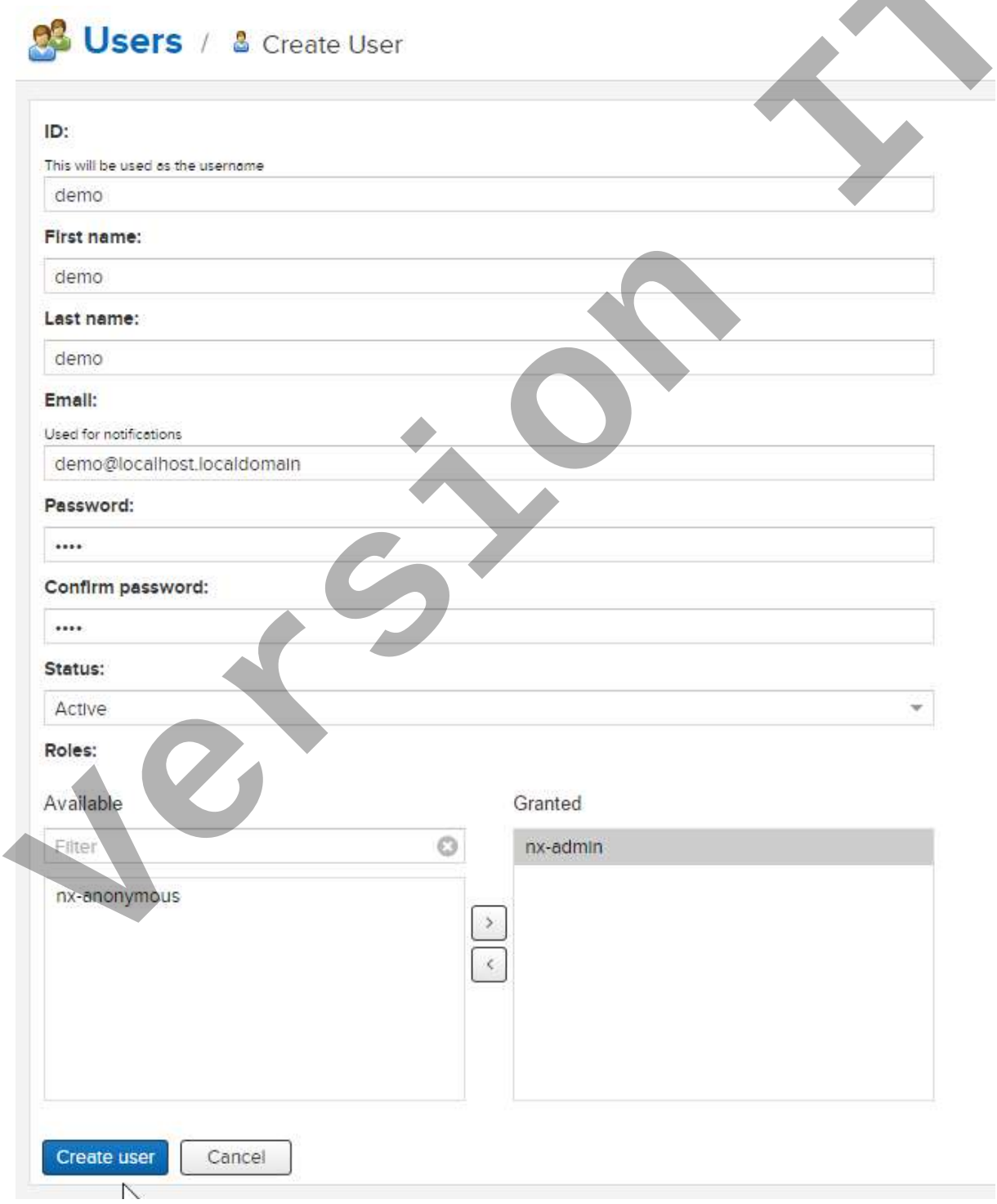


Goto “Server administration and configuration”. Then click on repositories.

Then click on “Create repository” and then “Maven2 (hosted)”



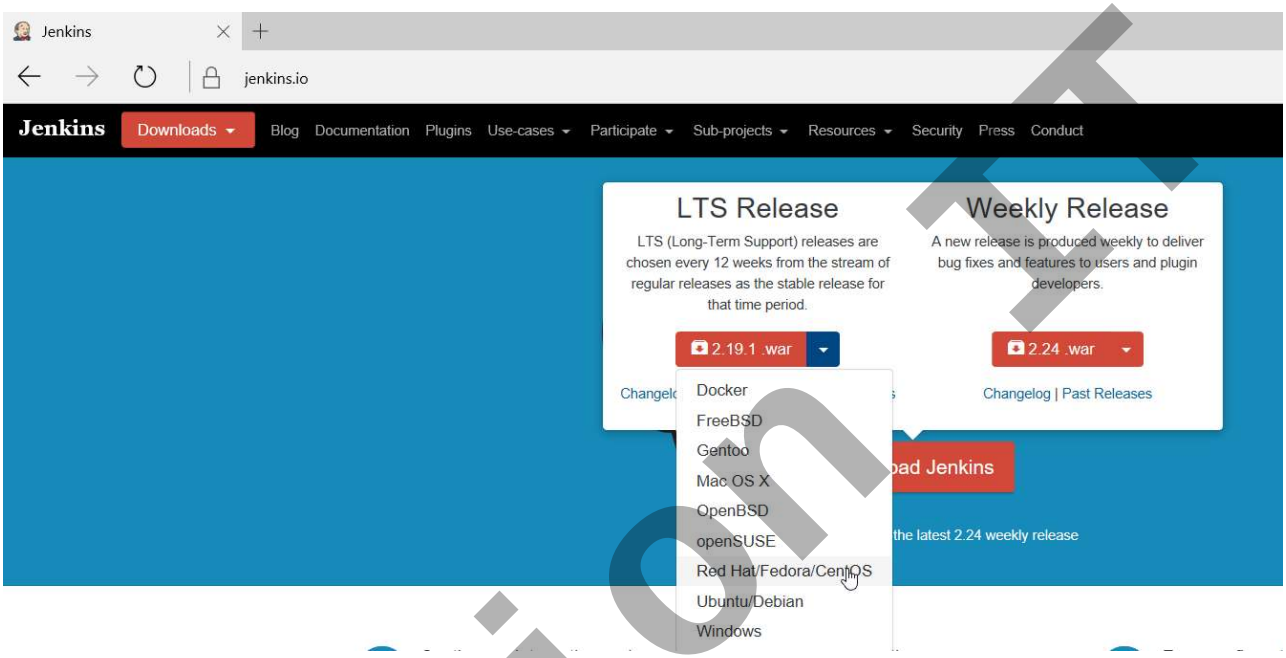
Then create the user.



JENKINS

Install Jenkins Master as a Service:

Download the Jenkins from the Jenkins Website, Select **Redhat** while downloading instead of war file.



Then follow the given steps.

Before starting the Jenkins service, you need to ensure that java has been installed, In case if it is not installed then install java package prior to start Jenkins service.

```
# yum install java -y
```

Then start the Jenkins as a service.

```
# systemctl start Jenkins
```

You can check whether the service is running or not using following commands.

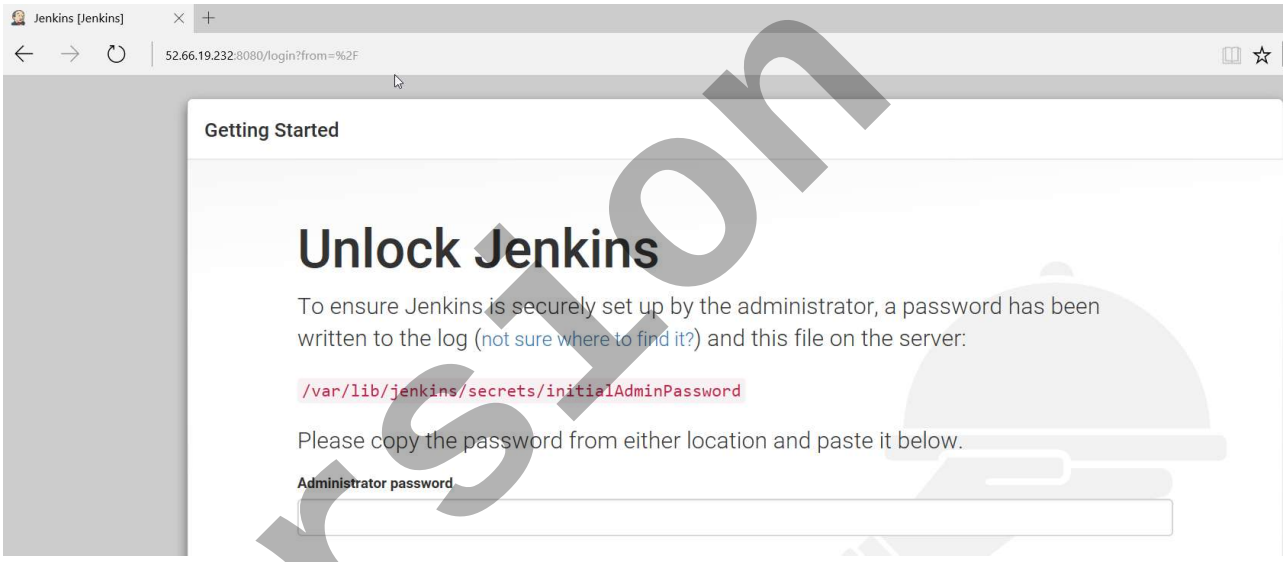
 # ps -ef |grep Jenkins

```
[root@jenkins-master ~]# ps -ef |grep jenkins
jenkins   2216     1  8 01:17 ?        00:00:23 /etc/alternatives/java -Dcom.sun.akuma.Daemon=daemo
nized -Djava.awt.headless=true -DJENKINS_HOME=/var/lib/jenkins -jar /usr/lib/jenkins/jenkins.war --
logfile=/var/log/jenkins/jenkins.log --webroot=/var/cache/jenkins/war --daemon --httpPort=8080 --de
bug=5 --handlerCountMax=100 --handlerCountMaxIdle=20
root      2302  2284  0 01:21 pts/0    00:00:00 grep --color=auto jenkins
[root@jenkins-master ~]#
```

Now you can access Jenkins webpage using the IP-Address of the server. Port number of Jenkins by default is 8080.

