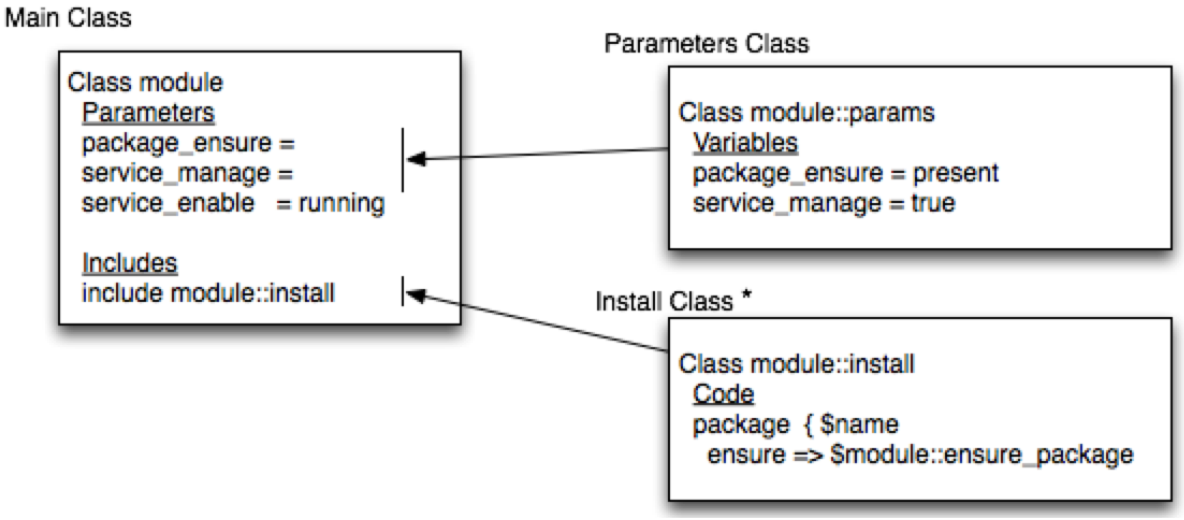
— — — — —Puppet

Beginner guide to Puppet Module - https://docs.puppet.com/guides/module_guides/bgtm.html

So Puppet is all about writing modules. Modules is made up of various step by step pp files. There is main.pp and then install.pp , config.pp etc

Each of these pp files explains how to change state of the server. Now , there are pre-coded modules at github that can be used directly (https://github.com/example42?page=2) Once Module is coded then there are testing frameworks like Rspec_puupet or Rspec_system , Serverspec etc

http://stackoverflow.com/questions/13318450/how-far-should-i-go-with-puppet



What is Hiera ?
https://docs.puppet.com/hiera/3.1/

Here is sample module example of installing and configuring Ansible using Puppet

https://github.com/example42/puppet-ansible

Here is sample module example to launch docker using puppet
https://github.com/example42/puppet-docker

So what is puppet templates ? - -https://docs.puppet.com/puppet/latest/reference/lang_template.html

———— SNMP - SIMPLE NETWORK MANAGEMENT PROTOCOL

Introduction video - https://www.youtube.com/watch?v=tg47MZdtcAE

https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-an-snmp-daemon-and-client-on-ubuntu-14-04

https://www.digitalocean.com/community/tutorials/how-to-use-the-net-snmp-tool-suite-to-manage-and-monitor-servers

———Scala

https://www.youtube.com/watch?v=Ytfw8Bg86p4

See this video https://www.youtube.com/watch?v=KOJeKey87eI

Every value in the scala is an object - https://www.youtube.com/watch?v=6TJVuuqHxKo&index=13&list=PL0B0820169DCF0AD2

