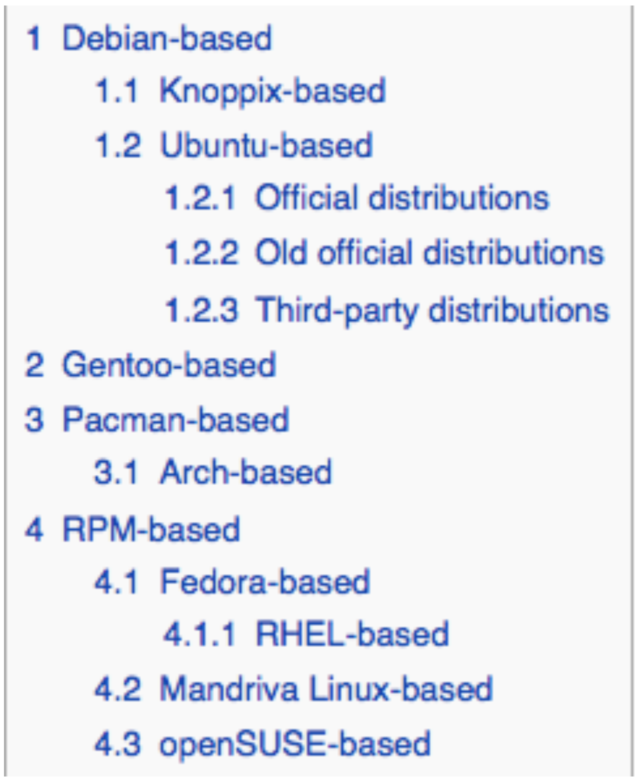
———————— Software installation commands for Linux distributions
Two major types -
1. Debian bases systems
2. RPM based systems.

Debian/Ubuntu	APT / DPKG (Apt-get)	
Fedora/ Redhat / CentOS	YUM using RPM package	RPM is the Red Hat Package Manager
SUSE	YAST / RPM packages	YAST - Yet another software tool

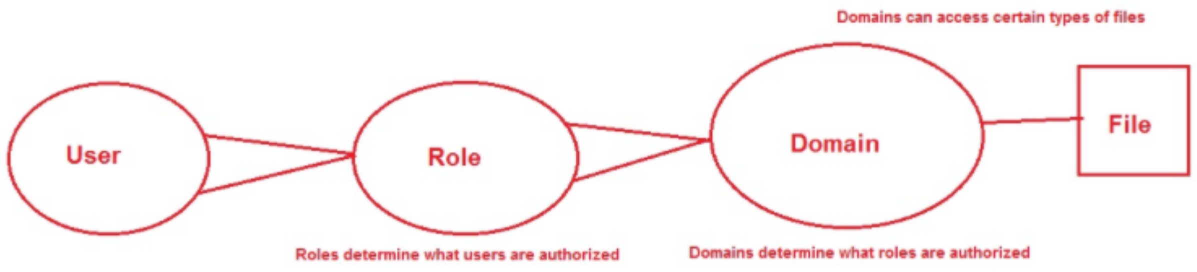

1 Debian-based
1.1 Knoppix-based
1.2 Ubuntu-based
1.2.1 Official distributions
1.2.2 Old official distributions
1.2.3 Third-party distributions
2 Gentoo-based
3 Pacman-based
3.1 Arch-based
4 RPM-based
4.1 Fedora-based
4.1.1 RHEL-based
4.2 Mandriva Linux-based
4.3 openSUSE-based

——————SElinux
https://www.digitalocean.com/community/tutorials/an-introduction-to-selinux-on-centos-7-part-1-basic-concepts

There are two directives in this file /etc/selinux/config. The SELINUX directive dictates the SELinux mode and Second is SELINUXTYPE determines the policy that will be used. A policy is what the name implies: a set of rules that define the security and access rights for (users, roles, processes(called subjects), and files(or port,directory or object)) in the system.

The purpose of SELinux is to secure how processes access files in a Linux environment. Without SELinux, a process or application like the Apache daemon will run under the context of the user that started it. So if your system is compromised by a rogue application that's running under the root user, the app can do whatever it wants because root has all-encompassing rights on every file.

The first part of security puts a *label* on each entity in the Linux system. A label is like any other file or process attribute (owner, group, date created etc.); it shows the *context* of the resource. So what's a context? Put simply, a context is a collection of security related information that helps SELinux make access control decisions. Everything in a Linux system can have a security context: a user account, a file, a directory, a daemon, or a port can all have their security contexts. However, security context will mean different things for different types of objects.


Domains can access certain types of files
User
Role
Domain
File
Roles determine what users are authorized
Domains determine what roles are authorized

A file security context is a *type* and a process security context is a *domain*.

A domain is the context within which an SELinux subject (process) can run. That context is like a wrapper around the subject. It tells the process what it can and can't do. For example, the domain will define what files, directories, links, devices, or ports are accessible to the subject.

process running within a particular domain can perform only certain operations on certain types of objects, is called Type Enforcement (TE)

SELinux policy is not something that replaces traditional DAC (Discretionary access control) security. If a DAC rule prohibits a user access to a file, SELinux policy rules won't be evaluated because the first line of defense DAC has already blocked access. SELinux security decisions come into play after DAC security has been evaluated. SELInux applied MAC (mandatory access controls)

Commands -
getenforce - Show current status of SELinux mode (enforcing, permissive or disabled)
sestatus - shows the SElinux policy store name
semodule -l - Shows list of SELinux modules loaded into the memory . It can be used for a number other tasks like installing, removing, reloading, upgrading, enabling and disabling SELinux policy modules.
semanage - This is to tweak policy module settings.
getsebool - to get the current boolean setting of the module

setsebool - use this command to set boolean settings or turn on module for SElinux enforcing . To enable and disable SELinux and how to change some of the policy settings using boolean values

`semanage login -l - This will list linux user and respective SELinux user`
`semanage user -l - This will list all SELinux user and respective roles`

ls -Z /etc/*.conf — This command will show security context of each file. Security context is like signature that SELinux verifies before giving access. example - `system_u:object_r:etc_t:s0`

File security Context — `system_u:object_r:etc_t:s0`

`id -Z -` This will display default user assignment to group, role etc like shown below

`unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023`

There are 4 parts separate by :
First part is SELinux equivalent to linux normal user
Second part of SElinux role , third part of type .. what type of file .. and forth part is multi-level security flag.

Process security context —
start httpd and vsftpd service and then check output of `ps -efZ | grep 'httpd\|vsftpd'`

```
system_u:system_r:httpd_t:s0           root       7126    1       0 16:50 ?        0
system_u:system_r:httpd_t:s0           apache     7127    7126    0 16:50 ?        0
system_u:system_r:httpd_t:s0           apache     7128    7126    0 16:50 ?        0
system_u:system_r:httpd_t:s0           apache     7129    7126    0 16:50 ?        0
system_u:system_r:httpd_t:s0           apache     7130    7126    0 16:50 ?        0
system_u:system_r:httpd_t:s0           apache     7131    7126    0 16:50 ?        0
system_u:system_r:ftpd_t:s0-s0:c0.c1023 root      7209    1       0 16:54 ?        0
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 root 7252 2636  0 16:57 pts/0 00
```

As shown above , process security context is `system_u:system_r:httpd_t:s0`
So this security context consists of user, role, domain, and sensitivity. Same as file security context.. Domain is unique to the process.

Using this model, even if a process is hijacked by another malicious process or user, the worst it can do is to damage the files it has access to. For example, the vsftp daemon will not have access to files used by say, sendmail or samba. This restriction is implemented from the kernel level: it's enforced as the SELinux policy loads into memory, and thus the access control becomes mandatory.

```
allow <domain> <type>:<class> { <permissions> };
```

Class defines what the resource actually represents (files, directories, symbolic links , ports , cursor etc)
sesearch command to check the type of access allowed for the httpd daemon.

```
sesearch --allow --source httpd_t --target
httpd_sys_content_t --class file
```

To change the context of the file , use below command -

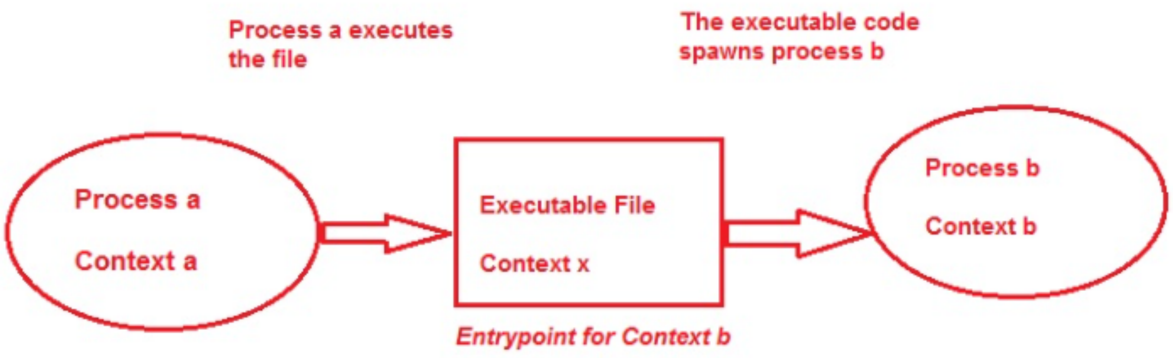
```
chcon --type var_t /var/www/html/index.html
```

see how we change type to var_t and now webpage index.html will not work.Forbidding access since it does not find file type.

to Restore use below command

```
restorecon -v /var/www/html/index.html
```

Domain Transition - Domain transition is the method where a process changes its context from one domain to another. To understand it, let's say you have a process called proca running within a context of contextat. With domain transition, proca can run an application (a program or an executable script) called appx that would spawn another process. This new process could be called procb and it could be running within the contextbt domain. So effectively, contextat is transitioning to contextbt through appx. The appx executable is working as an entrypoint to contextb_t.



Read few paragprah in document link above.. it is simple to under following 3 points -
Domain transition is subject to three strict rules:

- The parent process of the source domain must have the execute permission for the application sitting between both the domains (this is the entrypoint).
- The file context for the application must be identified as an entrypoint for the target domain.
- The original domain must be allowed to transition to the target domain.

SELinux users -https://www.digitalocean.com/community/tutorials/an-introduction-to-selinux-on-centos-7-part-3-users

SELinux users are different entities from normal Linux user accounts, including the root account. An SELinux user is not something you create with a special command, nor does it have its own login access to the server. Instead, SELinux users are defined in the policy that's loaded into memory at boot time, and there are only a few of these users. The user names end with _u, just like types or domain names end with _t and roles end with _r. Different SELinux users have different rights in the system and that's what makes them useful.

Any Linux user that maps to the unconfined_u user will have the privileges to run any app that runs within the unconfined_t domain

```
semanage login -a -s user_u regularuser
```

In this command, We are adding (-a) the regularuser account to the SELinux (-s) user account user_u group .

With that we can restrict users to do following (e.x.)

1. Restrict to switch users
2. Restrict users to run script from home folder
3. Restrict access to services

SELinux sits between the user and the process domain and controls what domains the user's process can get into. Roles are not that important when we see them in file security contexts. For files, it's listed with a generic value of object_r. Roles become important when dealing with users and processes.

To check what role is assigned to user group user_u (e.g.) run below command

```
seinfo -uuser_u -x
seinfo -ruser_r -x
```
– This will display details of role user_r

```
user_r
   Dominated Roles:
      user_r
   Types:
      git_session_t
      sandbox_x_client_t
      git_user_content_t
      virt_content_t
      policykit_grant_t
      httpd_user_htaccess_t
      telepathy_mission_control_home_t
      qmail_inject_t
      gnome_home_t
      ...
      ...
```

domain names ends with _t .. so if we remove any domain access to user_r role then any user assigned to user_u which in turn assigned to user_r will not get access to that service that belongs to that domain. — Simple !! :) SELinux user, SELinux role, and SELinux type or domain

Audit logs - `/var/log/audit/audit.log`
`/var/log/messages`

What is multi-level security ? -
In example shown below `-rw-------. root root system_u:object_r:etc_t:s0 /etc/vsftpd/vsftpd.conf`