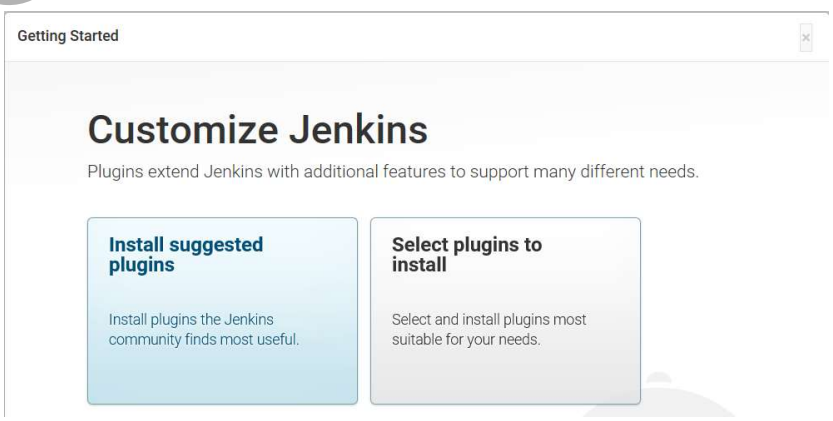
```
[root@jenkins-master ~]# netstat -antp |grep java
tcp6       0      0 :::8080                 :::*                    LISTEN      2216/java

tcp6       0      0 :::43316                :::*                    LISTEN      2216/java

[root@jenkins-master ~]#
```

Now open the browser and typein the URL with port 8080.

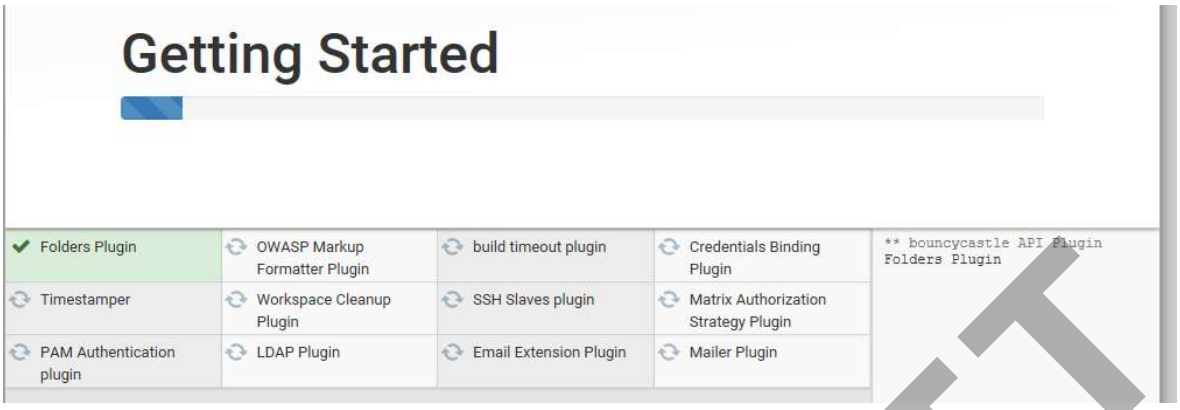


You would be landed in above webpage, you need to provide the default password came with installation mentioned in file in webpage and click on Continue.

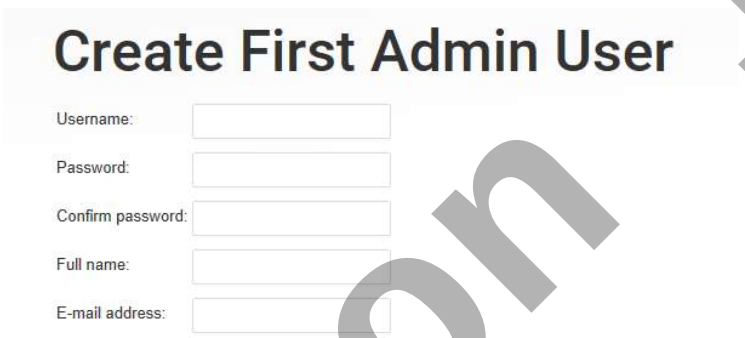
Select on “Install Suggested plugins” to install basic plugins. Else click on “Select plugins to install” to customize.

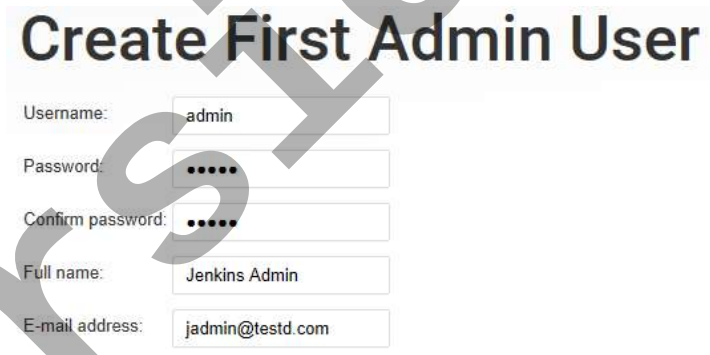


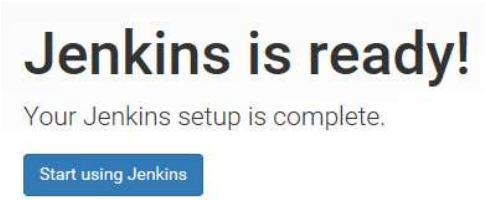
Then it starts downloading plugins.



Then create an administrator account with password and click on Save and Finish







Installation of Jenkins Slave:

After login to Jenkins click on "**Manage Jenkins**"

You can click on "**Manage Nodes**"

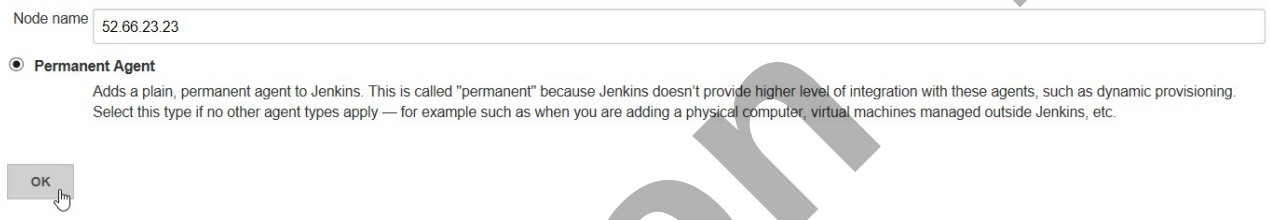
Now you can see already one Master node.

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	8.20 GB	0 B	8.20 GB	0ms
	Data obtained	43 min	43 min	43 min	43 min	43 min	43 min

Refresh status

Now click on "**New Node**"

Then provide the IP-Address (Node name) and click on "OK"



Provide the complete details and then click on "**Save**"

After some time, you can find the host comes online.

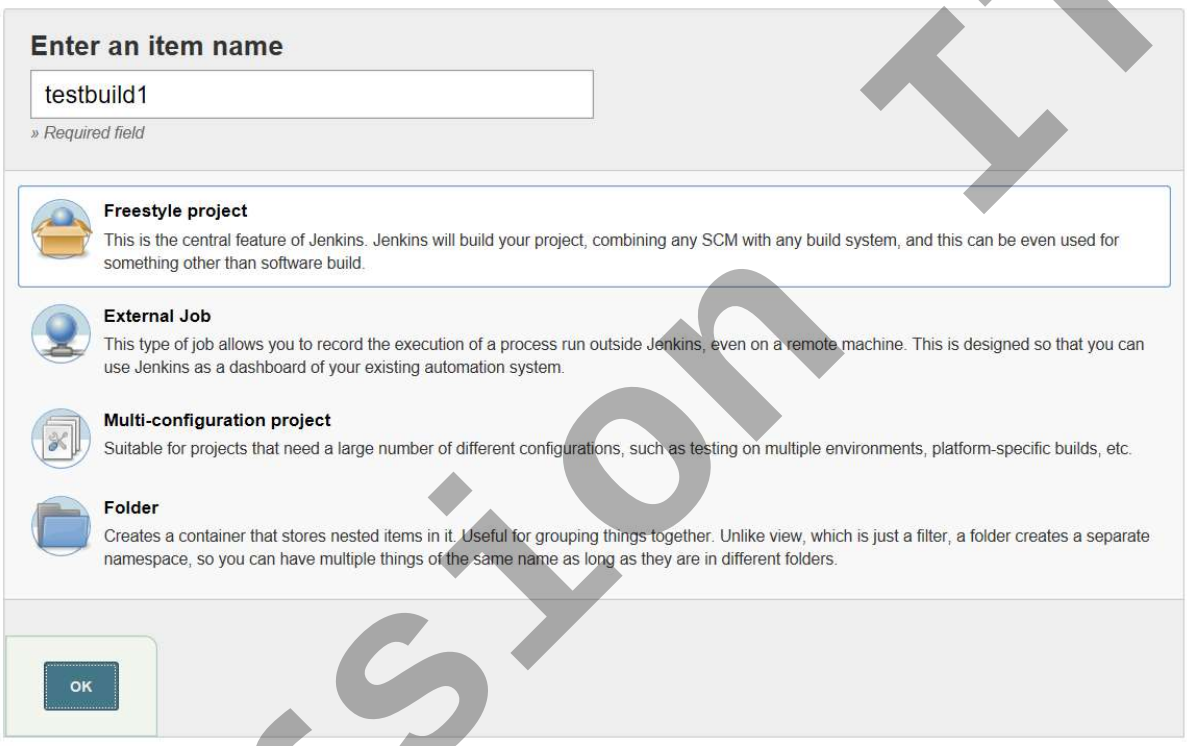
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	8.06 GB	0 B	8.06 GB	0ms
	Slave1	Linux (amd64)	In sync	8.23 GB	0 B	8.23 GB	4007ms
	Data obtained	20 sec	20 sec	20 sec	20 sec	20 sec	20 sec

Refresh status

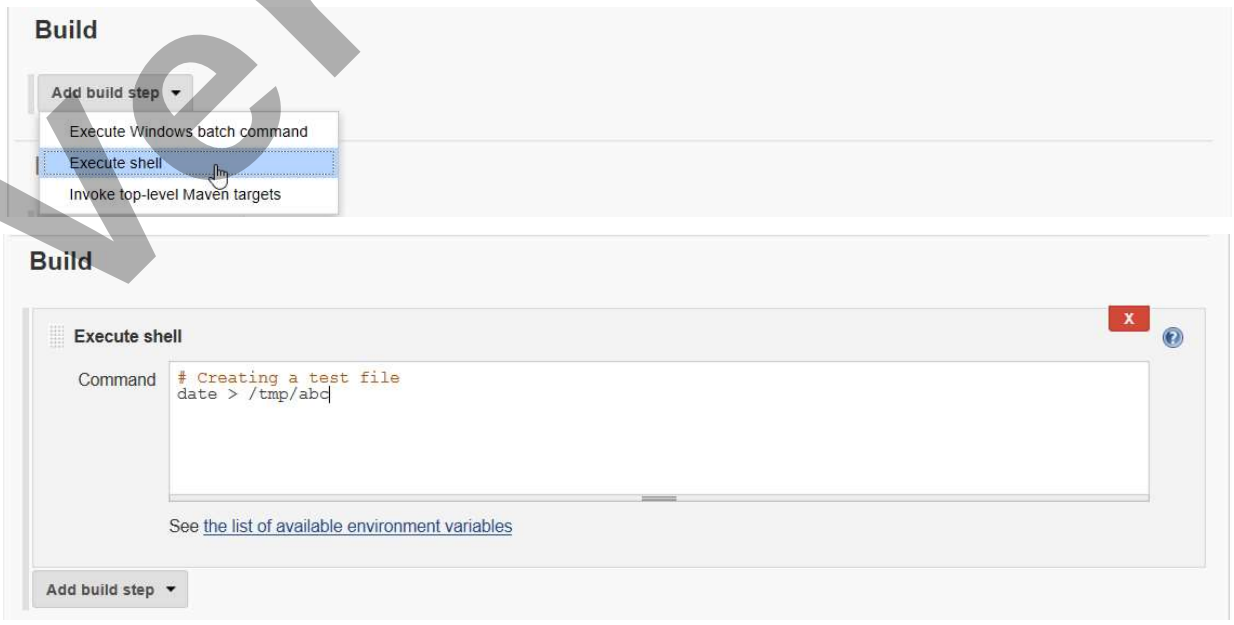
Create a Jenkins Job:

After login to Jenkins click on “New Item”.