5.ToString —here 5 is value and object so we are calling method of that object.

```
scala> "This is a test".replace("te","he")
res27: java.lang.String = This is a hest

scala> (5,8.5)
res28: (Int, Double) = (5,8.5)

scala> (5,8.5,"hi")
res29: (Int, Double, java.lang.String) = (5,8.5,hi)

scala> res28._1
res30: Int = 5

scala> res28._2
res31: Double = 8.5

scala> res29._3
res32: java.lang.String = hi

scala> res29._3.length
res33: Int = 2
```

Every object should be instance of a class . Scala makes that object a single instance of that class. As shown below, all methods and variables defined after object Logger { becomes object's class.

https://www.youtube.com/watch?v=axzjF-389U8&index=14&list=PL0B0820169DCF0AD2

Binary is base 2 and hexadecimal is base16

There are two types of variables in scala .. 1. Val as shown below. defining :Int is optional.

```
scala> val myvariable = 1000
myvariable: Int = 1000

scala> val myvariable:Int = 1000
myvariable: Int = 1000
```

Difference between VAR and VAL is that VAR allow us to re-assign value to variable whereas VAL doesn't allow us to re-assign values.

If you wish to interpolate the string in scala then use **s** before string and then **$** to represent value of the variable inside string as shown below

```
scala> val age = 42
age: Int = 42

scala> val mytext = s"This is called string interpolation,my age is $age"
mytext: String = This is called string interpolation,my age is 42
```

To evaluate any expression , use ${ }

-There are around 450 functions available for VAL and VAR each to use, example one shown below using readInt function

```
scala> var age = readInt()
```

See how we can use IF statement as a express in VAL variable

```
println("whats ur age")
val age = readInt()
val response = if (age>18){ "pass"} else { "failed" }
println(response)
```

Functions - It is defined by keyword DEF . Scala requires type to be specified for the function parameter and what will be type of the outcome of that function It is optional if you use VAL or VAR variables but not in functions. After = sign, it is expression written in { }

```
scala> def myfunction(x:Int):Int = x*x
myfunction: (x: Int)Int

scala> def myfunction(x:Int):Int = { x*x }
myfunction: (x: Int)Int

scala> myfunction(3)
res30: Int = 9
```

Now example of function

```
scala> def mysecondfunction(x:Int,y:Int):String = if(x>y) "passed" else "failed"
mysecondfunction: (x: Int, y: Int)String

scala> mysecondfunction(3,4)
res39: String = failed
```

Output of function can also be tuple as shown below in the second function which is also calling first function

```
def quadrant(x:Double,y:Double):Int = {
  if(x<0) {
    if(y<0) 3 else 2
  }  else {
    if(y<0) 4 else 1
  }
}

def quadrantAndMagitude(x:Double, y:Double):(Int,Double) = {
 val quad = quadrant(x,y)
 val mag = math.sqrt(x*x+y*y)
 (quad, mag)
}

println("Enter x-coordinate")
val ix = readDouble
println("now enter y cordinate")
val iy = readDouble
println(quadrantAndMagitude(ix,iy))
```

another example

```
def firstfun(x:Int, y:Int):Int = y -x



def getagefunction(x:Int,y:Int):(Int,String) = {
val age = firstfun(x,y)
val status = if(age>25) "qualified" else "not qualified"
(age,status)
}

println("when did you born?")
val byear = readInt()
println("current year?")
val cyear = readInt()
println(getagefunction(byear,cyear))
```

Non-functional functions are called procedures . In some cases, Unit is used to hold output of functions since there is no specific type that can be used. Println always gives back Unit as output . It;s like void in java.. It does not return any value.. it just print out stuff….Example as shown below

```
scala> val u = println("hi")
hi
u: Unit = ()
```

specifying unit is not mandatory .. it can be skipped as shown in second example below

```
scala> def greeting(name:String):Unit = {
     | println("Hello "+name+".")
     | }
greeting: (name: String)Unit

scala> greeting("Mark)
<console>:1: error: unclosed string literal
       greeting("Mark)
                ^

scala> greeting("Mark")
Hello Mark.

scala> def greeting(name:String){
     | println("Hello "+name+".")
     | }
greeting: (name: String)Unit
```