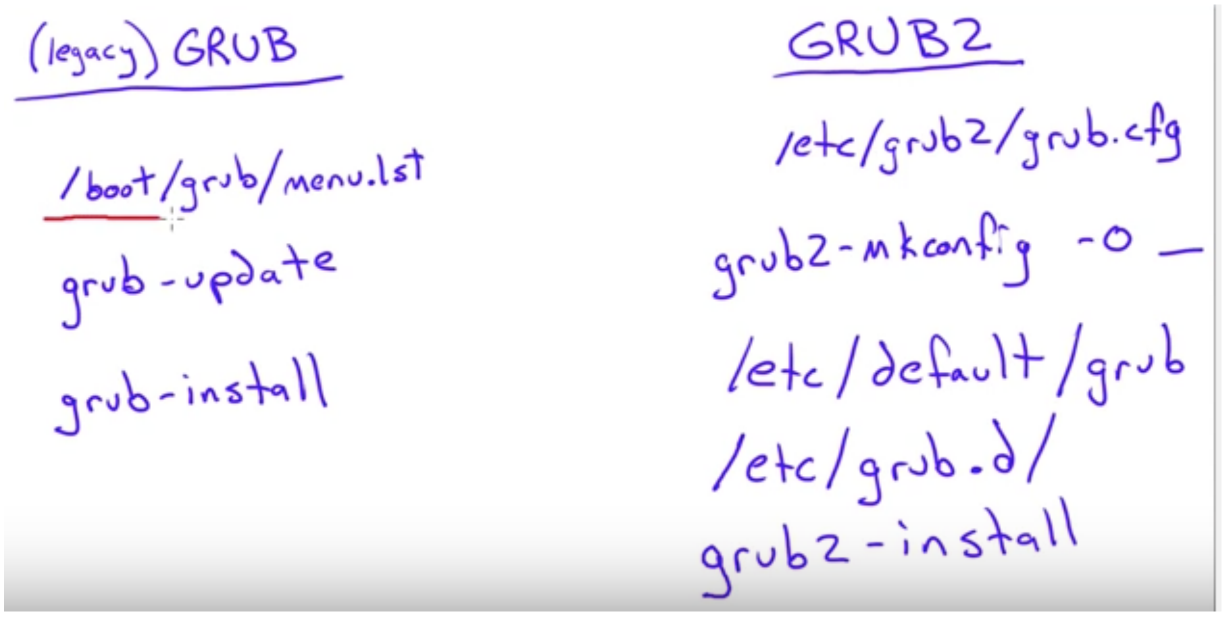
The sensitivity is part of the hierarchical multilevel security mechanism. By hierarchy, we mean the levels of sensitivity can go deeper and deeper for more secured content in the file system. Level 0 (depicted by s0) is the lowest sensitivity level, comparable to say, "public." There can be other sensitivity levels with higher s values: for example, internal, confidential, or regulatory can be depicted by s1, s2, and s3 respectively.
For SELinux security contexts, sensitivity and category work together when a category is implemented. When using a range of sensitivity levels, the format is to show

sensitivity levels separated by a hyphen (for example, s0-s2). When using a category, a range is shown with a dot in between. Sensitivity and category values are separated by a colon (:).

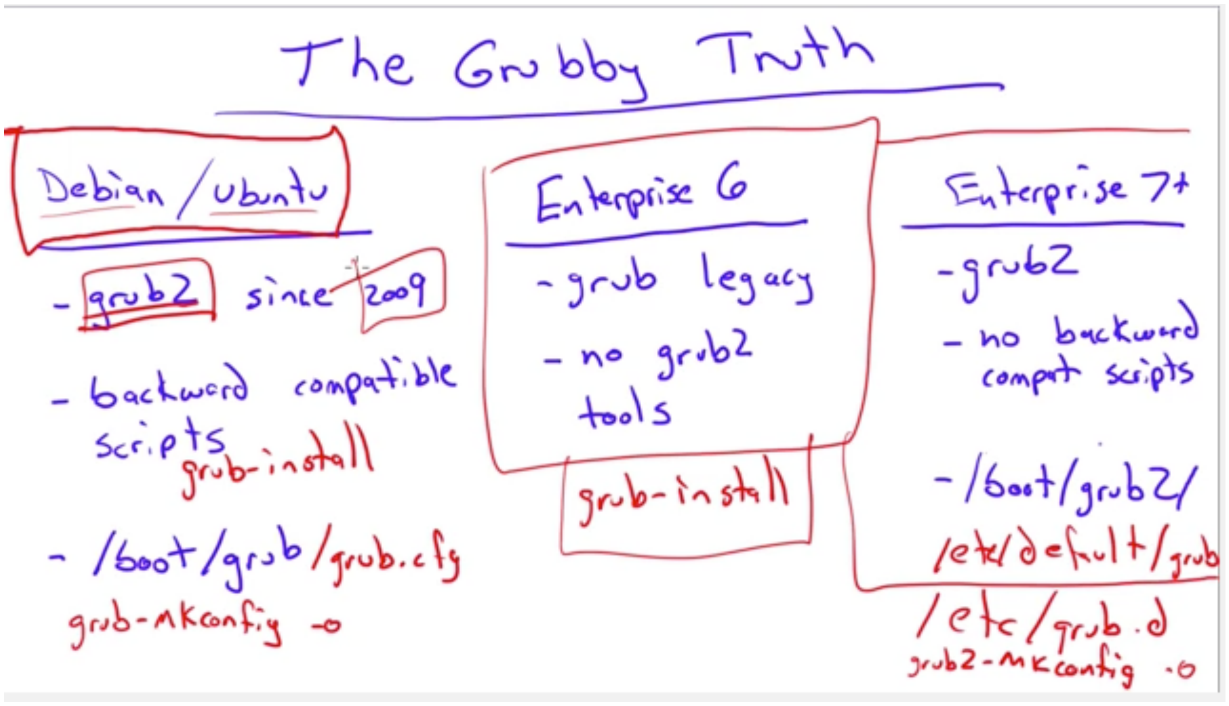
```
user_u:object_r:etc_t:s0:c0.c2
```

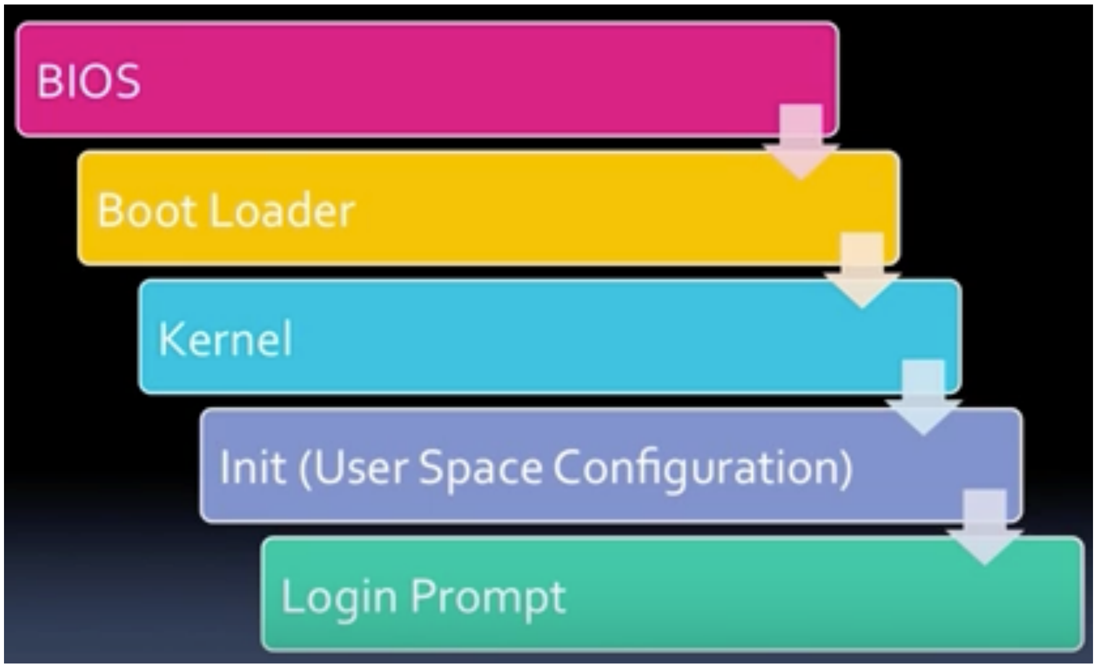
———— Linux - LILO BOOT LOADER

GRUB(now called legacy) and GRUB2 — **G**rand **U**nified **B**oot **S**ystem . As shown below menu.lst file exists only in older legacy distributions whereas new ones has grub.con files. This GRUB.CFG is not allowed to modify..



In diagram below, Enterprise 6 has centOs which was using legacy GRUB. Enterprise 7 (RHEL and CentOS) uses GRUB2. grub.cfg is not allowed to modify..We can only modify /etc/default/grub and /etc/grub.d files as shown below diagram enterprise 7 block. Once these two files are modified run grub-mkconfig -o /boot/grub/grub.cfg -o means output. this command will modify grub.cfg file. thats why we are not supposed to update grub.cfg directly.





When using Red Hat Enterprise Linux, two boot loaders are available: *GRUB* or *LILO*. GRUB is the default boot loader, but LILO is available for those who require or prefer it.
Boot Process —> Load kernel into memory —> hands off boot process to init command.

The `/sbin/init` program (also called `init`) coordinates the rest of the boot process and configures the environment for the user.

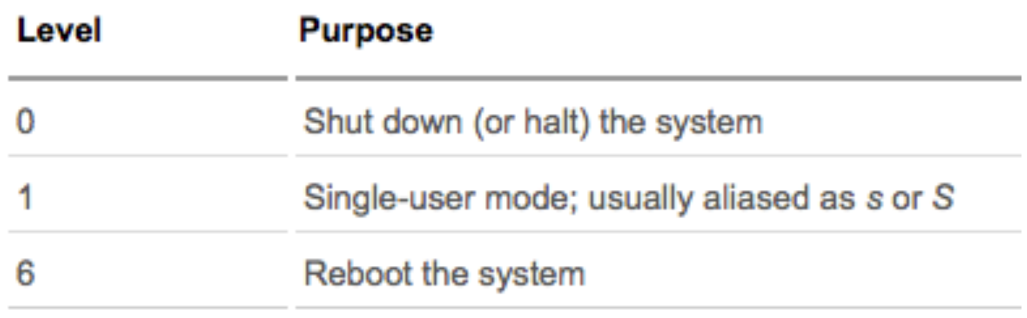
Command **dmesg** gives boot log
Best mote on above - https://access.redhat.com/documentation/en-US/

Red_Hat_Enterprise_Linux/3/html/Reference_Guide/s1-boot-init-shutdown-process.html

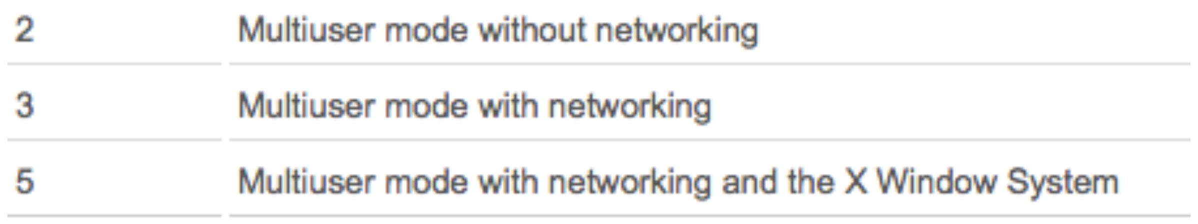
DirectBill.ESC@thehartford.com
Corinne.Mongillo@thehartford.com
Agency.Service@thehartford.com
————**Managing Runlevels**

https://www.youtube.com/watch?v=jbYucYX1WwM

https://www.ibm.com/developerworks/library/l-lpic1-v3-101-3/



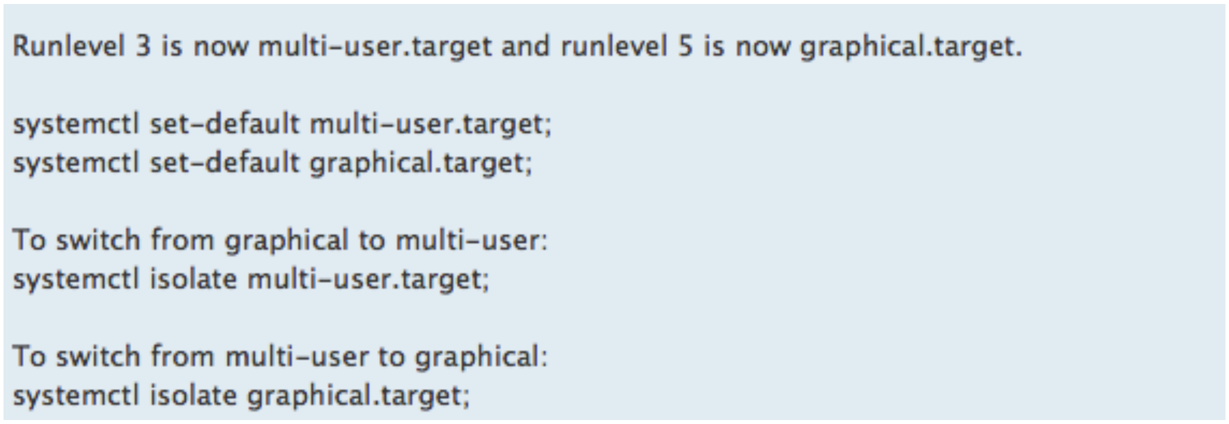
Level	Purpose
0	Shut down (or halt) the system
1	Single-user mode; usually aliased as *s* or *S*
6	Reboot the system



2	Multiuser mode without networking
3	Multiuser mode with networking
5	Multiuser mode with networking and the X Window System

There are various run levels rc0.d , rc1.d ……rc6.d under /etc/ . Dependng on run level that system is booting, it will execute script located in below location rc.local
Auto Run Script as Root at Startup - edit /etc/init.d/rc.local
rc5.d This will include graphical interface GUI/ network and multi-user environment

To enable units on any specific run level , use command chkconfig or systemctl (depending on OS). In case of CENTOS 7 , use below



Runlevel 3 is now multi-user.target and runlevel 5 is now graphical.target.

```
systemctl set-default multi-user.target;
systemctl set-default graphical.target;
```

To switch from graphical to multi-user:

```
systemctl isolate multi-user.target;
```

To switch from multi-user to graphical:

```
systemctl isolate graphical.target;
```

What is tty - "tty" originally meant "teletype" and "pty" means "pseudo-teletype". In UNIX, /dev/tty* is any device that acts like a "teletype", ie, terminal. (Called teletype because that's what we had for terminals in those benighted days.)
A pty is a pseudotty, a device entry that acts like a terminal to the process reading and writing there, but managed by something else. They first appeared (as I recall) for X Windows and screen and the like, where you needed something that acted ilke a terminal but could be used from another program.