Then provide the “Item Name” and click on “Freestyle Project” and click “OK”.



Then provide the required information.

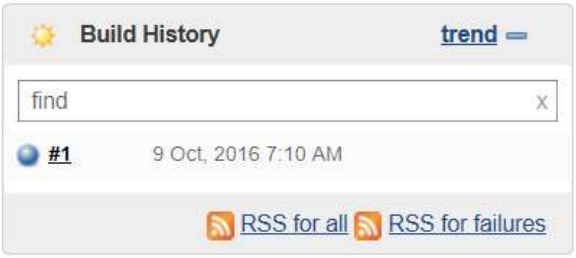


Then click on "**Save**".

You can manually run the job using 

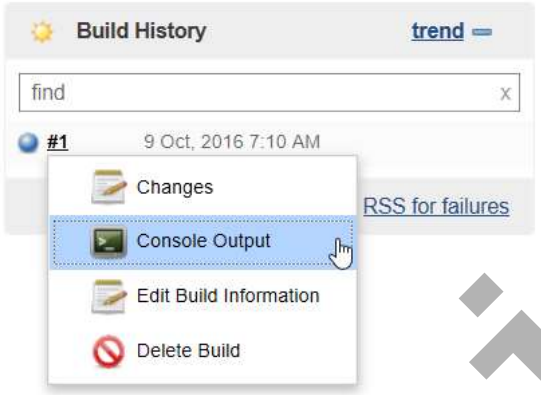
Option.

You can see the build status at below.



Blue colour denotes build successful, Red denotes build failure.

You can click on "**Console Output**" for clear information



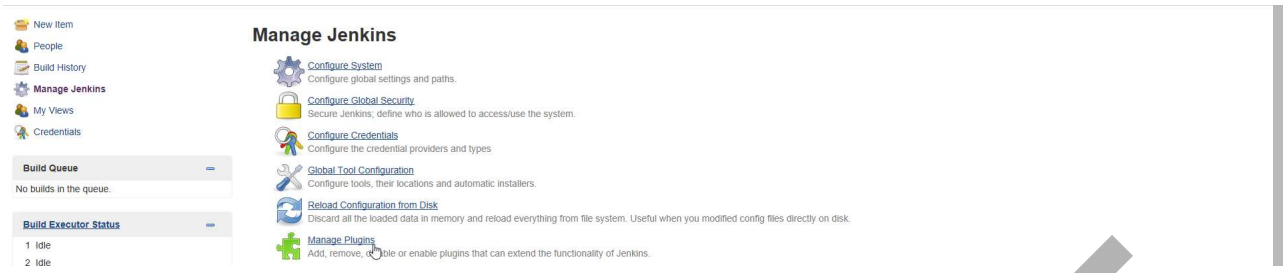
You can verify on the server that if it has created a file or not.

```
[ec2-user@jenkins-slave ~]$ cat /tmp/abc
Sun Oct  9 07:10:56 EDT 2016
[ec2-user@jenkins-slave ~]$
```

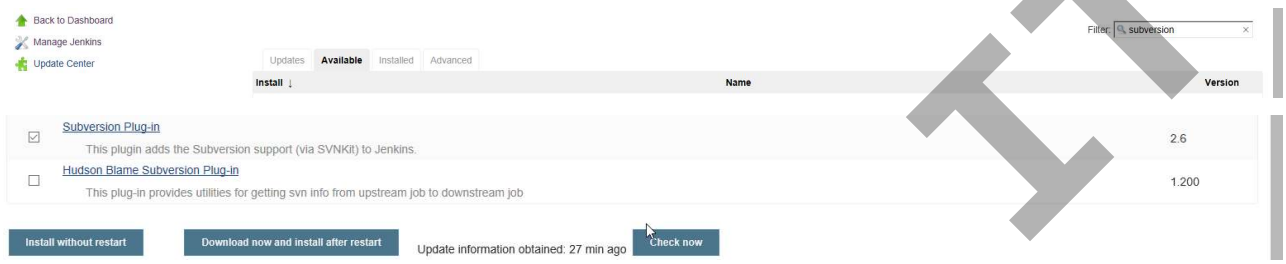
Integrate Jenkins with Subversion:

After login to Jenkins click on "**Manage Jenkins**"

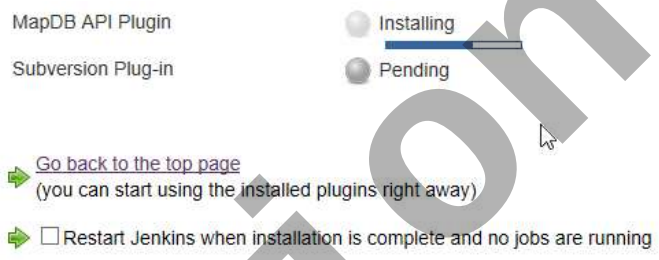
Then click on "**Manage Plugins**"



Go to "**Available**" and Search for "**Subversion**"



Click on required plugins and "**Install without restart**".



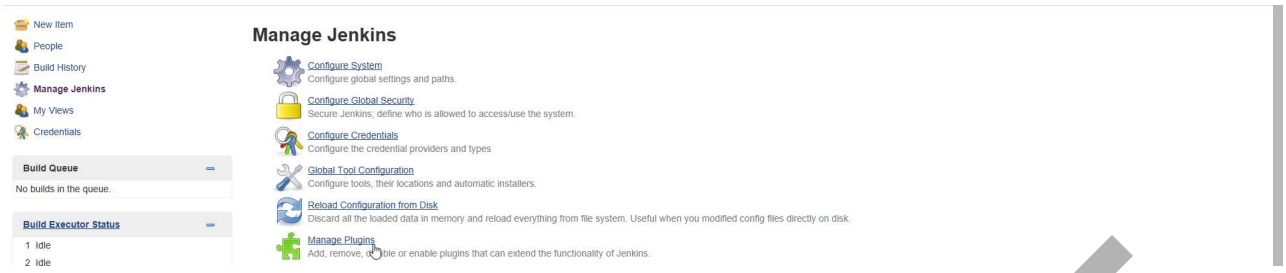
After completing click on "**Restart Jenkins**"

You can see now the subversion support for Jenkins.

Integrate Jenkins with GIT:

After login to Jenkins click on "**Manage Jenkins**"

Then click on "**Manage Plugins**"

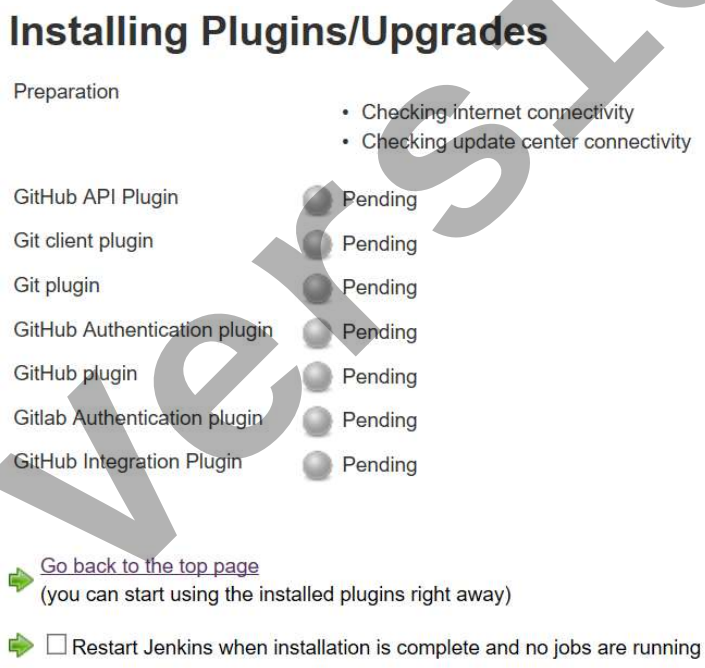


Go to "**Available**" and Search for "**git**" and select required plugins.

Click on 

.

Then it starts installing.

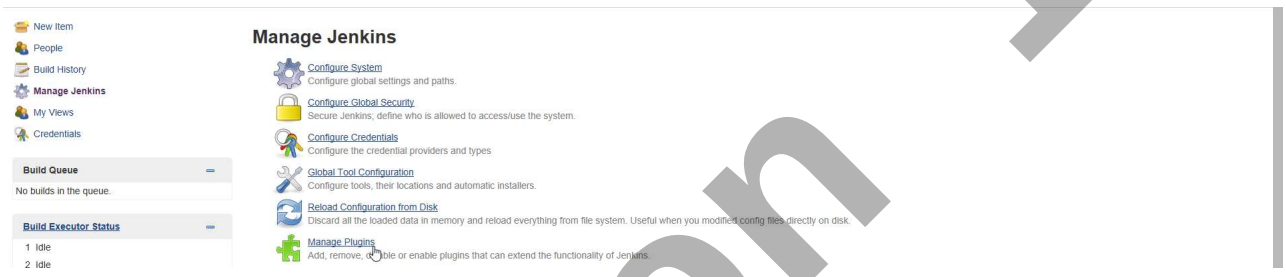


Now you can see Jenkins has GitHub.