If we see below, output of printquadrant is unit so we cannot pass that to quad variable. This code will fail

```
def randomInQuadrant(x:Double, y:Double):(Double,Double) = {
  val quad = printQuadrant(x,y)
  if(quad==1) (math.random, math.random)
  else if(quad==2) (-math.random, math.random)
  else if(quad==3) (-math.random, -math.random)
  else (math.random, -math.random)
}
```

There are 2 ways to define function literals as shown below

```
scala> def f(x:Double):Double = x*x
f: (x: Double)Double

scala> (x:Double) => x*x
res5: Double => Double = <function1>
```

Second method above of using rocket symbol (=>) will be used moving forward ….

```
scala> f(3)
res3: Double = 9.0

scala> (x:Double) => x*x
res4: Double => Double = <function1>

scala> res4(3)
res5: Double = 9.0

scala> def myfun = (x:Double) => x*x
myfun: Double => Double

scala> myfun(3)
res6: Double = 9.0
```

Now see some alternative way to use variable as function

```
scala> val P = (x:Int) => x*x
P: Int => Int = <function1>

scala> P(3)
res7: Int = 9

scala> val N = (y:Int) => y+y
N: Int => Int = <function1>

scala> N(4)
res8: Int = 8
```

```
scala> P(N(6))
res10: Int = 144
```

There is a built-in compose function as shown below

```
scala> val S = P.compose(N)
S: Int => Int = <function1>

scala> S(2)
res13: Int = 16
```

Higher Order Functions — meaning function as a argument to main function.
Now instead of passing parameters in function, we will pass another function. Syntax is whenever we pass another function as argument, include what type goes in and what type comes out. Finally output of main function types(in & out)

As shown below, f and g is function and return value is (in => out) which is also a function

```
scala> def compose(f:Double => Double, g:Double => Double):(Double => Double) =
{
     | (x:Double) => f(g(x))
     | }
compose: (f: Double => Double, g: Double => Double)Double => Double

scala> val h2 = compose(f,g)
h2: Double => Double = <function1>

scala> h2(3)
res12: Double = 64.0
```

```
scala> val f = (x:Double) => x*x
f: Double => Double = <function1>

scala> f(2)
res0: Double = 4.0

scala> val g = (x:Double) => x+x
g: Double => Double = <function1>

scala> g(3)
res1: Double = 6.0

scala> def compose(f:Double => Double, g:Double => Double):(Double => Double) =  {
     | (x:Double) => f(g(x))
     | }
compose: (f: Double => Double, g: Double => Double)Double => Double

scala> val mynewval = compose(f,g)
mynewval: Double => Double = <function1>
```

```
scala> mynewval(2)
res3: Double = 16.0
```

so g and then f function = (2+2) = 4 * 4 = 16

Factorial program below. In below code, we are calling same function in if block

```
def factorial(n:Int):Int = if(n>1) n*factorial(n-1) else 1

println("Enter a number.")
val n = readInt
println(n+"! = "+factorial(n))
```

```
object Logger {
  private def show(level: String, message: String) =
    println(level + ": " + message)

  def debug(message: String) = show("DEBUG", message)
  def info(message: String)  = show("INFO", message)
  def warn(message: String)  = show("WARN", message)
  def error(message: String) = show("ERROR", message)
  def fatal(message: String) = show("FATAL", message)
}
```

so now we don't have to define separate instances as shown below,….there are 2 instances created in two classes using logger keyword…

```
class Class1 {
  val logger = new Logging
  def method1(i: Int) = logger.info("i: " + i)
}

class Class2 {
  val logger = new Logging
  def method2(j: Int) = logger.info("j: " + j)
}
```

Above black can be updated as below - here we are now using object Logger directly and accessing methods of it. such as info.

```
class Class1 {
  def method1(i: Int) = Logger.info("i: " + i)
}

class Class2 {
  def method2(j: Int) = Logger.info("j: " + j)
}
```