——————————What is exec command
The exec command replaces the current shell process with the specified command. Normally, when you run a command a new process is spawned (forked). The exec command does not spawn a new process. Instead, the current process is overlaid with the new command. In other words the exec command is executed in place of the current shell without creating a new process.

example following command will quit shell after we exit vi editor because vi proceess is now taken over by exec

```
exec vi testfile.txt
```

Similarly , if we want to change shell for user , then use exec bash .. this will change shell for user and he or she does not need to manually change by going into /etc/passwd file

Another example -

```
while true
do
  echo "Go to Mail, Editor, or eXit:"
  read ANSWER
  case "$ANSWER" in
    [mM]) exec MAILX ;;
    [eE]) exec ${EDITOR:=vi} ;;
    [xX]) exec exit ;;
  esac
done
```

— — — —

STAT command in linux - It gives information about file or directory.

echo $$ will list current process id
You can see specific process id into /var/run/ … it ends with .pid

————————— Systemd or Systemctl or init process
systemd is an init system that provides many powerful features for starting, stopping and managing processes. Within the CoreOS world, you will almost exclusively use systemd to manage the lifecycle of your Docker containers.

- **System V** is the older init system:
  - Debian 6 and earlier
  - Ubuntu 9.04 and earlier
  - CentOS 5 and earlier
- **Upstart**:
  - Ubuntu 9.10 to Ubuntu 14.10, including Ubuntu 14.04
  - CentOS 6
- **systemd** is the init system for the most recent distributions featured here:
  - Debian 7 and Debian 8
  - Ubuntu 15.04 and newer
  - CentOS 7

Your running Linux or Unix system will have a number of background processes executing at any time. These processes - also known as services or daemons - may be native to the operating system, or run as part of an application.
Good exercise on systems - https://coreos.com/docs/launching-containers/launching/getting-started-with-systemd/
Examples of operating system services:
- sshd daemon that allows remote connections
- cupsd daemon that controls printing

Examples of application daemons:

- httpd/apache2 is a web server service
- mongod is a database daemons

There are important service that needs to run continuously to make sure websites, mail, database and other apps and also start these service automatically after reboot or crash. These self-healing linux is called init systems.

Systemd is a system and service manager for Linux operating systems. It is designed to be backwards compatible with SysV init scripts, and provides a number of features such as parallel startup of system services at boot time, on-demand activation of daemons, support for system state snapshots, or dependency-based service control logic.

Previous versions of Red Hat Enterprise Linux, which were distributed with SysV init or Upstart, used init scripts written in bash located in the /etc/rc.d/init.d/ directory. In RHEL 7 / CentOS 7, these init scripts have been replaced with service units. Service units end with the .service file extension and serve a similar purpose as init scripts. To view, start, stop, restart, enable, or disable system services you will use the systemctl instead of the old service command.

here is good tutorial on how to set up custom upstart event system - https://www.digitalocean.com/community/tutorials/the-upstart-event-system-what-it-is-and-how-to-use-it

https://www.certdepot.net/rhel7-get-started-systemd/

What is UPSTART ?
Designed with flexibility from the beginning, the Upstart event system utilizes a variety of concepts that differ from conventional initialization systems. The solution is installed by default on Red Hat Enterprise Linux (RHEL) 6, as well as Google's Chrome OS, and Ubuntu
1. Jobs - There are 3 types of jobs (task jobs - run with purpose , service jobs - which run in the background and abstract jobs - process that runs forever , until stopped by the user)
2. Events - Events, however, are the signals or "calls" used to trigger a certain action with a job or another event. The common forms of events refer to the monitoring of a process: starting, started, stopping and stopped.
3. Emitting events - It is like broadcasting events

Commands on Ubuntu

`initctl emit <event>` – This is to emit or broadcast events

Procedure for task job —
Create a file into /etc/init with .conf extension. Now in this file you can write any sh script to perform some actions on various run level example - We are writing some echo text to log file .

```
start on run level [2345]
exec echo Test Job ran at  `date` >> /var/log/
testjob.log
```

`init-checkconf /etc/init/testjob.conf -` this is to check syntax of conf file

`sudo service <name of conf file> start —` Once conf file is ready, these commands can be run on it
`sudo initctl status <name of conf file> -` To check status of conf service

So basically if we create conf file into init folder under ETC then it becomes service so we can use service related keywords like start, stop, restart etc. System jobs reside in the /etc/init/ directory, and user jobs reside in the user's own init directory, ~/.init/.Regardless of its type, a job is always defined in a configuration file (.conf) where its filename should represent the service or task involved.

Procedure for service job (jobs that runs in background) ——
here we defined same conf file but there are pre-start, post start, pre-stop , post stop scripts that we can implement. example below is to do some action to get node.js server started . In example below, filesystem means - normal running of the system. For node.js to run, we need to set env. path /src and also use bin folder to start nodetst.js script. Echo $$ for process id . In post-stop script, we are removing process before logging info.

```
description "Test node.js server"
author      "Your Name"

start on filesystem or runlevel [2345]
stop on shutdown

script

    export HOME="/srv"
    echo $$ > /var/run/nodetest.pid
    exec /usr/bin/nodejs /srv/nodetest.js

end script

pre-start script
    echo "[`date`] Node Test Starting" >> /var/log/nodetest.log
end script

pre-stop script
    rm /var/run/nodetest.pid
    echo "[`date`] Node Test Stopping" >> /var/log/nodetest.log
end script
```

Nodetest.js script may look like below. Above script can be built by taking reference from other script or service that are located under /ETC/INIT/….<e.g. mysql.conf>

```
var http = require("http");

http.createServer(function(request, response) {
    response.writeHead(200, {"Content-Type": "text/plain"});
    response.write("Hello World");
    response.end();
}).listen(8888);
```

Some of the scripting stanzas are -

- waiting: the initial state of processing
- starting: where a job is about to start
- pre-start: where the pre-start section is loaded
- spawned: where a script section is about to run
- post-start: where post-start operations take place
- running: where the job is fully operational
- pre-stop: where pre-stop operations take place
- stopping: where the job is being stopped
- killed: where the job is stopped
- post-stop: where post-stop operations take place - to clean up

Below is few lines of sample script for mysql ..

```
description     "MySQL Server"
author          "Mario Limonciello <superm1@ubuntu.com>"

start on runlevel [2345]
stop on starting rc RUNLEVEL=[016]

respawn
respawn limit 2 5
```

here we are using respawn directives .. The respawn directives restart the service after a crash. the respawn limit directive stipulates how many times Linux will try to restart the crashed service in an interval specified in seconds. In this case, the first argument (2) is the number of tries, and the second one (5) is the interval. If the service does not start up (respawn) successfully within this threshold, it will be kept in a stopped state.

——————What is File descriptor ?
#0 - STDIN - Standard input
#1 - STDOUT - Standard output

#2 - STDERR - Standard Error

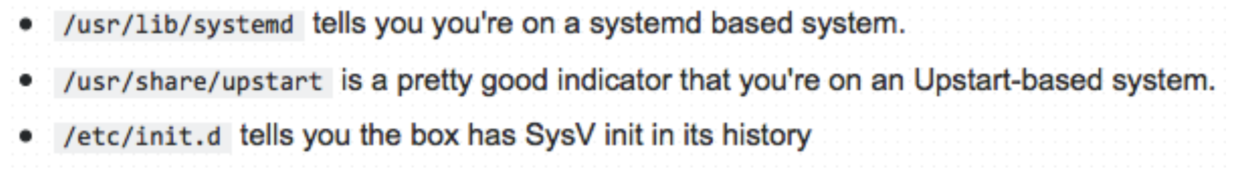
To move STDERR and STDOUT to some file , use &>
ls Filenotexist fileexists &> dumphere.txt

& - This sign tells linux that it is file descriptor

————————More exercise on SYSTEMD
First implemented in Fedora, systemd now comes with RHEL 7 and its derivatives like CentOS 7. Ubuntu 15.04 ships with native systemd as well. Other distributions have either incorporated systemd, or announced they will soon. systemd is backwards-compatible with System V commands and initialization scripts.

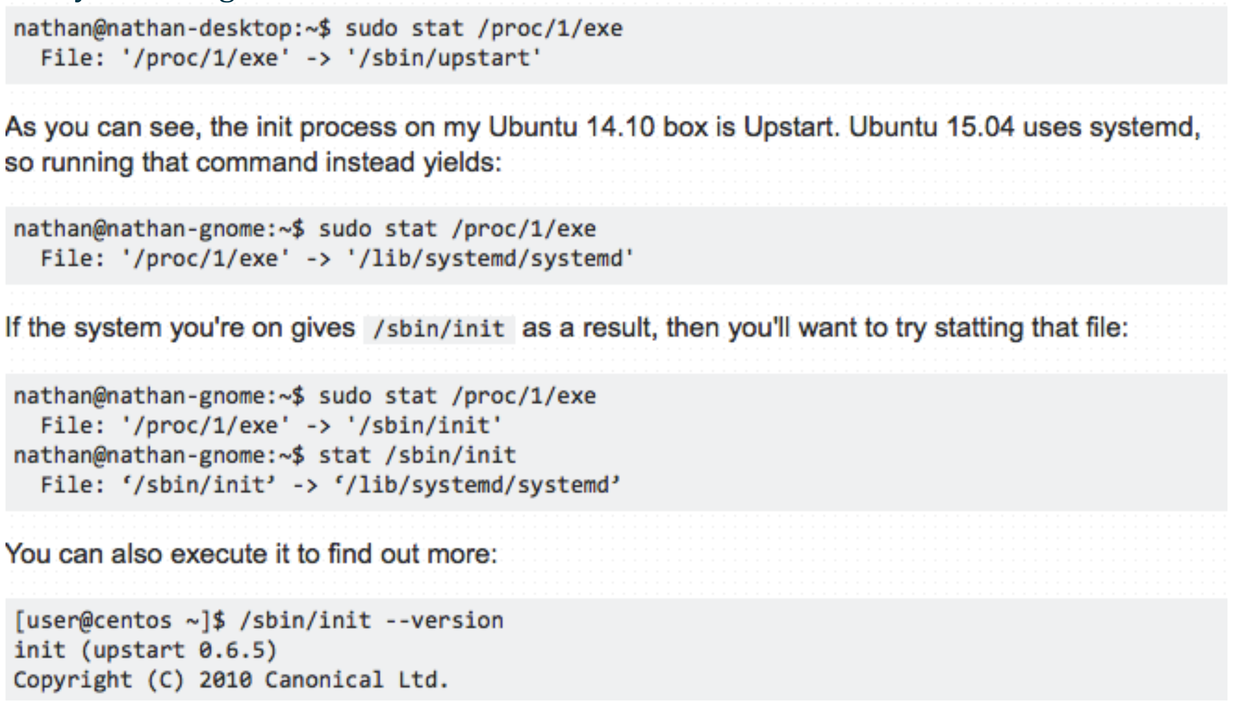
- Debian 7 and Debian 8
- Ubuntu 15.04
- CentOS 7

How do we know if system is using SysV or upstart or systemd… check if you find any of these directories



- `/usr/lib/systemd` tells you you're on a systemd based system.
- `/usr/share/upstart` is a pretty good indicator that you're on an Upstart-based system.
- `/etc/init.d` tells you the box has SysV init in its history

Or try following commands -



```
nathan@nathan-desktop:~$ sudo stat /proc/1/exe
  File: '/proc/1/exe' -> '/sbin/upstart'
```

As you can see, the init process on my Ubuntu 14.10 box is Upstart. Ubuntu 15.04 uses systemd, so running that command instead yields:

```
nathan@nathan-gnome:~$ sudo stat /proc/1/exe
  File: '/proc/1/exe' -> '/lib/systemd/systemd'
```

If the system you're on gives `/sbin/init` as a result, then you'll want to try statting that file:

```
nathan@nathan-gnome:~$ sudo stat /proc/1/exe
  File: '/proc/1/exe' -> '/sbin/init'
nathan@nathan-gnome:~$ stat /sbin/init
  File: '/sbin/init' -> '/lib/systemd/systemd'
```

You can also execute it to find out more:

```
[user@centos ~]$ /sbin/init --version
init (upstart 0.6.5)
Copyright (C) 2010 Canonical Ltd.
```