Integrate Jenkins with ANT:

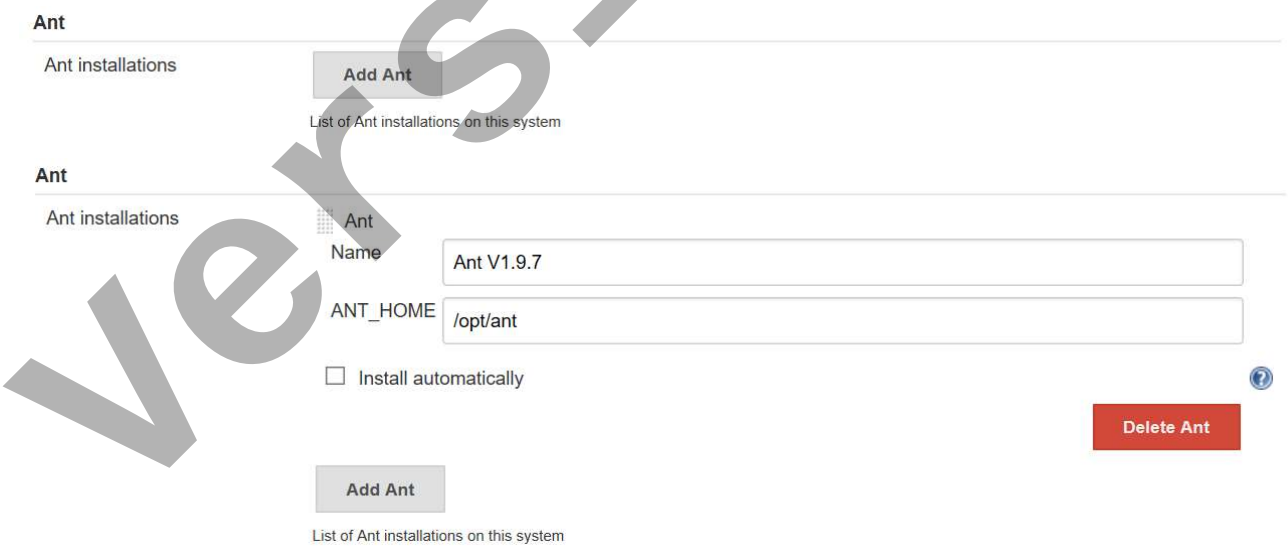
After login to Jenkins click on "**Manage Jenkins**"

Then click on "**Manage Plugins**"



Go to "**Available**" and Search for "**ant**" and select required plugins.

Now click on "**Manage Jenkins**" and click on "**Global Tool Configuration**".



Then click on "**Save**"

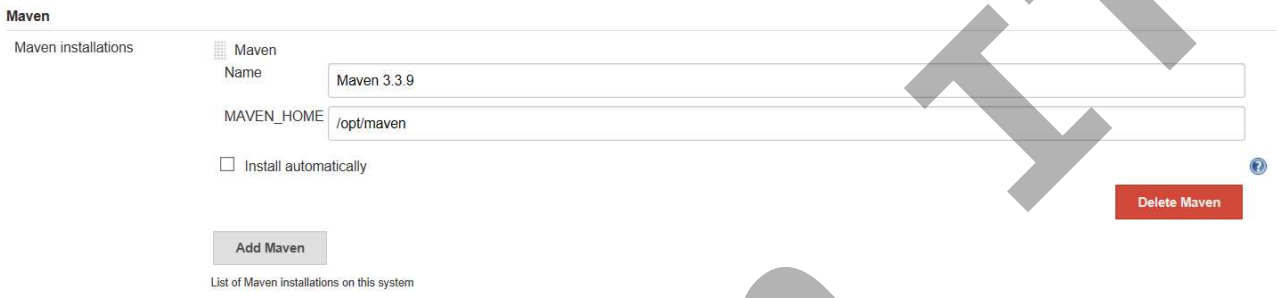
Now Jenkins is ready for Ant configuration.

Integrate Jenkins with MAVEN:

This version of Jenkins comes with MAVEN integrated by default. All you need to install Maven and Configure Maven.

To configure Maven, Click on "**Manage Jenkins**" and then "**Global Tool Configuration**".

Click on "**Add Maven**" and specify the location of Maven installed on server.



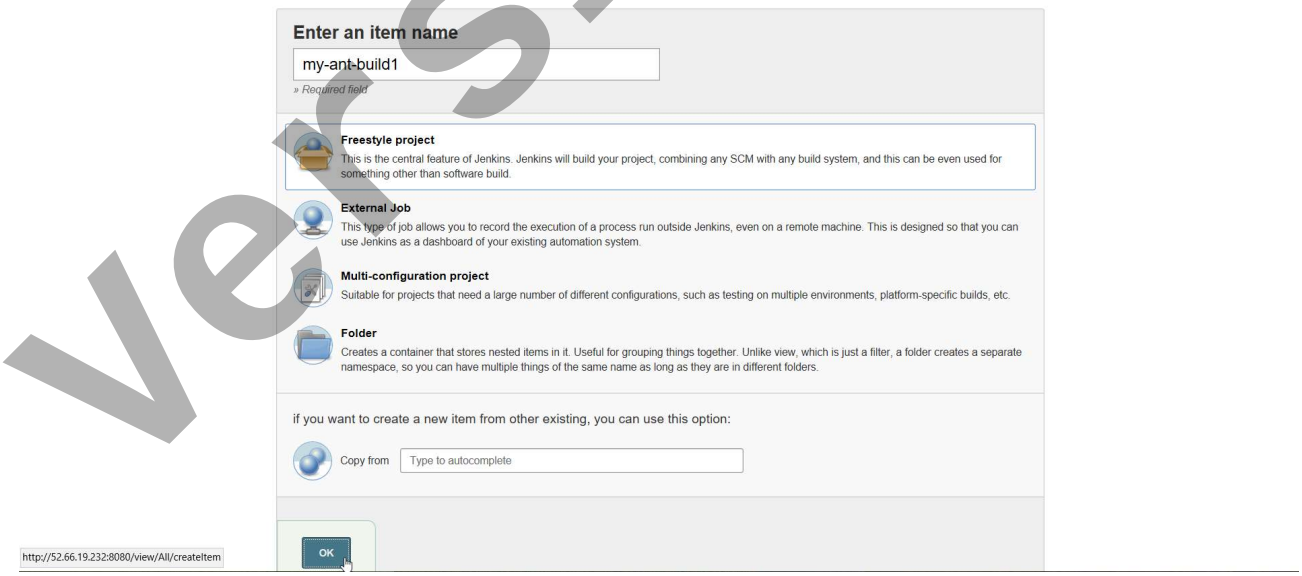
Then "**Save**" the configuration.

Now Jenkins is ready for Maven builds.

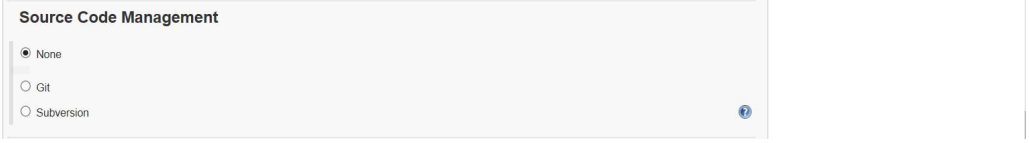
Create ANT Build Job in Jenkins:

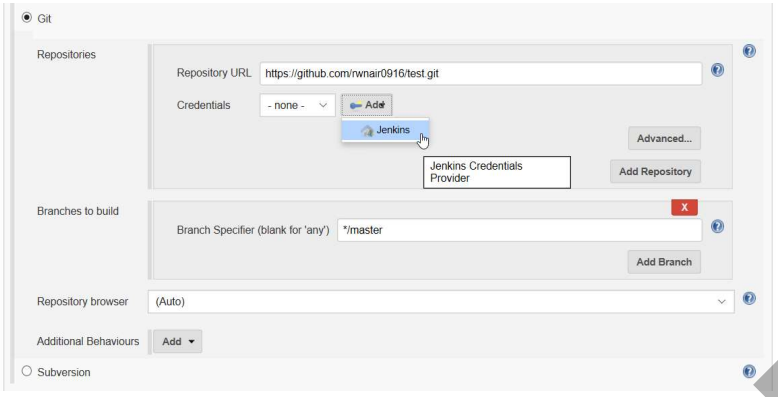
You can create a build job using Jenkins with Ant and Git in backend.

Create a "**New Item**" → Then "**Enter an Item Name**" → Select "**Freestyle Project**" → Then "**OK**"

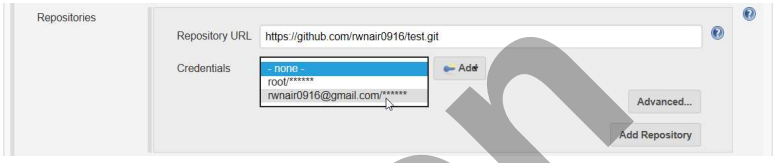


Then come to "**Source Code Management**"




Git
Repositories
Repository URL
https://github.com/rwnair0916/test.git
Credentials
- none -
Add
Jenkins
Jenkins Credentials Provider
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Subversion


Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
rwnair0916@gmail.com
Password

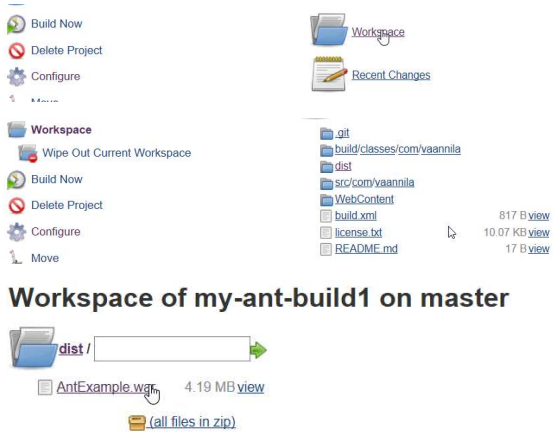

Repositories
Repository URL
https://github.com/rwnair0916/test.git
Credentials
- none -
root/******
rwnair0916@gmail.com/******
Add
Advanced...
Add Repository


Build
Add build step
Execute Windows batch command
Execute shell
GitHub PR: set 'pending' status
Invoke Ant
Invoke top-level Maven targets
Set build status to "pending" on GitHub commit


Build
Invoke Ant
Ant Version
Ant V1.9.7
Targets
Advanced...
Add build step


Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Move
Build History
trend
find
RSS for all
RSS for failures
Project my-ant-build1
Workspace
Recent Changes
Permalinks
add description
Disable Project

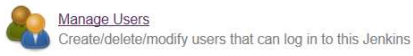
You can see that build was successful. You can verify the build from the following



You can download the WAR file and deploy in tomcat. Or you can invoke another deploy job from Jenkins directly.