————To add path variable into Mac - `echo 'export PATH=/Users/<your username>/scala/bin:$PATH' >> ~/.bash_profile`

————Blue-Green deployment

https://www.thoughtworks.com/insights/blog/implementing-blue-green-deployments-aws

————————Flask

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

See how to call this application in regular python at http://pymbook.readthedocs.io/en/latest/flask.html

——————Python

Everything is an object. Check this small video - https://www.youtube.com/watch?v=3oV23UcqTGg

Understand class, instance, object , attribute and method here - https://www.youtube.com/watch?v=VuaRZhROqPg

Some good Regular expression document - https://docs.python.org/2/library/re.html

Python offers two different primitive operations based on regular expressions: `re.match()` checks for a match only at the beginning of the string, while `re.search()` checks for a match anywhere in the string

There are good python libraries like OS SYS etc that gives info on operating system and system variables

CGI stands for- Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers.

Web Programming with Python

Authored by Mike McMillan

Programming Review
Database Programming with SQLite
FTP
Email
Newsgroups
Web Client Programming
Web Server Programming
Web / Database Programming
XML Programming
Web Framework Programming - Django

Some interesting python class cmd

https://pymotw.com/2/cmd/

All Python modules can be found here - https://pymotw.com/2/sys/index.html#module-sys

OS and SYS module info - http://www.thomas-cokelaer.info/tutorials/python/module_os.html

————————Java concept

https://www.youtube.com/watch?v=6KL4kdec9O4&list=PLtNErhYMkHnHq4jqUTr17FvGMOveQ3cc_&index=12&spfreload=5