`/sbin/init` is then a symbolic link to whichever was configured to run

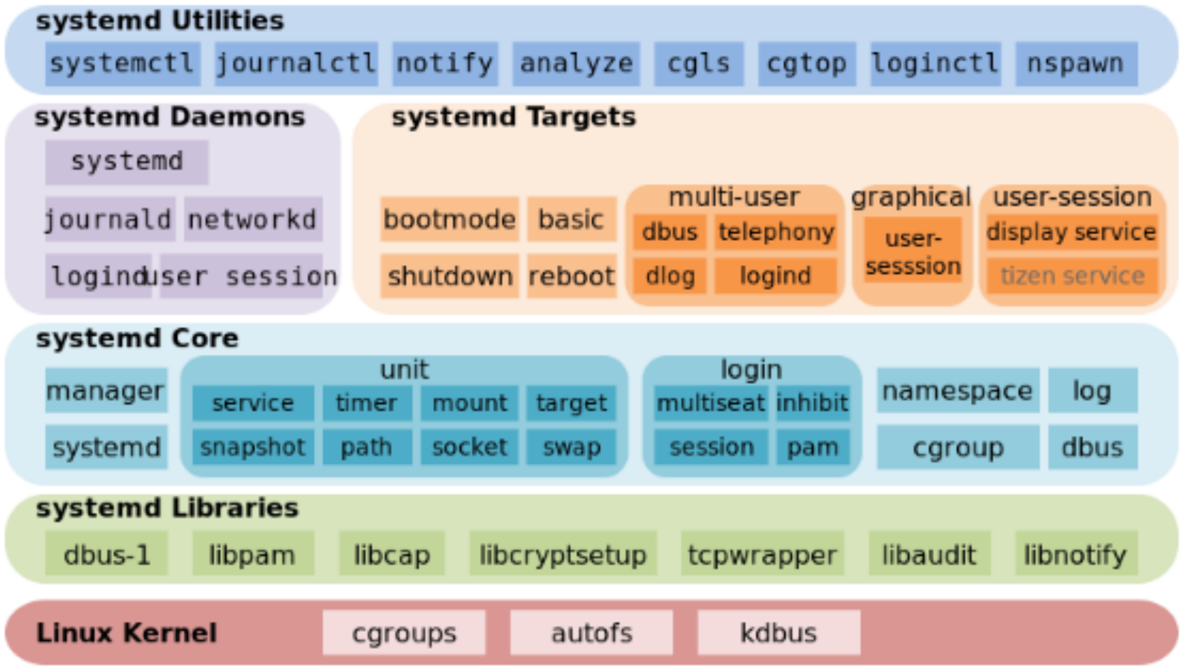
For RPM based systems like CentOS or Fedora or openSUSE , we can also use below command

```
fedora:~$ rpm -qf /sbin/init
systemd-216-24.fc21.x86_64

centos:~$ rpm -qf /sbin/init
upstart-0.6.5-12.el6_4.1.x86_64

opensuse:~$ rpm -qf /sbin/init
systemd-sysvinit-44-10.1.1.i586
```

For Debian based system like Ubuntu

```
ubuntu:~$ dpkg -S /sbin/init
upstart: /sbin/init
```

https://www.linux.com/learn/understanding-and-using-systemd



With systemd, most standard applications you can install, such as Nginx or MySQL, will start after reboot and also start after a crash by default, so you don't have to do anything to make this work. They will come with their own init scripts in /etc/systemd/system already. Systemd is backward compatible that means it can run some of the systemV commands also .(System V was older init module) . The init process is always assigned PID 1. The /proc filesystem provides a way to obtain the path to an executable given a PID.

here is sample unit file

```
[Unit]
Description=MyApp
After=docker.service
Requires=docker.service

[Service]
TimeoutStartSec=0
ExecStartPre=-/usr/bin/docker kill busybox1
ExecStartPre=-/usr/bin/docker rm busybox1
ExecStartPre=/usr/bin/docker pull busybox
ExecStart=/usr/bin/docker run --name busybox1 busybox /bin/sh -c "while true; do echo
Hello World; sleep 1; done"

[Install]
WantedBy=multi-user.target
```

—> COULD NOT INSTALL SYSTEMD ..Didnot find any document on net..

here is some theory to read —

https://coreos.com/docs/launching-containers/launching/getting-started-with-systemd/

https://www.youtube.com/watch?v=RvHJ1jxPzuE

https://www.digitalocean.com/community/tutorials/how-to-configure-a-linux-service-to-start-automatically-after-a-crash-or-reboot-part-1-practical-examples

https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/chap-Managing_Services_with_systemd.html

— — — — — — — — —disk command / Partitions SWAP memory
Read this first - http://litux.nl/Reference/books/7213/ddu0222.html
http://www.tecmint.com/fdisk-commands-to-manage-linux-disk-partitions/

http://www.howtogeek.com/184659/beginner-geek-hard-disk-partitions-explained/

Hard disks, USB drives, SD cards — anything with storage space must be partitioned. An unpartitioned drive can't be used until it contains at least one partition, but a drive can contain multiple partitions.Partitioning isn't something most users will need to bother with, but you may need to work with partitions when installing an operating system or setting up a new drive.

What is partition - Many drives come with a single partition already set up, but all storage devices are just treated as a mass of unallocated, free space when they contain no partitions. To actually set up a file system and save any files to the drive, the drive needs a partition.The partition can contain all of the storage space on the drive or just some of it. On many storage devices, a single partition will often take up the entire drive.Partitions are necessary because you can't just start writing files to a blank drive. You must first create at least one container with a file system. We call this container a partition. You can have one partition that contains all the storage space on the drive or divide the space into twenty different partitions. Either way, you need at least one partition on the drive.

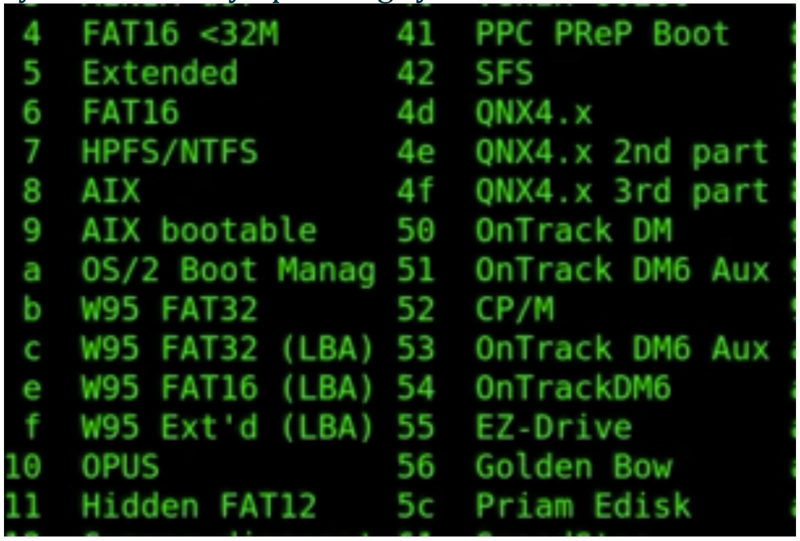
After creating a partition, the partition is formatted with a file system — like the NTFS file system on Windows drives, FAT32 file system for removable drives, HFS+ file system on Mac computers, or the ext4 file system on Linux. Files are then written to that file system on the partition.

Best material -
http://www.howtogeek.com/184659/beginner-geek-hard-disk-partitions-explained/

When partitioning, you'll need to be aware of the difference between primary, extended, and logical partitions. A disk with a traditional partition table can only have up to 4 partitions. Extended and logical partitions are a way to get around this limitation.Each disk can have up to four primary partitions or three primary partitions and an extended partition.

Some of the commands are
fdisk -l

fdisk <name of device>
command for new partition = n
to write partition into memory = w
to see different types of file systems = t and then L to list all hex codes — types of file system varies by operating systems.



```
 4  FAT16 <32M      41  PPC PReP Boot
 5  Extended        42  SFS
 6  FAT16           4d  QNX4.x
 7  HPFS/NTFS       4e  QNX4.x 2nd part
 8  AIX             4f  QNX4.x 3rd part
 9  AIX bootable    50  OnTrack DM
 a  OS/2 Boot Manag 51  OnTrack DM6 Aux
 b  W95 FAT32       52  CP/M
 c  W95 FAT32 (LBA) 53  OnTrack DM6 Aux
 e  W95 FAT16 (LBA) 54  OnTrackDM6
 f  W95 Ext'd (LBA) 55  EZ-Drive
10  OPUS            56  Golden Bow
11  Hidden FAT12    5c  Priam Edisk
```

to commit - mkfs.<type of parttion .e.g. ext4, VFAT etc>

Good video - https://www.youtube.com/watch?v=5kVAzxTwy5Q

For example, floppy drives are referred to as /dev/fd0 and /dev/fd1. IDE/EIDE hard drives are referred to as /dev/hda, /dev/hdb, /dev/hdc, and storage as sad, sa2 etc

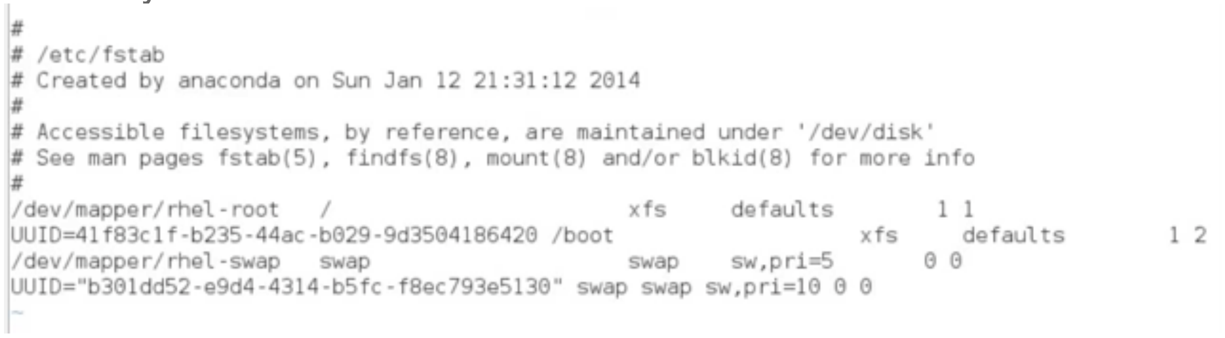
——— What is Gparted tool on linux to manage partition

http://www.howtogeek.com/114503/how-to-resize-your-ubuntu-partitions/

——————————**Swap Memory (virtual memory)**
Ram memory is physical but swap is partition logical files.
command ->
swapon -s (s for summary of all partitions)
cat /proc/swaps — It is stored under
fdisk -l to display list of partition tables

To mount swap file use /etc/fstab . In example below, pri stands for

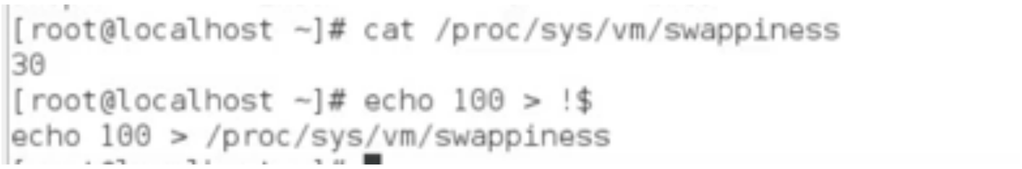
priority
last entry is for file mount



```
#
# /etc/fstab
# Created by anaconda on Sun Jan 12 21:31:12 2014
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/rhel-root   /                       xfs     defaults        1 1
UUID=41f83c1f-b235-44ac-b029-9d3504186420 /boot                   xfs     defaults        1 2
/dev/mapper/rhel-swap   swap                    swap    sw,pri=5        0 0
UUID="b301dd52-e9d4-4314-b5fc-f8ec793e5130" swap swap sw,pri=10 0 0
~
```

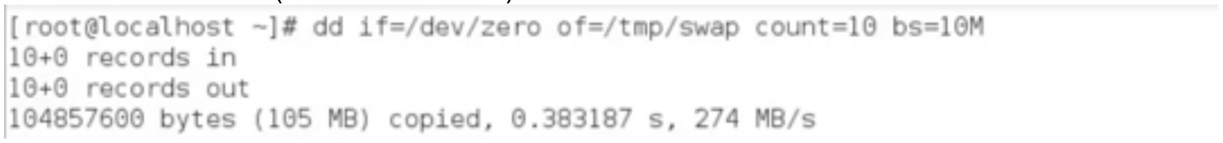
To check virtual memory statistics use this command -> vmstat

To check free space on disk use —> free -m command . If you don't wish to use swap then use below command to set value to low .. higher value means you wish to use more swap memory for processing. Here is !$ in echo command to get last argument that you typed.. we typed earlier cat so it is cat in this case.