Manage Users in Jenkins:

After login to Jenkins click on "**Manage Jenkins**"



Then click on 

.

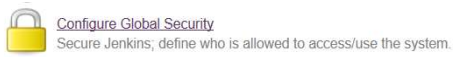
Then fill in the details.

You can customize the access to these users by using role based.

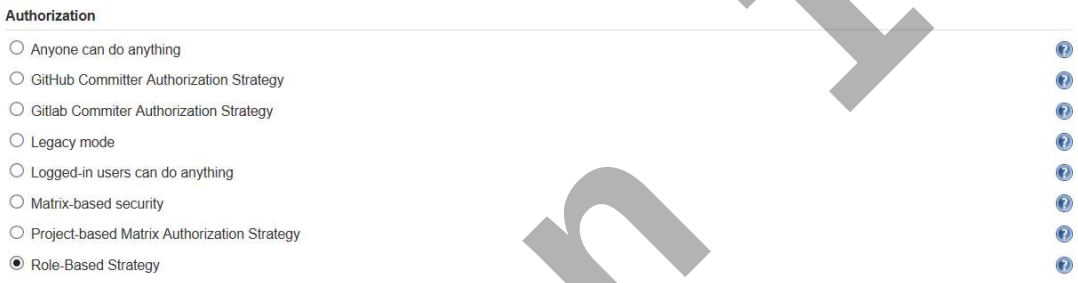
For that you need to install a Plugin "**Role-based Authorization Strategy**"

After that enable Role Based configuration in Jenkins Authorization.

Go to "**Manage Jenkins**" then click on "**Configure Global Security**".

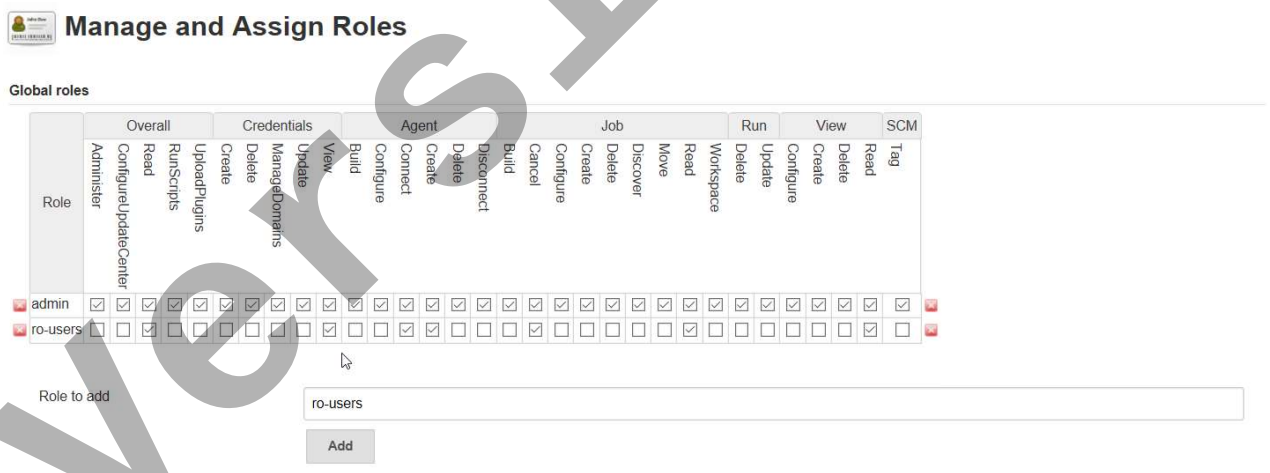


Then enable "**Role-Based Strategy**" under "**Authorization**".

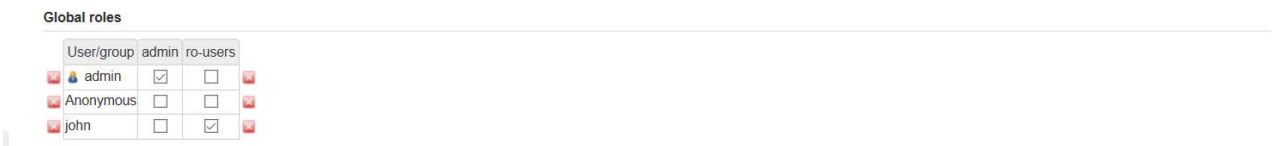


Then Save, Now you can see another option gets enabled under "**Manage Jenkins**".

Click on "**Manage and Assign Roles**" → "**Manage Roles**" → add a role using "**Role to add**"



Now map that role to the user "Manage Jenkins" → "**Manage and Assign Roles**" → "**Assign Roles**"



Now login as “John” user and you can verify that he doesn’t have much privileges like “admin” user.

…. ☺ .

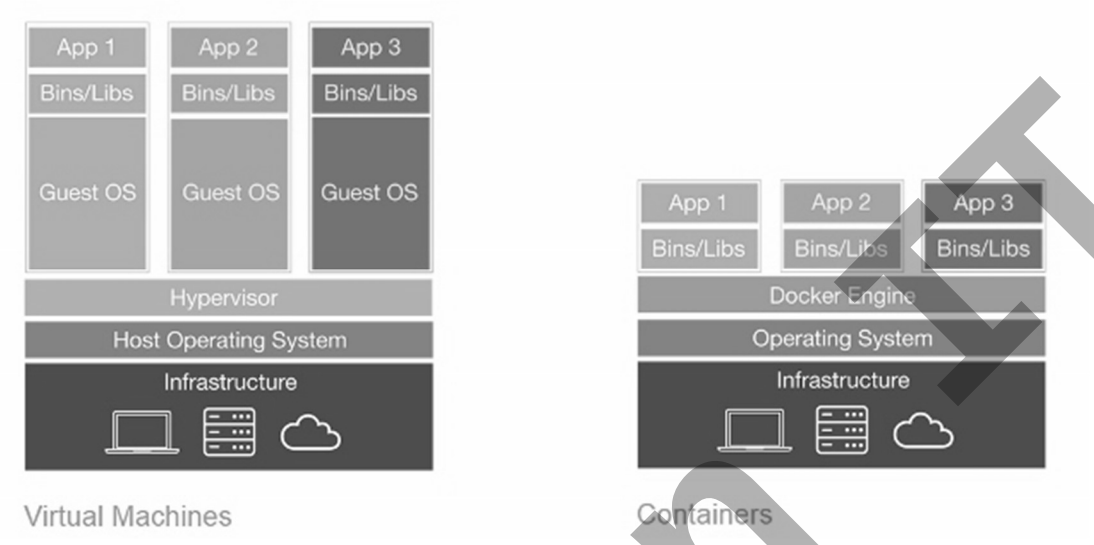
Issues

- Code works in development box but not in production.
- Developer develops always thinking about future and more interested in new features.
- Code comes out of such non-standard environments may or may not work in higher environments.
- Definitely the code works in developer Workstation as he was fixing the issues while developing the code.
- Missing the steps to transition while sending information to operations team.

Solution

- For these reasons we don't want to trust the humans, rather than that bring computers and tools to take care of that automation.
- Ideally we need to have development and production as same environments.
- So this problem can be solved by VM's and Containers.

VM vs Container



Docker Containers

- Light weight
- Less Secure
- Common OS
- Share Common resources
- More Performance

LINUX CONTAINERS

You can use the following command to install the linux containers and its dependencies.

 # yum -y install lxc lxc-templates libcap-devel libcgroup busybox wget bridge-utils libvirt

You can start the service of Linux containers now after successful installation.

 # systemctl start libvirtd

 # systemctl start lxc.service

You can verify the service running properly or not using the following command.

 # lxc-checkconfig

You can get the installed templates in the following location.

```
[root@lxcbox ~]# ls -l /usr/share/lxc/templates
total 340
-rwxr-xr-x 1 root root 10557 Nov 15  2015 lxc-alpine
-rwxr-xr-x 1 root root 13534 Nov 15  2015 lxc-altlinux
-rwxr-xr-x 1 root root 10556 Nov 15  2015 lxc-archlinux
-rwxr-xr-x 1 root root  9878 Nov 15  2015 lxc-busybox
-rwxr-xr-x 1 root root 29149 Nov 15  2015 lxc-centos
-rwxr-xr-x 1 root root 10486 Nov 15  2015 lxc-cirros
-rwxr-xr-x 1 root root 17354 Nov 15  2015 lxc-debian
-rwxr-xr-x 1 root root 17757 Nov 15  2015 lxc-download
-rwxr-xr-x 1 root root 49319 Nov 15  2015 lxc-fedora
-rwxr-xr-x 1 root root 28253 Nov 15  2015 lxc-gentoo
-rwxr-xr-x 1 root root 13962 Nov 15  2015 lxc-openmandriva
-rwxr-xr-x 1 root root 14046 Nov 15  2015 lxc-opensuse
-rwxr-xr-x 1 root root 35540 Nov 15  2015 lxc-oracle
-rwxr-xr-x 1 root root 11868 Nov 15  2015 lxc-plamo
-rwxr-xr-x 1 root root  6851 Nov 15  2015 lxc-sshd
-rwxr-xr-x 1 root root 23494 Nov 15  2015 lxc-ubuntu
-rwxr-xr-x 1 root root 11349 Nov 15  2015 lxc-ubuntu-cloud
[root@lxcbox ~]#
```

To create a new container using any of the template then run the following command.

```
[root@lxcbox ~]# lxc-create -t centos -n demo_centos
Host CPE ID from /etc/os-release: cpe:/o:redhat:enterprise_linux:7.2:GA:server
This is not a CentOS or Redhat host and release is missing, defaulting to 6 use -R|--release t
o specify release
```

Then it starts installing the packages for that container.

Once after installation a temporary root password will be set to the system and you can reset the password using the following commands given during the installation.

```
The temporary root password is stored in:

        '/var/lib/lxc/demo_centos/tmp_root_pass'


The root password is set up as expired and will require it to be changed
at first login, which you should do as soon as possible.  If you lose the
root password or wish to change it without starting the container, you
can change it from the host by running the following command (which will
also reset the expired flag):

        chroot /var/lib/lxc/demo_centos/rootfs passwd
```

Change the password by running the same above command.

```
[root@lxcbox ~]# chroot /var/lib/lxc/demo_centos/rootfs passwd
Changing password for user root.
New password:
BAD PASSWORD: it is based on a dictionary word
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@lxcbox ~]#
```

You can start the container using the following command.