————————SELinux —

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

Video tutorial - https://www.youtube.com/watch?v=O8YQVpneH5w

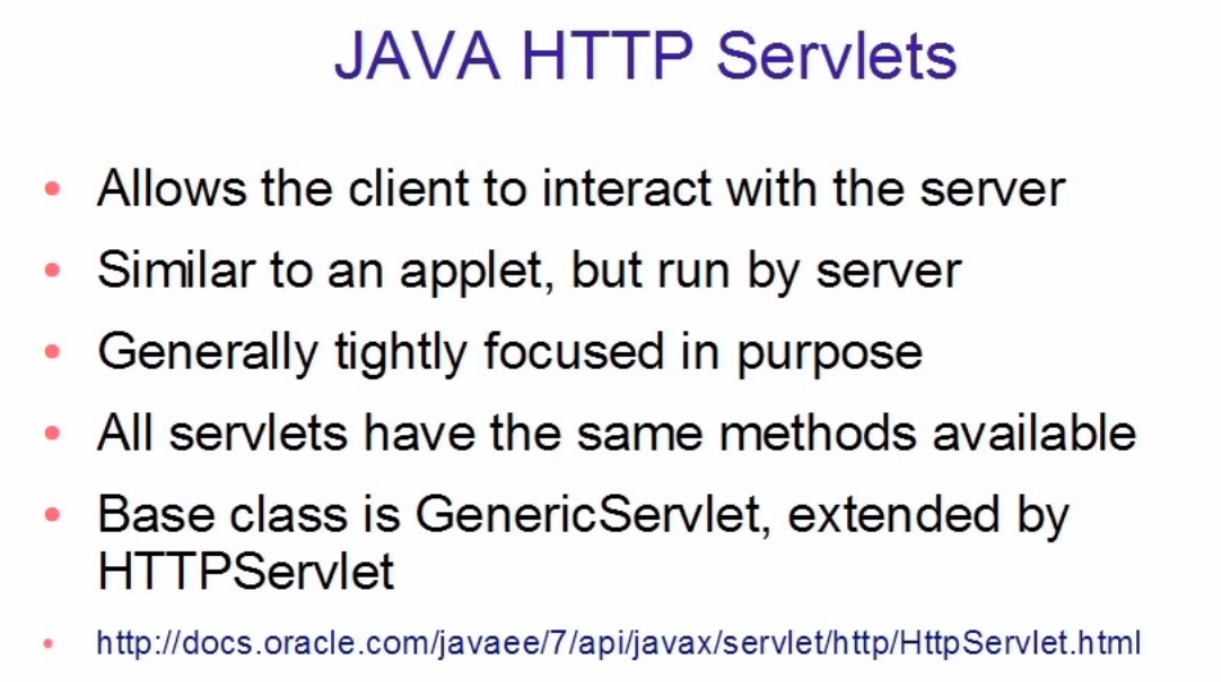
It all about label and booleans

Commands - To show current mode - GETENFORCE or /etc/sysconfig/SELinux . There are 3 modes 1.Enforcing 2. Permissive and 3. Disabled.

In case of enforcing - all SELinux policies are enforced and permissive is warning and disabled is no SELinux policy are enabled. To set enforcing - use SETENFORCE. There are 3 policies - 1. Targeted policy .2. minimum policy 3. MLS (multi-level security)

To understand policy — there are labels . These labels are set in Context. So context is shown when u use ls -z command. Context consists of user, roles and types. For SELinux we focus on type. So these source types are tied to target types. SELinux policy is to define which source type has access to which target type.

————————Java Servlet



——————Docker

Command Docker build builds an image from Dockerfile and a context. The build's context is the files at a specified location PATH or URL. The PATH is a directory on your local filesystem. The URL is a the location of a Git repository.

```
$ docker build -f /path/to/a/Dockerfile .
```

Build ——> run by Docker Daemon

Once build is successful , it can be tagged (using -t option) and saved at new image at some place

```
$ docker build -t shykes/myapp .
```

Docker Daemon runs instruction from Dockerfile one by one .

```
$ docker build -t svendowideit/ambassador .
Sending build context to Docker daemon 15.36 kB
Step 0 : FROM alpine:3.2
 ---> 31f630c65071
Step 1 : MAINTAINER SvenDowideit@home.org.au
 ---> Using cache
 ---> 2a1c91448f5f
Step 2 : RUN apk update &&      apk add socat &&        rm -r
 ---> Using cache
 ---> 21ed6e7fbb73
Step 3 : CMD env | grep _TCP= | (sed 's/.*_PORT_\([0-9]*\)_TC
 ---> Using cache
 ---> 7ea8aef582cc
Successfully built 7ea8aef582cc
```

When build is done, image can be pushed to Docker repository if required for public to use.

Rest all is about Dockerfile template at — https://docs.docker.com/engine/reference/builder/

—— Lets discuss more on Dockerfile Format ….

```
FROM busybox
ENV foo /bar
WORKDIR ${foo}   # WORKDIR /bar
ADD . $foo       # ADD . /bar
COPY \$foo /quux # COPY $foo /quux
```

DockerCompose -

Docker Compose is an open-source tool for defining and running multi-container applications using a declarative approach.

.dockerignore file is used by Daemon to put exclusion filter on Dockerfile content

So Docker is all about pushing image and then execute Shell script using RUN command on top of that image. "Layering RUN instructions and generating commits conforms to the core concepts of Docker where commits are cheap and containers can be created from any point in an image's history, much like source control."

—— what is apt-key

http://man.he.net/man8/apt-key

```
apt-key is used to manage the list of keys used by apt to
authenticate packages. Packages which have been authenticated
using these keys will be considered trusted.
```

—— sed command

check video https://www.youtube.com/watch?v=4vr8Aao0Mfo

example.txt may look like below

```
first is blue
second is green
thid is yellow
```

E.g.1 cat example.txt | sed 'n; N; P'

First line comes to patten space. Patern space is like a runtime space before stdout .