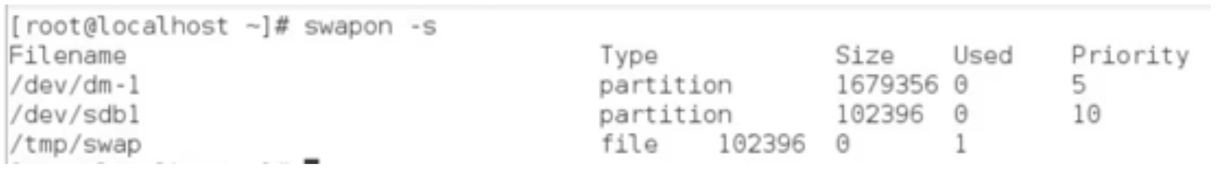


```
[root@localhost ~]# cat /proc/sys/vm/swappiness
30
[root@localhost ~]# echo 100 > !$
echo 100 > /proc/sys/vm/swappiness
```

To create new file into swap , use below dd command . if (input file) and of (output file) , count= 10 block bs (block size in mb)



```
[root@localhost ~]# dd if=/dev/zero of=/tmp/swap count=10 bs=10M
10+0 records in
10+0 records out
104857600 bytes (105 MB) copied, 0.383187 s, 274 MB/s
```



```
[root@localhost ~]# swapon -s
Filename                                Type            Size    Used    Priority
/dev/dm-1                               partition       1679356 0       5
/dev/sdb1                               partition       102396  0       10
/tmp/swap                               file    102396  0       1
```

— — — — — —Linux Desktop Environment-Gnome,KDE,Xfce,MATE,Cinnamon
— — — — — —

Backward compatible (or sometimes backward-compatible or backwards compatible) refers to a hardware or software system that can successfully use interfaces and data from earlier versions of the system or with other systems. For example, Perl, the scripting language, was designed to be backward compatible with awk, an earlier language that Perl was designed to replace.

————Default ports for following service and applications

SMTP	25
DNS	53
FTP	20 (data transfer) , 21 (Connection established)
DHCP	67/UDP(dhcp server) , 68/UDP(dhcp client)
SSH	22
tomcat	8080

nodejs - 8888

————————

echo $$ - this will give process id

————————Java
what is super class method and super class constructor

where we use THUS and GETTER and SETTER ?
https://www.youtube.com/watch?v=OF3vBYWikYs

——————————Linux all about process PS Command
https://www.digitalocean.com/community/tutorials/how-to-use-ps-kill-and-nice-to-manage-processes-in-linux

———————Collection

ArrayList looks like below
[0] [1] [2] [3] [4] ……

LinkedList looks like below

```
[0] -> [1] -> [2] …..
    <-     <-
```

Both ArrayList and LinkedList uses LIST interface

It is faster to add element at the end in case of array list whereas it is faster to add element at the beginning or in middle in case of linkedinlist

In case of HashMap , it stores key , value pair. There are there hash map classes -

HashMap, TreeMap, LinkedHashMap
TreeMap sorts the list , HashMap does sometimes and LinkedHashMap never. All use MAP interface

HashSet ; TreeSet and LinkedHashSet is nothing but set of values as shown below. repeat of value not allowed. It uses SET interface
{ dog , cat, mouse, snake} .

— — — — — — — — — — — —
RESTFUL API

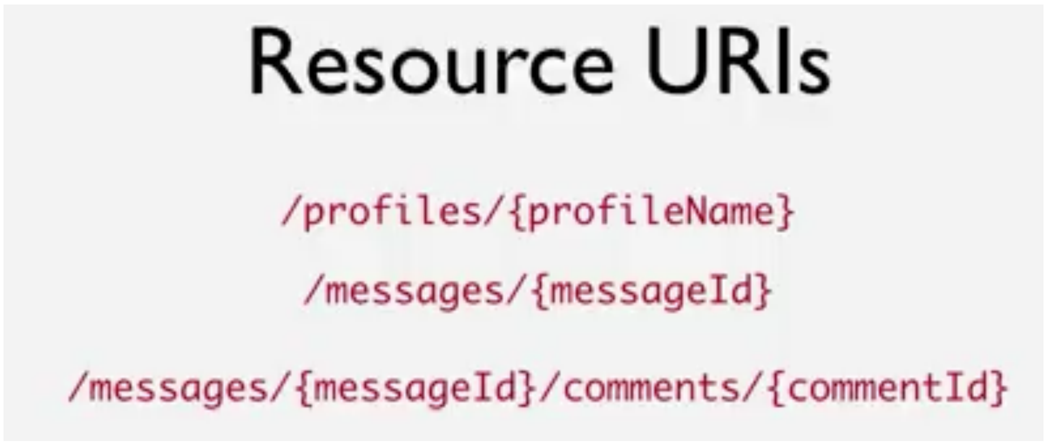
There are only guidelines to use data , no rules , unlike in SOAP .
Data can be in XML, JSON or text or anything for that matter

1. There is something called action based URL vs resource based URL

Example - weather.com/weather_cty?zipcode=123243 This is the action based URL since it has ? for action
whereas weather.com/city=sanjose is resource based URL

2. Status code of the web services tells about if service is active, or dead
3. HTTP Methods - GET / POST / PUT / DELETE
4. Content type tells if data is in XML or JSON or other format
5. Content negotiation - This happens between client and server

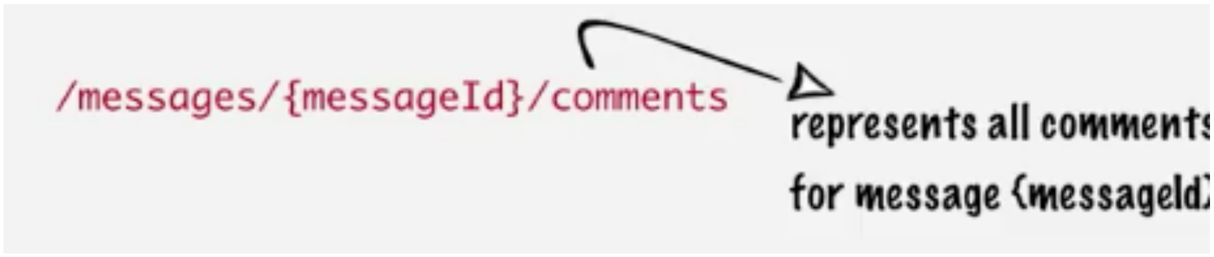
Two types of URI 1. Instance Resource URI



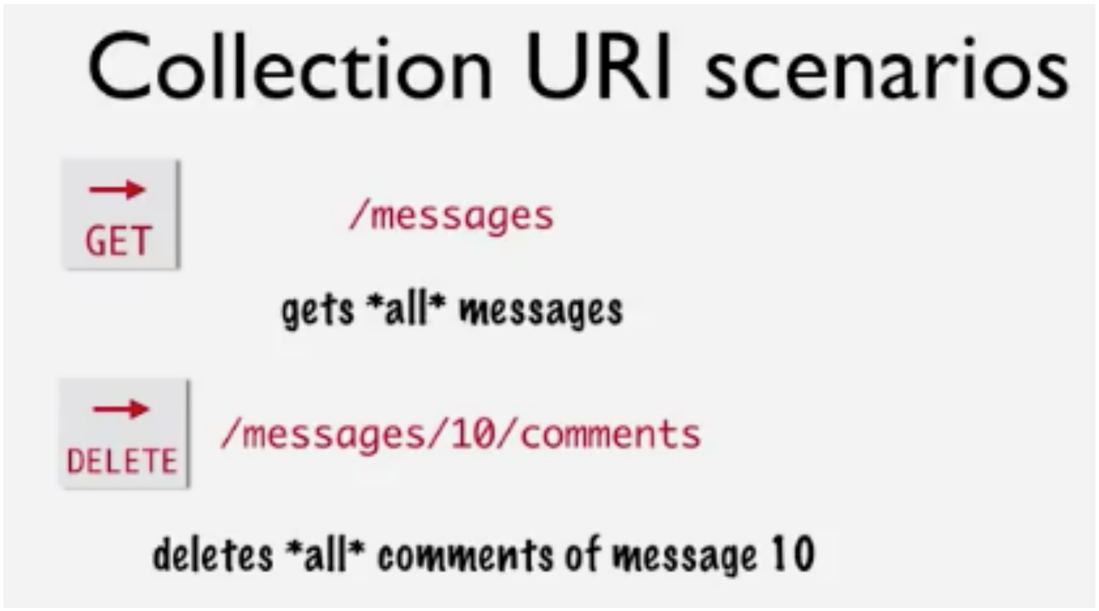
Resource URIs

```
/profiles/{profileName}
/messages/{messageId}
/messages/{messageId}/comments/{commentId}
```

2. Collection Resource URI


/messages/{messageId}/comments

represents all comment
for message {messageId


Collection URI scenarios

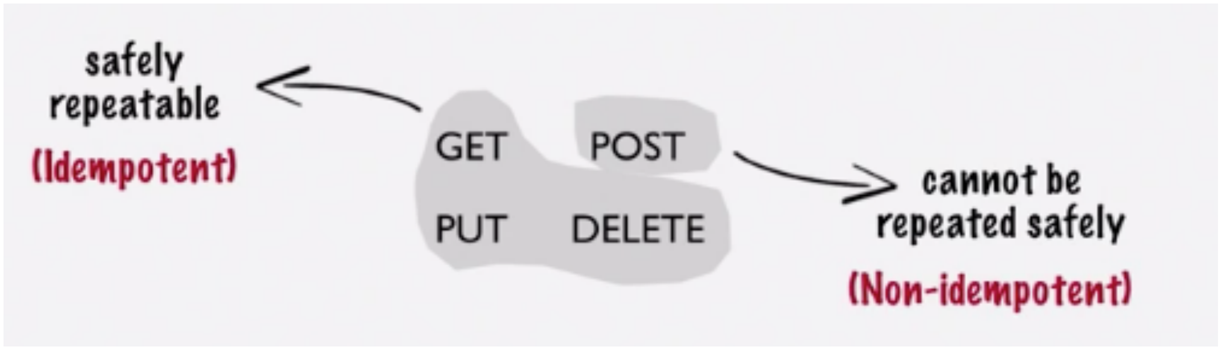
GET /messages

gets *all* messages

DELETE /messages/10/comments

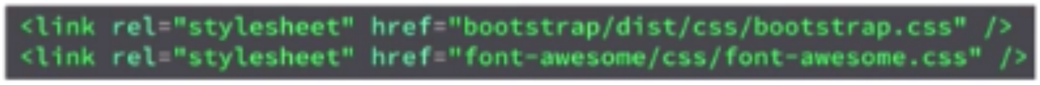
deletes *all* comments of message 10

PUT - Update existing resource body
POST - Create new resource

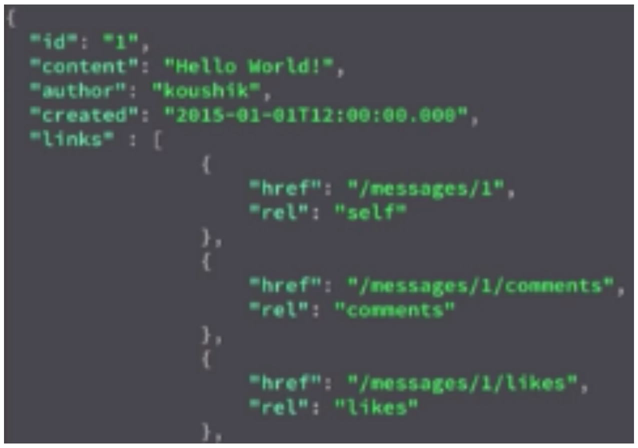


HATEOAS (internally tied hyperlinks ….)

REL Attribute

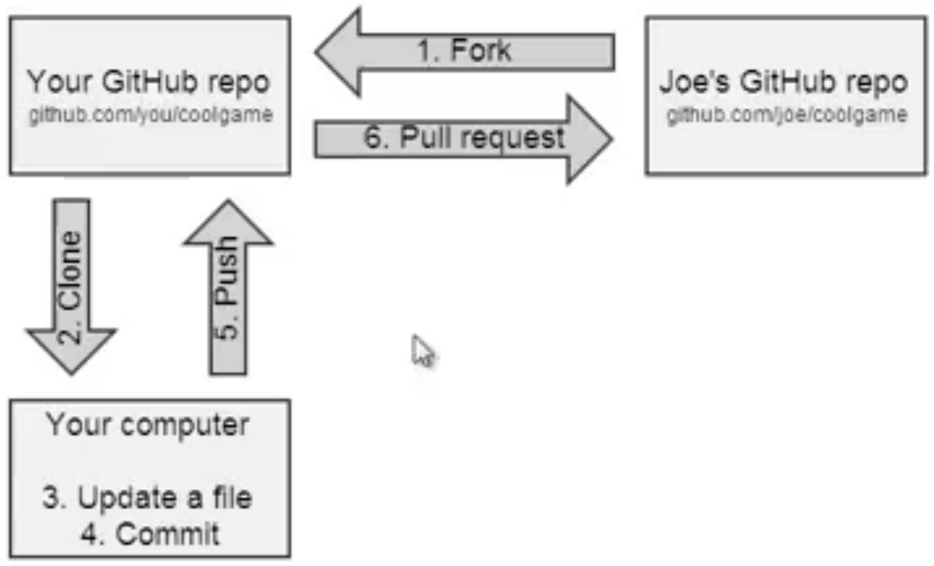


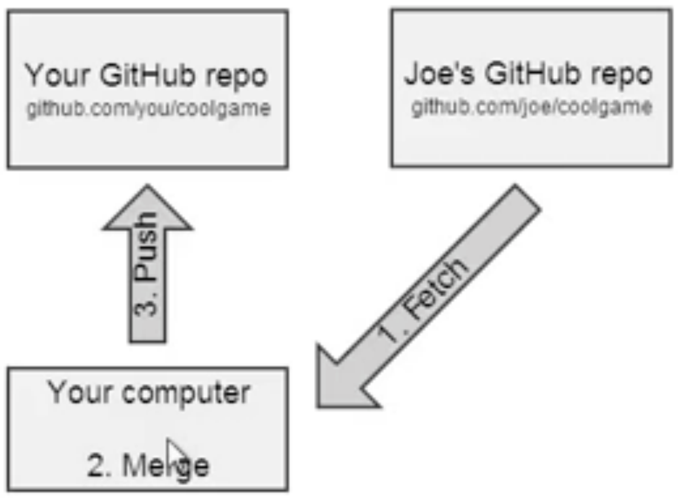
```
<link rel="stylesheet" href="bootstrap/dist/css/bootstrap.css" />
<link rel="stylesheet" href="font-awesome/css/font-awesome.css" />
```