 # lxc-start -n demo_centos -d

You can see the containers running on the machine using the following commands

 # lxc-ls

 # lxc-ls -l

 # lxc-ls --active

You can launch the console of container using the following command.

```
[root@lxcbox ~]# lxc-console -n demo_centos

Connected to tty 1
Type <Ctrl+a q> to exit the console, <Ctrl+a Ctrl+a> to enter Ctrl+a itself


CentOS release 6.8 (Final)
Kernel 3.10.0-327.36.2.el7.x86_64 on an x86_64

demo_centos login:
```

You can see the status of the container using the following command.

```
[root@lxcbox ~]# lxc-info -n demo_centos
Name:           demo_centos
State:          RUNNING
PID:            3672
IP:             192.168.122.26
CPU use:        0.75 seconds
BlkIO use:      4.00 KiB
Memory use:     2.60 MiB
KMem use:       0 bytes
Link:           vethNWEXMI
 TX bytes:      1.76 KiB
 RX bytes:      5.95 KiB
 Total bytes:   7.71 KiB
[root@lxcbox ~]#
```

You can stop the container using the following command.

 # lxc-stop -n demo_centos

You can clone a Container using the following command.

```
[root@lxcbox ~]# lxc-info -n demo_centos
Name:           demo_centos
State:          STOPPED
```

```
[root@lxcbox ~]# lxc-clone demo_centos new_centos
Created container new_centos as copy of demo_centos
[root@lxcbox ~]#
```

Ensure you stop the Container before cloning.

To remove the container completely use the following command.

```
[root@lxcbox ~]# lxc-ls -l
drwxrwx--- 3 root root 52 Oct 24 19:36 demo_centos
drwxrwx--- 3 root root 32 Oct 24 19:49 new_centos
[root@lxcbox ~]# lxc-destroy -n new_centos
[root@lxcbox ~]# lxc-ls -l
drwxrwx--- 3 root root 52 Oct 24 19:36 demo_centos
[root@lxcbox ~]#
```

DOCKER CONTAINERS

You can install Docker containers by setting the yum repository.

```
[root@dockers ~]# cat /etc/yum.repos.d/docker.repo
[dockerrepo]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/centos/7/
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
[root@dockers ~]#
```

You can install the docker service.

 # yum install docker-engine -y

You can start and enable the docker service.

 # systemctl enable docker

 # systemctl start docker

Now you can check the docker version.

```
[root@dockers ~]# docker version
Client:
 Version:      1.12.3
 API version:  1.24
 Go version:   go1.6.3
 Git commit:   6b644ec
 Built:
 OS/Arch:      linux/amd64

Server:
 Version:      1.12.3
 API version:  1.24
 Go version:   go1.6.3
 Git commit:   6b644ec
 Built:
 OS/Arch:      linux/amd64
[root@dockers ~]#
```

You can run the following command to ensure that docker is working correctly.

```
[root@dockers ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world

c04b14da8d14: Pull complete
Digest: sha256:0256e8a36e2070f7bf2d0b0763dbabdd67798512411de4cdcf9431a1feb60fd9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Now you can run another docker along with some command.

```
[root@dockers ~]# docker run centos ls
anaconda-post.log
bin
dev
etc
home
lib
lib64
lost+found
```

The above command has run inside the docker and then it came out after once it is done.

Sometimes this cause some issue, You can run the following command and see that the issue with running the command.

```
[root@dockers ~]# docker run centos yum install httpd
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
```

```
Transaction Summary
================================================================================
Install  1 Package (+5 Dependent packages)

Total download size: 24 M
Installed size: 31 M
Is this ok [y/d/N]: Exiting on user command
Your transaction was saved, rerun it with:
 yum load-transaction /tmp/yum_save_tx.2016-11-02.17-09.i6fYNZ.yumtx
```

So some commands need user intervention and hence you can run the commands in interactive mode.

```
[root@dockers ~]# docker run -i -t centos yum install httpd
Loaded plugins: fastestmirror, ovl
base                                                     | 3.6 kB  00:00:00
```

```
Total download size: 24 M
Installed size: 31 M
Is this ok [y/d/N]: y
```

```
Installed:
  httpd.x86_64 0:2.4.6-40.el7.centos.4

Dependency Installed:
  apr.x86_64 0:1.4.8-3.el7
  apr-util.x86_64 0:1.5.2-6.el7
  centos-logos.noarch 0:70.0.6-3.el7.centos
  httpd-tools.x86_64 0:2.4.6-40.el7.centos.4
  mailcap.noarch 0:2.1.41-2.el7

Complete!
```

So now after our command runs successfully and then the docker is stopped.

You can run the docker in background using the following command.

```
[root@dockers ~]# docker run -itd centos
c2450a8c0c13890ec895956f2473c6601210f05a5abedc4bc2a53924b9f5693a
[root@dockers ~]#
```

You can check the running dockers using the following command.

```
[root@dockers ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
              PORTS               NAMES
c2450a8c0c13        centos              "/bin/bash"         32 seconds ago      Up 29
seconds                                 sad_fermat
[root@dockers ~]#
```

You can get all running and stopped containers using the following command.

```
[root@dockers ~]# docker ps -a
```

To get the latest container information you can use the following command.

```
[root@dockers ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
              PORTS               NAMES
c2450a8c0c13        centos              "/bin/bash"         3 minutes ago       Up 3 m
inutes                                  sad_fermat
```

To get only the container ID which was the latest one then you can run the following command.

```
[root@dockers ~]# docker ps -l -q
c2450a8c0c13
[root@dockers ~]#
```

If you need all the container ID's then you can use the following command.

```
[root@dockers ~]# docker ps -a -q
c2450a8c0c13
3860f942919e
be0e62b87b7f
854cec3d74ff
ac9530eed4ce
1a7b1cba78ad
cefb81d6b5ab
[root@dockers ~]#
```

You can get the output table with only the fields you want using the following command.

```
[root@dockers ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
              PORTS               NAMES
c2450a8c0c13        centos              "/bin/bash"         9 minutes ago       Up 9 m
inutes                                  sad_fermat
[root@dockers ~]# docker ps --format "{{.ID}}: {{.Status}}"
c2450a8c0c13: Up 9 minutes
[root@dockers ~]#
```

You can stop the running docker using the following command.

```
[root@dockers ~]# docker ps -q
c2450a8c0c13
[root@dockers ~]# docker stop c2450a8c0c13
c2450a8c0c13
[root@dockers ~]# docker ps -q
[root@dockers ~]#
```

Likewise, you can also kill a running docker using the following command.

```
[root@dockers ~]# docker ps -q
9f74ab55934f
[root@dockers ~]# docker kill 9f74ab55934f
9f74ab55934f
[root@dockers ~]# docker ps -q
[root@dockers ~]#
```

You can get the total docker information using the following command.

```
[root@dockers ~]# docker inspect 9f74ab55934f
```

You will receive the output in JSON format from the above command.

If you want to check the logs of a container using the following command.

```
[root@dockers ~]# docker logs f6f9ef580751
```

You can login to console of any docker by using attach option.

```
[root@dockers ~]# docker ps -q
f6f9ef580751
[root@dockers ~]# docker attach f6f9ef580751
[root@f6f9ef580751 /]# yum update
```

To come out of docker you need to press **Ctrl + p** followed by **Ctrl + q.**

To remove a docker you need to use the following command.

```
[root@dockers ~]# docker rm f6f9ef580751
f6f9ef580751
[root@dockers ~]#
```

To remove all the images the you use the following command.

```
[root@dockers ~]# docker rm `docker ps -a -q`
9f74ab55934f
c2450a8c0c13
3860f942919e
be0e62b87b7f
854cec3d74ff
ac9530eed4ce
1a7b1cba78ad
cefb81d6b5ab
[root@dockers ~]#
```

To check the downloaded container images, you can use the following command.

```
[root@dockers ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              980e0e4c79ec        8 weeks ago         196.7 MB
hello-world         latest              c54a2cc56cbb        4 months ago        1.848 kB
[root@dockers ~]#
```

You can run a container from an image and try to modify the container.

```
[root@dockers ~]# docker run -i -t centos bash
[root@86ac89d44b61 /]# yum install httpd -y
```

```
[root@86ac89d44b61 /]# grep ^Listen /etc/httpd/conf/httpd.conf
Listen 8000
[root@86ac89d44b61 /]#
```

```
[root@dockers ~]# docker diff `docker ps -l -q`
A /boot
A /boot/grub
A /boot/grub/splash.xpm.gz
C /etc
```

To make the changes to File System then

```
[root@dockers ~]# docker commit `docker ps -lq` versionit/httpd
sha256:d6d5e6605e8681c42053a025317e05e4d97f05936c1401aaf8bf5d76d167d13e
[root@dockers ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
versionit/httpd     latest              d6d5e6605e86        8 seconds ago       338.5 MB
centos              latest              980e0e4c79ec        8 weeks ago         196.7 MB
hello-world         latest              c54a2cc56cbb        4 months ago        1.848 kB
[root@dockers ~]#
```

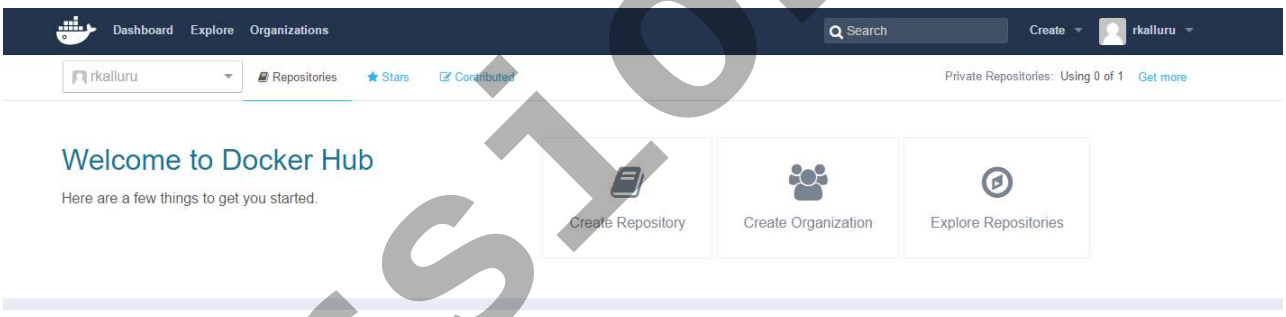
Docker Containers in Docker Hub:

https://hub.docker.com/

This is the docker images public repository, as per previous example you can see hello-world was downloaded from somewhere else and downloaded on local box, it means somewhere location is nothing but your docker hub repo.

Now go head and signup docker hub repository, it is free of cost.

Once after you login you can see the following screen.



You can search for the official and other user docker images in this website.

You can pull and use any repository here using the following command.

```
[root@lxcbox ~]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd

43c265008fae: Pull complete
2421250c862c: Pull complete
f267bf8fc4ac: Pull complete
48efff98b4ba: Pull complete
acb686eb7ab7: Pull complete
Digest: sha256:9b29c9ba465af556997e6924f18efc1bbe7af0dc1b3f11884010592e700ddb09
Status: Downloaded newer image for httpd:latest
```

Let us create our own image and push it to Docker Hub repository.