-n - this will hide output from patten space(pattern space is middle runtime space) and dump first line into output space. check video

now second line comes into pattern space. remember once output is done, next line comes into pattern space. it does not do any other command like N till second lines arrives in pattern space

N - will bring next line into patten space only. already there is 2nd line , but N will bring next line and keep there. It does not go to output .Output goes only from n or p or =

P- will print first line from pattern space since there are now 2 lines(2nd and 3rd). If it were small

p then it would have taken all lines from pattern space to output space.

finally whatever is left in pattern space will be outputted since we cannot keep anything in pattern space. So now 2nd and 3rd lines go to pattern space.

Output may look like -

```
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt | sed 'n;N;P'
first is blue
second is green
second is green
thid is yellow
```

E.g. 2 cat example.txt | sed -n '1,3p' This will print 1st line till 3rd line

s - substitute command

I - ignore case

y - This is same as tr command . it modifies input character to output character

```
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt | sed 'y/f/X/'
Xirst is blue
```

w - This command writes portion of the file into second file if needed. example anything after / second goes to next file .

```
PAOC02PTAMJG8WP:Desktop amgonsal$ <test.txt sed '/second/ w
newfile.txt'
first is blue
second is green
thid is yellow
fourth is f
five is fibe
six is six
PAOC02PTAMJG8WP:Desktop amgonsal$ cat newfile.txt
second is green
```

q - is to stop execution. If it hits that then it will stop execution

```
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt | sed -n 'p;q'
first is blue
```

sed can also allow us to write dedicated script files . # is used for comments. So sed command can be part of command line or it can be in the file and then invoke that script by using sed -f <name of the script file.sed>

If you want to print from any specific line then use below . In example below, there is a line that starts with lxx.

```
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt
first is blue
second is green
thid is yellow
fourth is f
lxx
five is fibe
six is six
ninith commandline text here
PAOC02PTAMJG8WP:Desktop amgonsal$ <test.txt sed -n '/^l/,$p'
lxx
five is fibe
six is six
ninith commandline text here
```

To debug sed command output, use l as shown below example. As shown below , line that ends with $ is in pattern space area.So because of N specififed in command, 2 lines came into pattern space then those got printed due to p command.

```
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt
first is blue
second is green
thid is yellow
fourth is f
five is fifth line
PAOC02PTAMJG8WP:Desktop amgonsal$ <test.txt sed 'N;l; p'
first is blue$
second is green$
first is blue
second is green
first is blue
second is green
thid is yellow $
fourth is f$
thid is yellow
fourth is f
thid is yellow
fourth is f
```