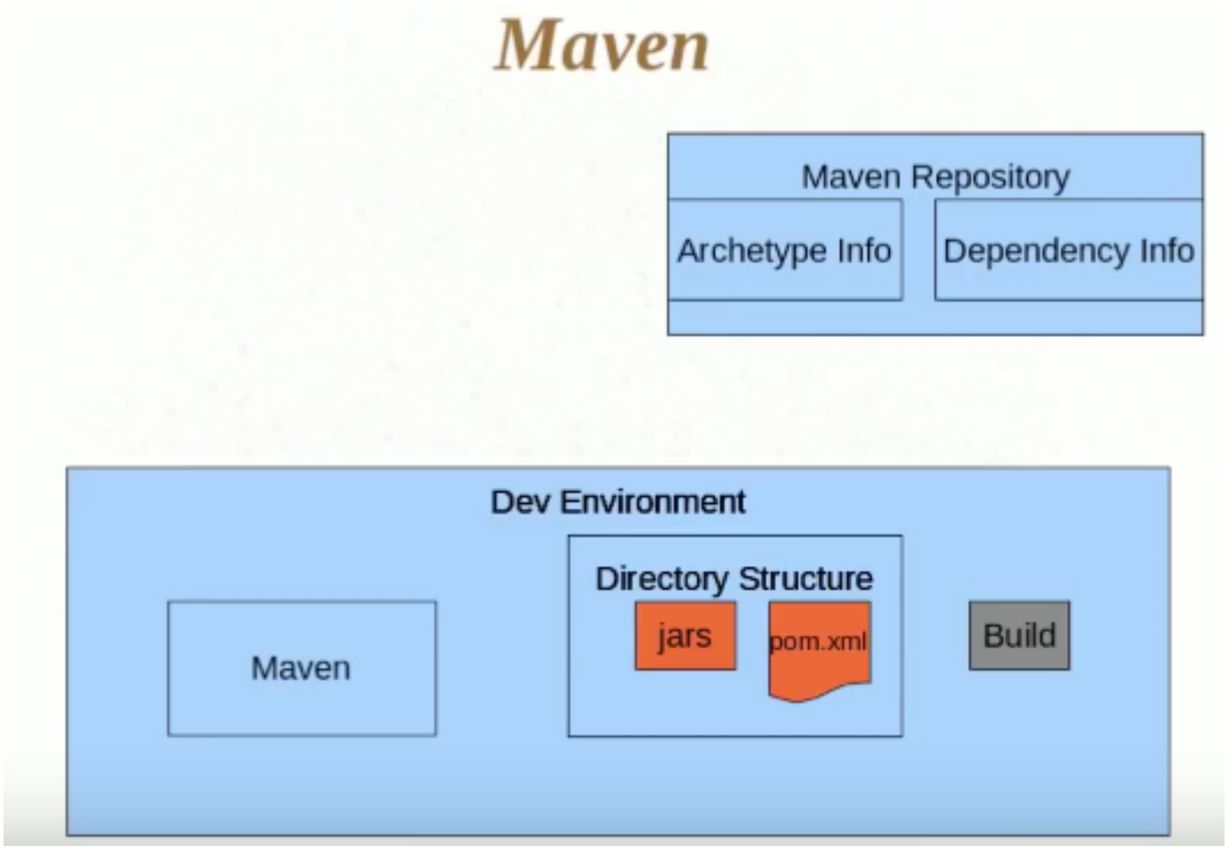


```
{
  "id": "1",
  "content": "Hello World!",
  "author": "koushik",
  "created": "2015-01-01T12:00:00.000",
  "links" : [
                {
                    "href": "/messages/1",
                    "rel": "self"
                },
                {
                    "href": "/messages/1/comments",
                    "rel": "comments"
                },
                {
                    "href": "/messages/1/likes",
                    "rel": "likes"
                },
```

———————————


Your GitHub repo
github.com/you/coolgame
1. Fork
6. Pull request
Joe's GitHub repo
github.com/joe/coolgame
2. Clone
5. Push
Your computer
3. Update a file
4. Commit


Your GitHub repo
github.com/you/coolgame
Joe's GitHub repo
github.com/joe/coolgame
3. Push
1. Fetch
Your computer
2. Merge



Firewall commands -
sudo ufw allow 22/tcp
sudo ufw show added
————
Self signed certificate
https://help.ubuntu.com/12.04/serverguide/certificates-and-security.html

——— FQDN - Fully qualified domain name - www.HarimIT.com.
There are 4 DNS servers as Root DNS server ; Top level domain DNS server (TLD) ; Domain name DNS server and Web server DNS server.
Your local server will send request to . or root server first and then comes down in the hierarchy as shown

How To Configure BIND as a Private Network DNS Server on Ubuntu 14.04 — https://www.digitalocean.com/community/tutorials/how-to-configure-bind-as-a-private-network-dns-server-on-ubuntu-14-04

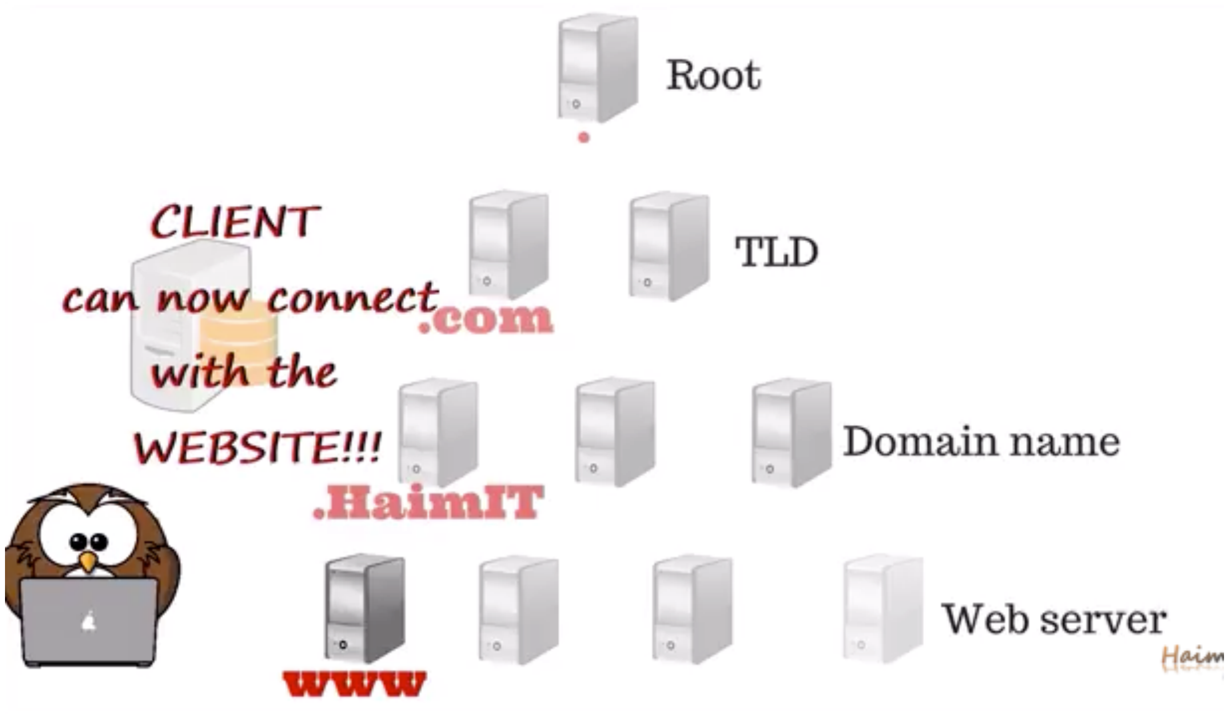
Steps -

Install BIND9 - > This is used to set up internet DNS server , BIND is name server software which will be used by ubuntu (virtual private server) to resolve host names and IP addresses .
Configure DNS server

1. Update trusted list of clients in /ETC/BIND/named.conf.options —> This is to identify which query client to trust and which is not
2. Update local file to add Zones —> Zone is place where we specify which file will hold mappings of IP address to host names etc
3. Create those zone files —> Here is where we have NS, A , AAAA record which is record data of the zone file
4. Create reverse zone files too

Finally configure DNS clients - Before client or trusted clients uses DNS server, we need to tell these clients to use our name servers so that they can get the IP address to host mapping.. This is done in resolveconf.

Test our DNS servers using **nslookup** command



———
To grant root level permission to normal user

sudo visudo `ALL=(ALL:ALL) ALL`

——
Chef installation and configuration
https://www.digitalocean.com/community/tutorials/how-to-understand-the-chef-configuration-environment-on-a-vps

————

Reverse Proxy -

Great material - http://nginx.org/en/docs/beginners_guide.html#proxy

```
main context
    events
    http
        server
            location
```

—here is proxed server behind new port 8080 and all its content is under root /data/upl folder

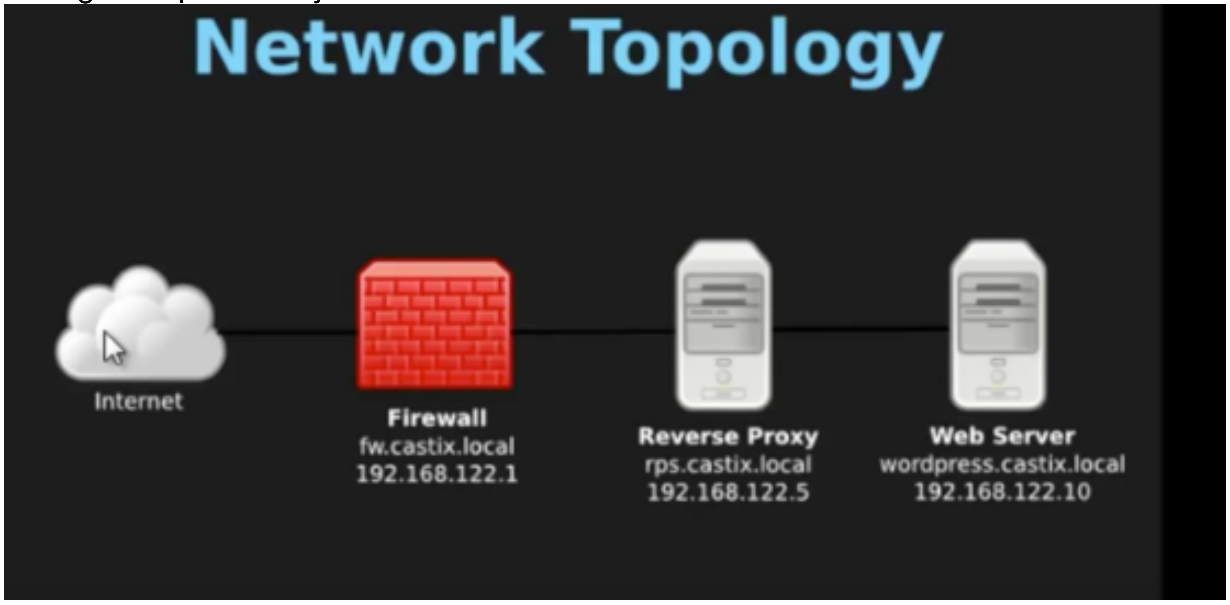
```
server {
    listen 8080;
    root /data/up1;

    location / {
    }
}
```

—here is first block on nginx.conf file — which will direct all requests to proxied server

```
server {
    location / {
        proxy_pass http://localhost:8080;
    }

    location /images/ {
        root /data;
    }
}
```

/data/www
/data/images

On Nginx https://www.youtube.com/watch?v=W5E0M9u8nZw



———
YUM Repositories
Most common and largest CentOS YUM Repositories:

- CentOS Official Repository Mirrors
- EPEL Repository Mirrors
- RPMforge Repository
- ElRepo Repository

———

Jenkins set up on Apache as a proxy server

- Install Apache2
- Enable proxy and proxy_http
- Configure con file in apache2 sites-available
- reload apache2 service

Watch https://www.youtube.com/watch?v=0ZS2BL5R3Ow&nohtml5=False

to check if any specific service starts during boot -
chkconfig vsftpd on

to see if auto boot set up . use below command and check above service
ntsysv

to store ssh keys - use folder .ssh/authorized_keys or .ssh/ida_rsa and .ssh/

ida_rsa_pub
——

Puppet video

There is puppet master and puppet agent.