```
[root@lxcbox ~]# mkdir my-docker
[root@lxcbox ~]# cd my-docker/
[root@lxcbox my-docker]# vim Dockerfile
[root@lxcbox my-docker]# cat Dockerfile
From centos:7
MAINTAINER "rkalluru" <raghukx@gmail.com>
RUN yum -y update
RUN yum install -y tomcat*
RUN yum install -y https://s3.ap-south-1.amazonaws.com/versionithyd/jdk-8u101-linux-x64.rpm
[root@lxcbox my-docker]#
```

Now start building the Docker.

```
[root@lxcbox my-docker]# docker build -t 'tomcat-image:latest' .
Sending build context to Docker daemon 2.048 kB
Step 1 : FROM centos:7
7: Pulling from library/centos

Step 2 : MAINTAINER "rkalluru" <raghukx@gmail.com>
 ---> Running in b7fc527c7456
 ---> 37725b7a1be4
Removing intermediate container b7fc527c7456

Step 3 : RUN yum -y update
 ---> Running in f2451a74c7b4

Step 4 : RUN yum install -y tomcat*
 ---> Running in bd89b2dde3e5

Step 5 : RUN yum install -y https://s3.ap-south-1.amazonaws.com/versionithyd/jdk-8u101-linux-x64.rp
m
 ---> 734464042cb3
Removing intermediate container a66a84da851b
Successfully built 734464042cb3
```

You can check the new image has been added.

```
[root@lxcbox ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
tomcat-image        latest              734464042cb3        About a minute ago   1.062 GB
centos              7                   980e0e4c79ec        6 weeks ago          196.7 MB
centos              latest              980e0e4c79ec        6 weeks ago          196.7 MB
[root@lxcbox ~]#
```

You can launch new containers using our new image.

```
[root@lxcbox ~]# docker run --name tomcat1 -idt 734464042cb3
c05ba09c271a4c5cf7e2191912bd493b0814f3dbcfdd0d159e513b8d39a98779
[root@lxcbox ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
 PORTS               NAMES
c05ba09c271a        734464042cb3        "/bin/bash"         4 seconds ago       Up 3 seconds
                     tomcat1
```

Now let's push this image to our docker hub.

First we need to change tag name with repository with docker hub username like this:

```
[root@lxcbox ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
tomcat-image        latest              734464042cb3        6 minutes ago       1.062 GB
centos              7                   980e0e4c79ec        6 weeks ago         196.7 MB
centos              latest              980e0e4c79ec        6 weeks ago         196.7 MB
[root@lxcbox ~]# docker tag 734464042cb3 rkalluru/tomcat-java:latest
[root@lxcbox ~]#
```

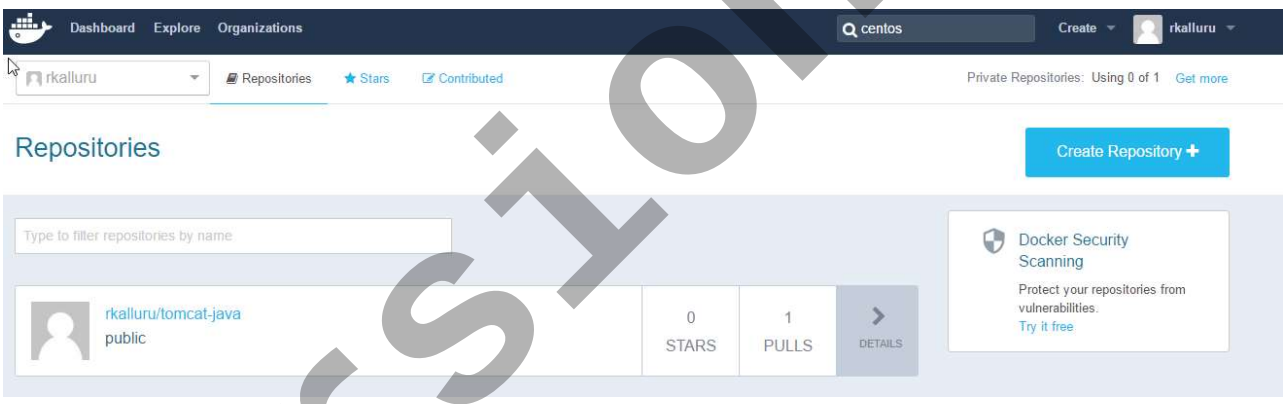
Second login to Docker Hub to push the Docker.

```
[root@lxcbox ~]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, h
ead over to https://hub.docker.com to create one.
Username: rkalluru
Password:
Login Succeeded
[root@lxcbox ~]#
```

Push the docker from your local machine to Docker Hub.

```
[root@lxcbox ~]# docker push rkalluru/tomcat-java:latest
The push refers to a repository [docker.io/rkalluru/tomcat-java]
ff18b75aaadf: Pushed
e77e9ae32829: Pushed
4e530017bef2: Pushed
0aeb287b1ba9: Mounted from library/centos
latest: digest: sha256:31cae066c847e239ca0f6ba619373234cd7e8d593c1f34348e64c61365996932 size: 1166
[root@lxcbox ~]#
```

You can see your image in Docker Hub.



What is Ansible

- Ansible is capable of handling many powerful automation tasks with the flexibility to adapt to many environments and workflows. With Ansible user can quickly getup and running to do real work.
- Ansible does configuration management
- Ansible can also be used for deployment purpose even.

Why Ansible

- Open Source and Enterprise Support.
- Provisioning.
- Configuration Management.
- Application Deployment
- Simple & Compliance.
- Server Orchestration.
- Have almost lot of modules.
- Support from RedHat.

Why Ansible

- Agentless
- Simple
- **Use YAML**
- Cross Platform
- Dynamic Inventory

YAML

- YAML is a human friendly data serialization standard for all programming languages.
- Data Serialization language : A language used to convert or represent structured data or objects as a series of characters that can be stored on a disk.
- Well Known example are
  - CSV - Comma Separated Values
  - XML - Extensible Markup Language
  - JSON - Java Script Object Notation
  - YAML – Yet Another Markup Language

YAML

- Unlike XML which is not easily readable by humans, YAML was created to be human-friendly and integrate easily wit modern programming languages.
- Unlike with XML, YAML was intended to simplify the viewing and understanding of config files, log files, object persistence and messaging, it allow the programmer more time programming and less time worrying about formatting data.

Basic YAML Syntax Rules

- Document begins with --- and end with …
- Indentation of lines denotes the structure within the document.
- Comments begin with #
- Members of lists begin with –
- Key value pairs use the following syntax.

  <key>:<value>

Basic YAML Example

```
---
Student-Id: 11223344
First-Name: John
Last-Name: Smith

Phone-numbers:
  - 281.555.7689
  - 713.555.8967
  - 832.555.9980

Addresses:
  - street: 123 Main St.
    city: Houston
    state: TX
...
```

Ansible Terminology

- **Inventory** : A list of hosts, groups and variables
- **Modules** : Actually do the work
- **Plugins** : Call back, Actions and other hooks
- **Facts** : Data gathered from target hosts
- **Playbooks** : A collection of plays
- **Plays** : Loops over a list of tasks mapped to a list of hosts
- **Tasks** : Invokes a module to do the work

Ansible Configuration File

- **Changes can be made and used in a configuration file which will be processed in the following order:**

*** ANSIBLE_CONFIG (an environment variable)**
*** ansible.cfg (in the current directory)**
*** .ansible.cfg (in the home directory)**
*** /etc/ansible/ansible.cfg**

Ansible Host Inventory Parameters

*** ansible_port=2222**
*** ansible_user=admin**
*** ansible_ssh_private_key_file=/opt/ec2.pem**
*** ansible_python_interpreter=/usr/local/bin/python**

Ansible PlayBook

```
---
-hosts: dev_servers
  user: ubuntu
  sudo: true
  tasks:
    - name: Install Web Server
      yum: pkg=httpd state=installed
…
```

Task to perform on the nodes

Ansible PlayBook

```
-hosts: web    group name from the inventory
  user: ec2-user  Server Auth
  sudo: true
  roles:
    - webserver Role which should be installed on the server
```

Ansible Commonly Used Facts

- "ansible_distribution”
- "ansible_distribution_version”
- "ansible_distribution_major_version"
- "ansible_os_family”
- "ansible_kernel”
- "ansible_domain”
- "ansible_architecture"

Ansible Variables

```
-hosts:
   vars:
      pack_name: httpd
      serv_name: httpd
   tasks:
      - name: Install {{ pack_name }}
        yum: pkg={{ pack_name }} state=installed
      - name: Start {{ serv_name }}
        service: name={{ serv_name }} state=started
```

Ansible Variable Files

```
-hosts:
   vars:
      pack_name: httpd
   vars_files:
      - web_vars.yml
   tasks:
      - name: Install {{ pack_name }}
        yum: pkg={{ pack_name }} state=installed
      - name: Start {{ serv_name }}
        service: name={{ serv_name }} state=started
```

Ansible Variable from Prompt

```
-hosts:
   vars:
      pack_name: httpd
   vars_prompt:
      - name: web_pass
        prompt: Web Server Password:
    tasks:
      - name: Install {{ pack_name }}
        yum: pkg={{ pack_name }} state=installed
      - name: Start {{ serv_name }}
        service: name={{ serv_name }} state=started
```

Ansible Install Package PlayBook

```
-hosts:
     vars_prompt:
       - name: pack_name
          prompt: Enter the Package Name
     tasks:
      - name: Install {{ pack_name }}
        yum: pkg={{ pack_name }} state=installed
```

Ansible Install Package PlayBook

```
-hosts:
     vars_prompt:
       - name: pack_name
         prompt: Enter the Package Name
         default: telnet
         private: no
     tasks:
      - name: Install {{ pack_name }}
        yum: pkg={{ pack_name }} state=installed
```

Ansible Task Handlers

```
-hosts:
   vars:
     pack_name: httpd
   tasks:
     - name: Install {{ pack_name }}
       yum: pkg={{ pack_name }} state=installed
       notify: Restart HTTPD
   handlers:
     - name: Restart HTTPD
       action: service name=httpd state=restarted
```

Ansible Playbook Loops

```
-hosts:
   tasks:
     - name: Add a list off users
       user: name={{ item }} state=present
       with_items:
          - user1
          - user2
          - user3
```

Play Book :: Conditions

```
---
- hosts: webserver
  tasks:
   - name: Install httpd redhat
     yum: name=httpd state=installed
     when: ansible_distribution == "RedHat"
   - name: Install httpd ubuntu
     apt: name=apache2 state=installed
     when: ansible_distribution == "Ubuntu"
   - name: Start httpd
     service: name=httpd state=started
     when: ansible_distribution == "RedHat"
   - name: Start apache2
     service: name=apache2 state=started
     when: ansible_distribution == "Ubuntu"
...
```

Play Book :: until