https://www.youtube.com/watch?v=pnn7fqBKI4E&index=3&list=PLtNErhYMkHnEu1_ZHSJt1xF--zU35dRgZ

www.chadthompson.me

https://www.youtube.com/watch?v=61X2Armexf8&list=PLijGikwhOQLbXm2xuUeMV4YMCnz-eMP2J

```
sudo service puppet server start
puppet resource service puppetserver ensure=running
puppet resource service puppetserver enable=true
```

Most important component of puppet is NTP - This is network time protocol.. Puppet uses this to sync up config between master and agent.
https://docs.puppet.com/pe/latest/quick_start_ntp.html

——————————Docker

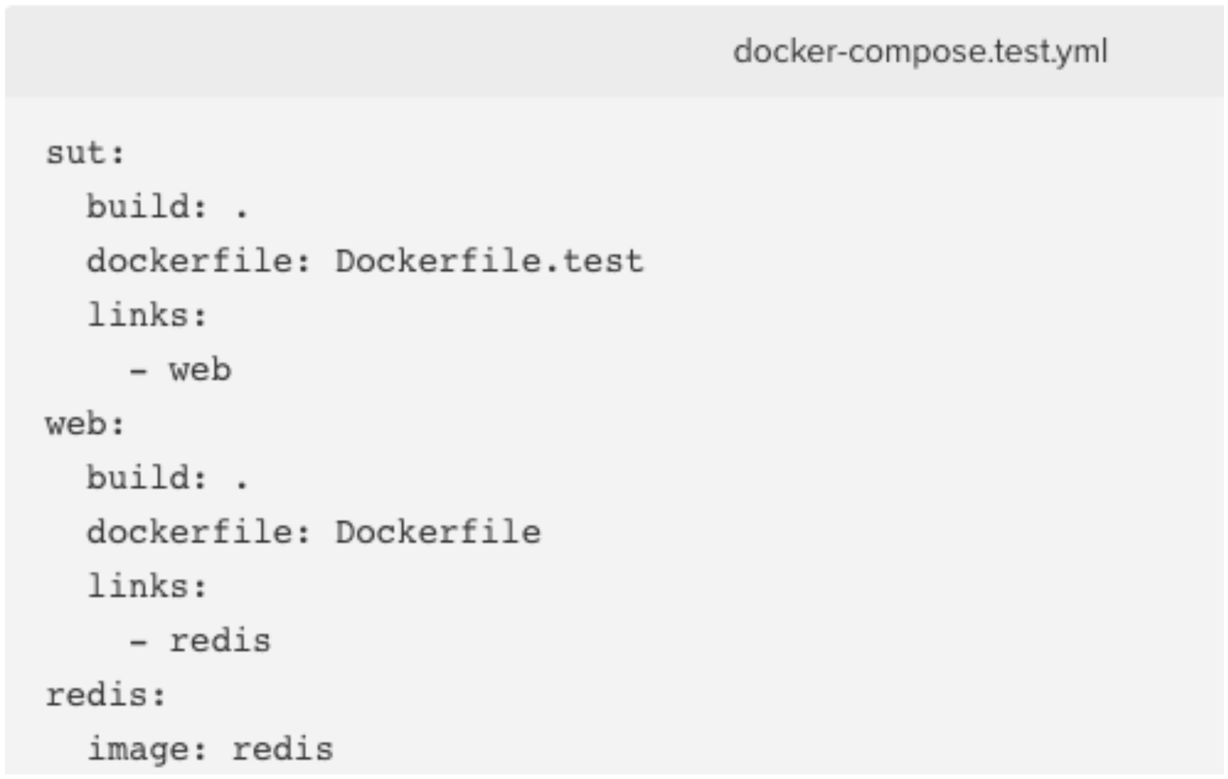
Docker is a containerization platform which aims to simplify the problems of environment standardization so the deployment of applications can also be standardized (find out more about Docker). For developers, Docker allows you to simulate production environments on local machines by running application components in local containers. These containers are easily automatable using Docker Compose, independently of the application and the underlying OS.

Dockerfile is not yet-another shell. Dockerfile has its special mission: automation of Docker image creation.Once, you write build instructions into Dockerfile, you can build the same image just with docker build command.Dockerfile is also useful to tell the knowledge of what a job the container does to somebody else. Your teammates can tell what the container is supposed to do just by reading Dockerfile. They don't need to know login to the container and figure out what the container is doing by using ps command.
ADD is the instruction to add local files to Docker image. This image should reside in Context of the docker. Context in Dockerfile means files and directories available to the Dockerfile instructions.

Basically, we need image (application) and then definition of what needs to be done on that image. Example we use
Docker file — this is nothing but sh program with command specific format of Docker .. like it has FROM. WORKDIR , ADD , RUN , CMD commands
requirements.txt — If there are multiple application and multiple docker container required then speciiy those application into this file
Docker-compose.yml — This is like how to resolve dependency between those multiple application

Commads -
`docker-compose` — This is to build multiple docker containers.
`sudo docker-compose up` — To run docker daemon
`export COMPOSE_API_VERSION=1.18`
`docker ps` — to see all docker process and list of containers

To Run test , create Docker.test and docker-compose.test.yml … Docker.test will have all steps and sh program to do testing.. docker-compose.test will simply identify environment and dependencies as shown below


docker-compose.test.yml

```
sut:
  build: .
  dockerfile: Dockerfile.test
  links:
    - web
web:
  build: .
  dockerfile: Dockerfile
  links:
    - redis
redis:
  image: redis
```

here `sut` stands for `System under test. All we are doing here is creating dependencies using links switch.`
Note that docker-compose.test.yml might include dozens of external services and multiple test containers. Docker will be able to run all these dependencies on a single

host because every container shares the underlying OS.

```
docker-compose -f ~/hello_world/docker-compose.test.yml
-p ci build
```

This command builds the local images needed by docker-compose.test.yml. -f to point to docker-compose.test.yml and -p to indicate a specific project name.

Now spin up your fresh testing environment by executing:

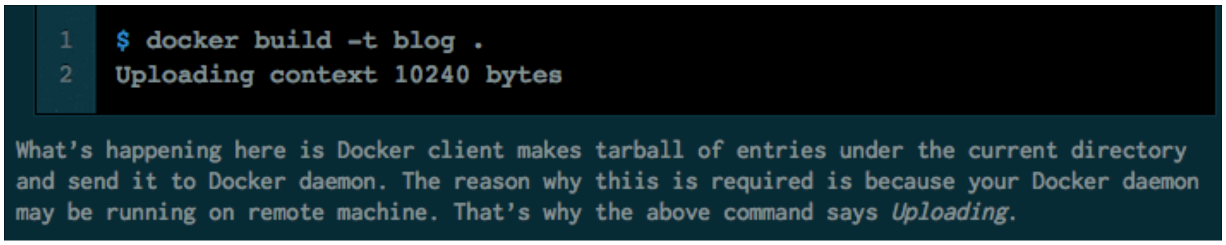
```
docker-compose -f ~/hello_world/docker-compose.test.yml
-p ci up -d
```

Once docker is up , then there are commands like

```
docker start
docker stop
docker rm
```

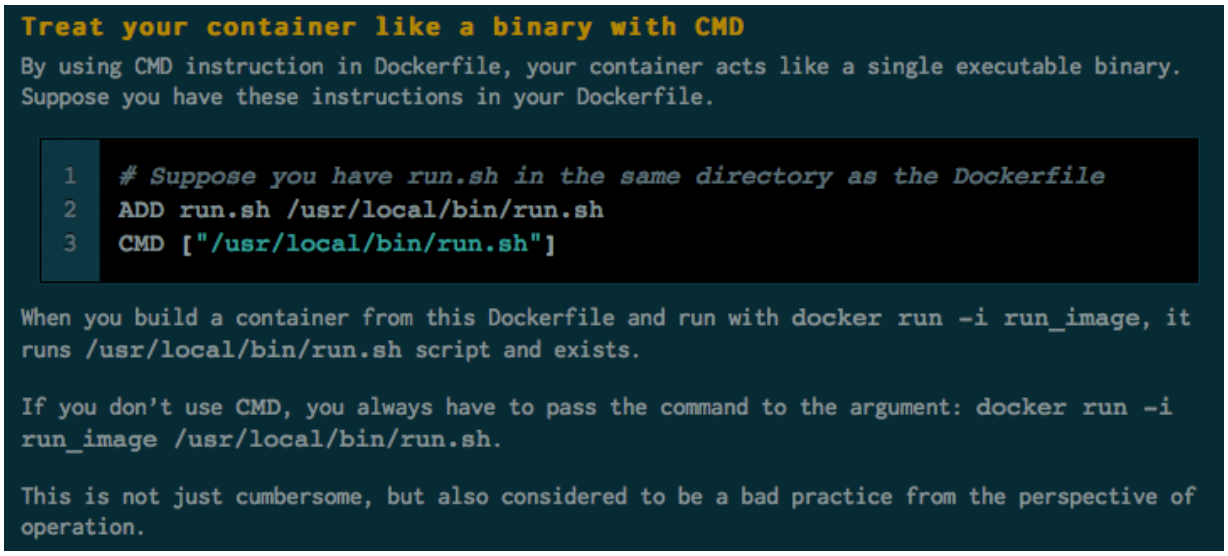
see https://docs.docker.com/engine/reference/commandline/stop/

ADD command is to add local files into docker context . Example you can add RPM package using ADD. This rpm packages should be placed in same docker directory



```
$ docker build -t blog .
Uploading context 10240 bytes
```

What's happening here is Docker client makes tarball of entries under the current directory and send it to Docker daemon. The reason why thiis is required is because your Docker daemon may be running on remote machine. That's why the above command says *Uploading*.



Treat your container like a binary with CMD

By using CMD instruction in Dockerfile, your container acts like a single executable binary. Suppose you have these instructions in your Dockerfile.

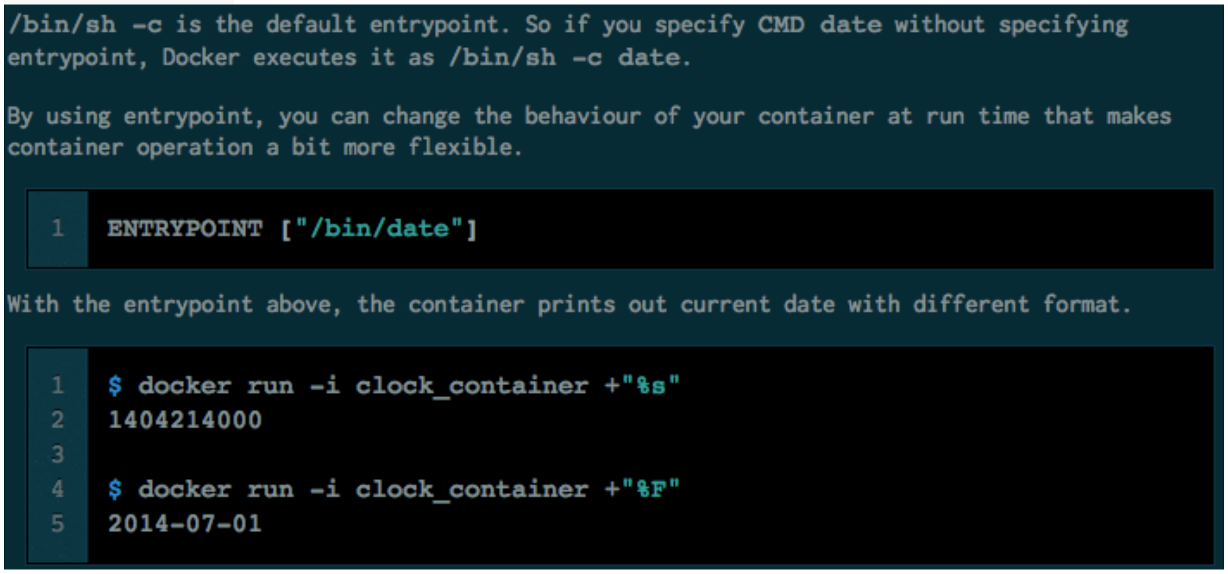
```
# Suppose you have run.sh in the same directory as the Dockerfile
ADD run.sh /usr/local/bin/run.sh
CMD ["/usr/local/bin/run.sh"]
```

When you build a container from this Dockerfile and run with docker run -i run_image, it runs /usr/local/bin/run.sh script and exists.

If you don't use CMD, you always have to pass the command to the argument: docker run -i run_image /usr/local/bin/run.sh.

This is not just cumbersome, but also considered to be a bad practice from the perspective of operation.

The main purpose of a `CMD` is to provide defaults for an executing container. These defaults can include an executable, or they can omit the executable, in which case you must specify an `ENTRYPOINT` instruction as well.



/bin/sh -c is the default entrypoint. So if you specify CMD date without specifying entrypoint, Docker executes it as /bin/sh -c date.