```
--- # UNTIL EXAMPLE
- hosts: webserver
  become: true
  user: ec2-user
  tasks:
    - name: Installing Web Server
      yum: pkg=httpd state=latest
    - name: Verify Service Status
      shell: systemctl status httpd
      register: result
      until: result.stdout.find("active (running)") != -1
      retries: 5
      delay: 5
...
```

Ansible Vault

```
ansible-vault create mydata.yml

cat mydata.yml

ansible-vault edit mydata.yml

ansible-vault rekey mydata.yml

ansible-vault decrypt mydata.yml

cat mydata.yml

ansible-vault encrypt mydata.yml
```

Ansible Play in Playbook

```
$ cat playbooks/plays/package.yml
-name: Install the telnet client
  yum: pkg=telnet state=latest

$ cat playbooks/pack_include.yml
---
-hosts: webserver
 user: root
 tasks:
   - include: plays/package.yml
   - name: Verify the telnet package
     raw: yum list installed |grep telnet >/tmp/pack
```

Ansible Roles

```
roles/
    role-name/            # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the jinja template resource
            sample.conf.j2#  <------- templates end in .j2
        files/            #
            sample.conf   #  <-- files for use with the copy resource
            script2.sh    #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
```

Ansible Role Path

```
$ cat ansible.cfg
# additional paths to search for roles in, colon separated
#roles_path    = /etc/ansible/roles
roles_path = /root/ansible/v2/roles

$ cd /root/ansible/v2/roles
$ mkdir
webserver/{tasks,handlers,templates,files,vars,defaults,met
a}
```

ANSIBLE

Install EPEL repository and then you can go with Ansible package installation.

 # yum -y install http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-8.noarch.rpm

You can now install the Ansible Community edition by using the following command.

 # yum -y install ansible

Consider you have three machines already with SSH service configured.

Here I am using the following IP addresses.

172.31.20.47

172.31.20.48

172.31.20.49

Just to ensure that you are able to login to all the servers, Execute the following command. Provide the valid password and see whether you are able to get the o/p from all the servers.

```
[versionit@ansible ~]$ for i in 172.31.20.47 172.31.20.48 172.31.20.49 ;do ssh -o StrictHostKeyChecking
=no $i -l ec2-user uptime ;done
ec2-user@172.31.20.47's password:
 04:24:02 up  3:25,  0 users,  load average: 0.00, 0.01, 0.05
ec2-user@172.31.20.48's password:
 04:24:05 up  3:25,  0 users,  load average: 0.00, 0.01, 0.05
ec2-user@172.31.20.49's password:
 04:24:07 up  3:25,  0 users,  load average: 0.00, 0.01, 0.05
[versionit@ansible ~]$
```

Copy /etc/ansible directory to your home directory and

Here I logged in as versionit user.

```
[versionit@ansible ~]$ cp -r /etc/ansible .
[versionit@ansible ~]$ ls
ansible
[versionit@ansible ~]$
```

You have hosts file as default file for host inventory. I am creating a new file for inventory and put all the IP addresses under that.

```
[versionit@ansible ansible]$ cat inventory
172.31.20.47
172.31.20.48
172.31.20.49
[versionit@ansible ansible]$
```

```
[versionit@ansible ansible]$ cat inventory
[web]
172.31.20.47
[dbserver]
172.31.20.48
[tomcat]
172.31.20.49
[versionit@ansible ansible]$
```

You can also group the nodes as shown in above. Even you can group a group also using the following syntax.

```
[versionit@ansible ansible]$ cat inventory
[web]
172.31.20.47
[dbserver]
172.31.20.48
[tomcat]
172.31.20.49
[app_and_db:children]
tomcat
dbserver
[versionit@ansible ansible]$
```

Now you can go with running the ansible commands on command line.

```
[versionit@ansible ansible]$ ansible -u ec2-user -i inventory all -m ping -k
SSH password:
172.31.20.49 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
172.31.20.48 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
172.31.20.47 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[versionit@ansible ansible]$
```

You can get all the options explained with ansible --help options, but here we are using the following options.

-u → To specify the SSH username.

-k → To ask for password prompt.

-i → To specify the custom inventory file

-m → To run the specific module.

To get all the modules then refer the following web page.

http://docs.ansible.com/ansible/list_of_all_modules.html

You can run the command on a particular group using the following command.

```
[versionit@ansible ansible]$ ansible -u ec2-user -i inventory web -m ping -k
SSH password:
172.31.20.47 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[versionit@ansible ansible]$
```

```
[versionit@ansible ansible]$ ansible -u ec2-user -i inventory app_and_db:children -m ping -k
SSH password:
172.31.20.48 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
172.31.20.49 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[versionit@ansible ansible]$
```

You can update the inventory file path in ansible.cfg so that ansible command will pick the file from configuration.

```
[versionit@ansible ansible]$ grep ^inventory ansible.cfg
inventory = /home/versionit/ansible/inventory
[versionit@ansible ansible]$
```

So without -i option you can run the command.

```
[versionit@ansible ansible]$ ansible web -u ec2-user -m ping -k
SSH password:
172.31.20.47 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[versionit@ansible ansible]$
```

In case if you have a key to connect to clients then you can use the key option.

```
[versionit@ansible ansible]$ ansible web -u ec2-user -m ping --private-key /tmp/ec2.pem
172.31.20.47 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[versionit@ansible ansible]$
```

Assume in some hosts you have a password and some machines you have key configured then we can handle with inventory parameters. You can get the full length of parameters in the following link.

http://docs.ansible.com/ansible/intro_inventory.html#list-of-behavioral-inventory-parameters

```
[versionit@ansible ansible]$ cat inventory
[web]
172.31.20.47
[dbserver]
172.31.20.48 ansible_ssh_private_key_file=/tmp/ec2.pem
172.31.20.47  ansible_ssh_pass=password
```

You can get the connections done now one with key and another once with password.

```
[versionit@ansible ansible]$ ansible dbserver -m ping -u ec2-user
172.31.20.48 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
172.31.20.47 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

You can verify that by enabling the verbose mode. Using -vvv option.

You can also get the list of hosts using --list-hosts option.

```
[versionit@ansible ansible]$ ansible dbserver --list-hosts
  hosts (2):
    172.31.20.48
    172.31.20.47
[versionit@ansible ansible]$ ansible all --list-hosts
  hosts (3):
    172.31.20.47
    172.31.20.49
    172.31.20.48
[versionit@ansible ansible]$ ansible web --list-hosts
  hosts (1):
    172.31.20.47
[versionit@ansible ansible]$
```

Now lets go with another module which is used to run the commands on the remote hosts.

So let us try to install a package on the remote nodes.

```
[versionit@ansible ansible]$ ansible web -u ec2-user -m shell -a 'yum install httpd -y'
172.31.20.47 | FAILED | rc=1 >>
Loaded plugins: amazon-id, rhui-lb, search-disabled-reposRepo rhui-REGION-client-config-server-7 forced
 skip_if_unavailable=True due to: /etc/pki/rhui/cdn.redhat.com-chain.crt
Repo rhui-REGION-client-config-server-7 forced skip_if_unavailable=True due to: /etc/pki/rhui/product/r
hui-client-config-server-7.crt
Repo rhui-REGION-client-config-server-7 forced skip_if_unavailable=True due to: /etc/pki/rhui/rhui-clie
nt-config-server-7.key
You need to be root to perform this command.
```

So here the command needs to be run as root user and hence we are getting that error. So here you need another option to handle that which sudo.

```
[versionit@ansible ansible]$ ansible web -u ec2-user -m shell -a 'yum install httpd -y' -s
172.31.20.47 | SUCCESS | rc=0 >>
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
Resolving Dependencies
--> Running transaction check
```

Here you can use -s or --sudo option.

Usually Ansible is going to collect the facts and you can see those facts using **setup** module.

```
[versionit@ansible ansible]$ ansible web -m setup -u ec2-user
172.31.20.47 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.31.20.47"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::c6c:51ff:fed0:db5a"
        ],
```

You can filter and get some specific facts.

```
[versionit@ansible ansible]$ ansible web -u ec2-user -m setup -a 'filter=*ipv4*'
172.31.20.47 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.31.20.47"
        ],
        "ansible_default_ipv4": {
            "address": "172.31.20.47",
            "alias": "eth0",
            "broadcast": "172.31.31.255",
            "gateway": "172.31.16.1",
            "interface": "eth0",
            "macaddress": "0e:6c:51:d0:db:5a",
            "mtu": 9001,
            "netmask": "255.255.240.0",
            "network": "172.31.16.0",
            "type": "ether"
        }
    },
    "changed": false
}
```

You can start keeping the configuration in a file and run that file across the nodes.

Create Playbooks directory and put all the files under that directory.

```
[versionit@ansible ansible]$ cat playbooks/http.yml
---
- hosts: web
  user: ec2-user
  become: true
  tasks:
   - name: Install httpd
     yum: name=httpd state=installed

   - name: Start httpd
     service: name=httpd state=started
...
[versionit@ansible ansible]$
```

Run the following command to run the playbook on the servers.

```
[versionit@ansible ansible]$ ansible-playbook playbooks/http.yml

PLAY [web] *********************************************************************

TASK [setup] *******************************************************************
ok: [172.31.20.47]

TASK [Install httpd] ***********************************************************
ok: [172.31.20.47]

TASK [Start httpd] *************************************************************
ok: [172.31.20.47]

PLAY RECAP *********************************************************************
172.31.20.47               : ok=3    changed=0    unreachable=0    failed=0

[versionit@ansible ansible]$
```

You can do a dry run with --check option to check is there anything changed.

```
[versionit@ansible ansible]$ ansible-playbook playbooks/http.yml --check

PLAY [web] *********************************************************************

TASK [setup] *******************************************************************
ok: [172.31.20.47]

TASK [Install httpd] ***********************************************************
changed: [172.31.20.47]

TASK [Start httpd] *************************************************************
fatal: [172.31.20.47]: FAILED! => {"changed": false, "failed": true, "msg": "Could not find the request
ed service \"'httpd'\": "}
        to retry, use: --limit @/home/versionit/ansible/playbooks/http.retry

PLAY RECAP *********************************************************************
172.31.20.47               : ok=2    changed=1    unreachable=0    failed=1

[versionit@ansible ansible]$
```

```
[versionit@ansible ansible]$ ansible-playbook playbooks/http.yml --list-tasks

playbook: playbooks/http.yml

  play #1 (web): web    TAGS: []
    tasks:
      Install httpd     TAGS: []
      Start httpd       TAGS: []
[versionit@ansible ansible]$
```