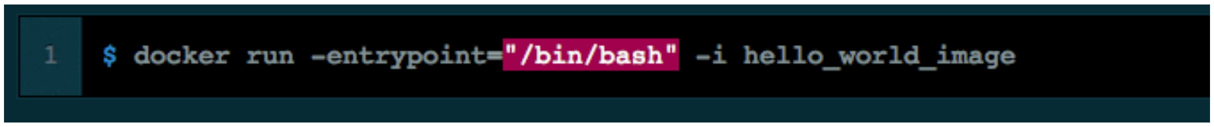
By using entrypoint, you can change the behaviour of your container at run time that makes container operation a bit more flexible.

```
ENTRYPOINT ["/bin/date"]
```

With the entrypoint above, the container prints out current date with different format.

```
$ docker run -i clock_container +"%s"
1404214000

$ docker run -i clock_container +"%F"
2014-07-01
```

If you are running any shell script in the ENTRYPOINT then always specify shebang as shown below

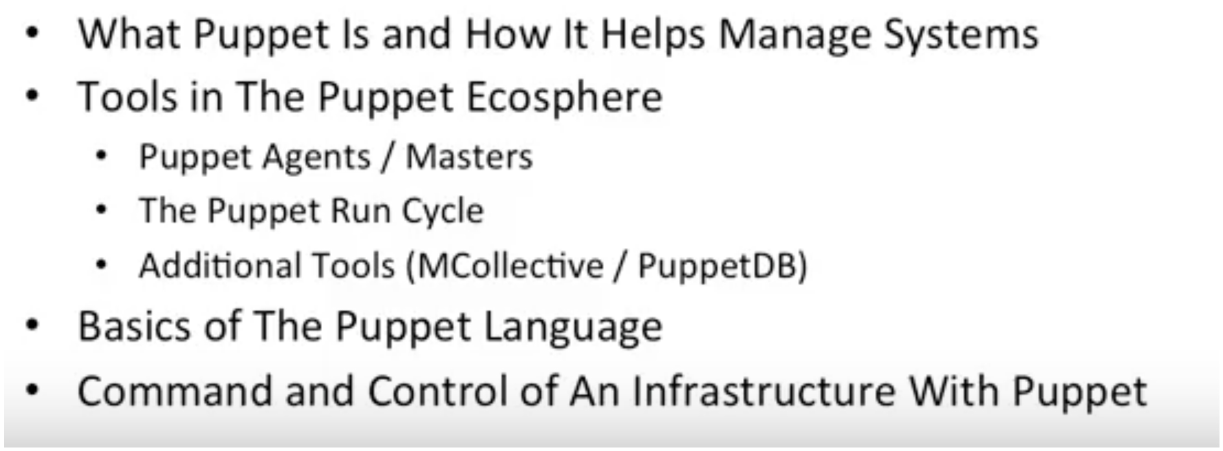


```
$ docker run -entrypoint="/bin/bash" -i hello_world_image
```

— — — — —Puppet —
Puppet, from Puppet Labs, is a configuration management tool that helps system administrators automate the provisioning, configuration, and management of a server infrastructure. Planning ahead and using config management tools like Puppet can cut down on time spent repeating basic tasks, and help ensure that your configurations are consistent and accurate across your infrastructure. Once you get the hang of managing your servers with Puppet and other automation tools, you will have more free time that can be used to improve other aspects of your setup

Puppet comes in two varieties, Puppet Enterprise and open source Puppet.

Puppet master server—which runs the Puppet Server software—can be used to control all your other servers, or Puppet agent nodes



- What Puppet Is and How It Helps Manage Systems
- Tools in The Puppet Ecosphere
  - Puppet Agents / Masters
  - The Puppet Run Cycle
  - Additional Tools (MCollective / PuppetDB)
- Basics of The Puppet Language
- Command and Control of An Infrastructure With Puppet

———
iptables for connections =
http://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/

There is INPUT, FORWARD AND OUTPUT

——
Jenkins by default listen on some port.. It can be changed to 8081 by going to /etc/default/jenkins file
If tomcat by default listen on 8080 port. it can be changed to other port by going to server.xml file. May need to change ownership of this file. This server file is located in tomcat folder

——
nowdoc functions like double quote and with variable whereas nowdoc works like a single quote and cannot use variables
—

https://www.youtube.com/watch?v=AI8Kjag1vGk&index=3&list=PL92E89440B7BFD0F6&nohtml5=False
Maven commands
mvn —version
mvn archetype:generate (here browse the list and select the number that we wish to use to build your application)
mvn compile
mvn test
mvn package
mvn install
java -cp <name of target snapshot jar> <name of grouped app>

— 1. Archetype - folder structure ; which framework or repository we wish to use , web application archetype , Spring, Hybernate etc

— 2. Group ID - Project id
— 3. Artifact ID (output of your project) - What type of application we want to build e.g. web application (war) ; enterprise application (er) , java application (Jar)
— 4. Version (mainly used when we release our code)
— 5. Package - what package class should belong to … where we wish to place our source code.

Build lifecycle or phases -
Validate, Compile, Test(snapshot), Package, Install on local repository, Deploy on repository
Clean - This will delete target folder that Maven generated for phases above

in the Pom.xml , compile is default scope. So no need to specify in <dependency> section

If you add new plugin into POM.xml file then that can be run by using mvn <name of the plugin>:run command
In install eclipse plugin in maven, use mvn eclipse:eclipse command

— —
Maven videos -
https://www.youtube.com/watch?v=0CFWeVgzsqY
https://www.youtube.com/watch?v=mpNnbBmh5J4&index=6&list=PL92E89440B7BFD0F6&nohtml5=False
— — — — —
What is MAC Address -
It is used by network adapter for communication.

Run command below
arp -a
This will list all IP and MAC address

Network tutorial
https://www.youtube.com/watch?v=3QWrq5gN8VY&list=PLEbnTDJUr_IegfoqO4iPnPYQui46QqT0j&index=3

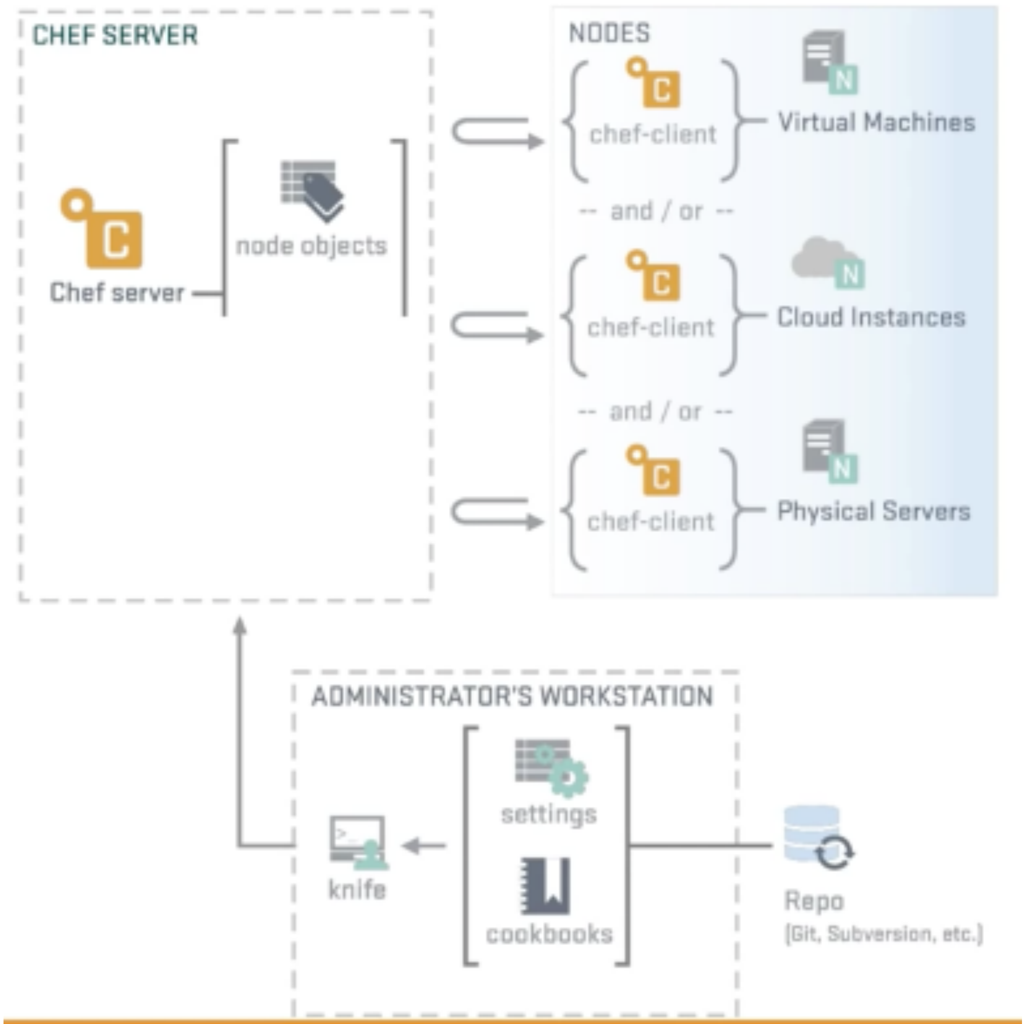
— — — — — — —

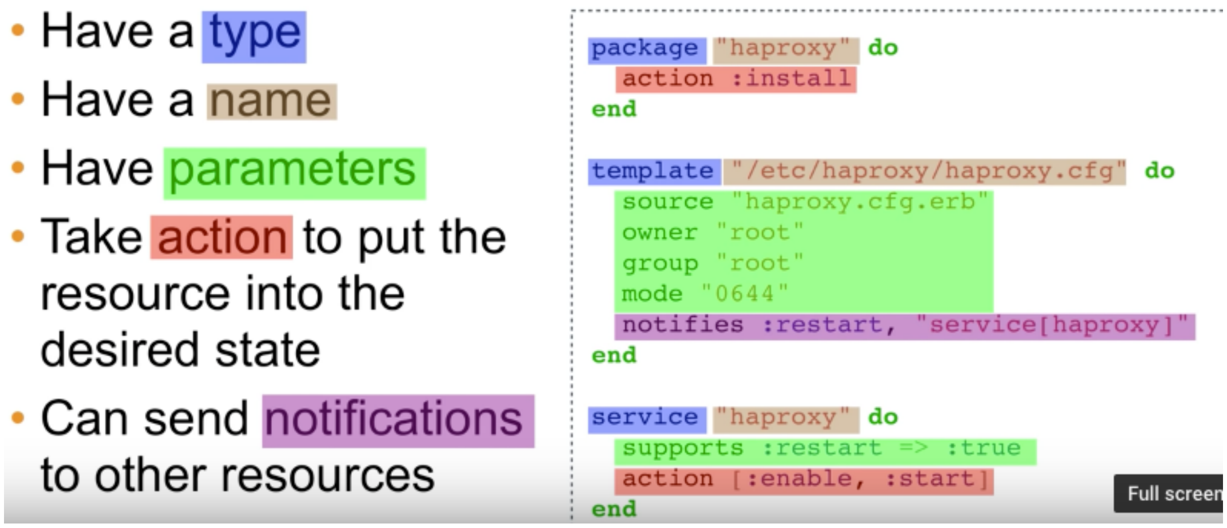
— — —Chef — tutorial - https://www.youtube.com/watch?v=71Cq4bCxgDk&list=PLrmstJpucjzWKt1eWLv88ZFY4R1jW8amR&index=4
https://www.youtube.com/watch?v=4RrzK1ozitE&index=3&list=PLrmstJpucjzWKt1eWLv88ZFY4R1jW8amR

check https://www.youtube.com/watch?v=egvEPsVMfK0 — installation video

After installation run following commands
sudo chef-server-ctl reconfigure

sudo chef-server-tcl status





- Have a type
- Have a name
- Have parameters
- Take action to put the resource into the desired state
- Can send notifications to other resources

```
package "haproxy" do
  action :install
end

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

————————

command ‘Sar’ Collect, report, or save system activity information. The default version of the sar command (CPU utilization report) might be one of the first facilities the user runs to begin system activity investigation, because it monitors major system resources. By default log files of Sar command is located at /var/log/sa/sadd file

Good linux concepts -
http://www.linuxtechi.com/experience-linux-admin-interview-questions/

LVM stands for Logical Volume Manager , to resize filesystem’s size online we required LVM partition in Linux. Size of LVM partition can be extended and reduced using the lvextend & lvreduce commands respectively.

umask stands for ‘User file creation mask’, which determines the settings of a mask that controls which file permissions are set for files and directories when they are created.

Using ‘showmount’ command we can see what directories are shared via nfs e.g ‘showmount -e <ip address of nfs server>’.Using mount command we can mount the nfs share on linux machine.

Using the Commands ‘netstat -nr’ and ‘route -n’ we can see the default route and routing tables.
Command to see which port is listening on linux ——> ‘netstat –listen’ and ‘lsof -i’

With the help of command ‘chkconfig –list l grep 5:on’ we can list all the service that are enabled in run level5. For other run levels just replace 5 with the respective run level.We can enable a service using the Command ‘chkconfig <Service-Name> on –level 3’.
We should never upgrade Linux Kernel , always install the new New kernel using rpm command because upgrading a kenel can make your linux box in a unbootable state.
Load Average is defined as the average sum of the number of process waiting in the run queue and number of process currently executing over the period of 1,5 and 15 minutes. Using the ‘top’ and ‘uptime’ command we find the load average of a linux sever.