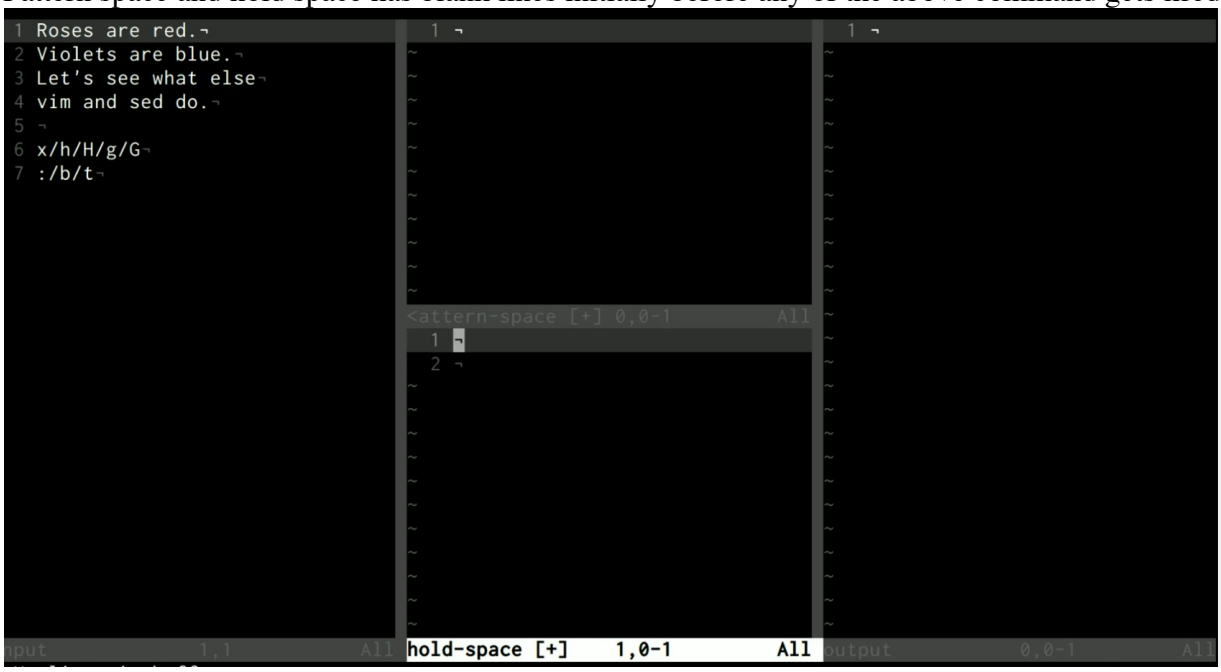
Next commands are a/i/c — These are similar to append/insert/change in vim

```
sed: <poem sed '2aThis is after line 2.'
Roses are red.
Violets are blue.
This is after line 2.
If you like vim,
you might like sed too.
sed: <poem sed '2iThis is before line 2.'
Roses are red.
This is before line 2.
Violets are blue.
If you like vim,
you might like sed too.
```

Commands - x/h/H/g/G
So now we are going to introduce hold space. This hold space is internal and never gets printed into stdout.

x = this command swap the content of pattern space and hold space. Hold space is like set/get memory
h = this sets the content into hold space. This will replace content.
H = this sets the content into hold space. This will append the content.
g = this will get the content from hold space into pattern space. This will replace content
G = this will get the content from hold space into pattern space. This will append content

Pattern space and hold space has blank lines initially before any of the above command gets fired



Now lets fire each command

```
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt
first is blue
second is green
thid is yellow
fourth is f
five is fifth line
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt | sed 'G'
first is blue

second is green

thid is yellow

fourth is f

five is fifth line
```

'x' command swaps the content.. Since hold space has empty line before execution, it swapped that line with 1st line

```
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt
first is blue
second is green
thid is yellow
fourth is f
five is fifth line
PAOC02PTAMJG8WP:Desktop amgonsal$ cat test.txt | sed 'x'

first is blue
second is green
thid is yellow
fourth is f
```

:/b command - This is like function module commands. They are called flow control. So : is used for label and b stands for branch to the label
-e is to separate commands like a |

```
cat test.txt | sed -e :function -e p
```

if we use below line, then b will go to function keyword and start execute again from there . So this below command will form a endless loop

```
cat test.txt | sed -e :function -e p -e bfunction
```

t command - This commands work similar to b command above but only if substitute command is successful . This works as a IF type command. So if substitute is successful then to go label else end the command.

```
[2addr]t [label]
        Branch to the ``:'' function bearing the label if any substitutions have been made
        since the most recent reading of an input line or execution of a ``t'' function.  If no
        label is specified, branch to the end of the script.
```

Using search pattern as shown.

—— Cloud config files

https://www.digitalocean.com/community/tutorials/an-introduction-to-cloud-config-scripting

——— Echo and eval command

echo is to output variable value on screen , whereas eval will consider variable as a command. Example below

```
PAOC02PTAMJG8WP:~ amgonsal$ x="date"
PAOC02PTAMJG8WP:~ amgonsal$ echo $x
date
PAOC02PTAMJG8WP:~ amgonsal$ eval $x
Mon May 30 09:49:30 PDT 2016
```

```
PAOC02PTAMJG8WP:~ amgonsal$ x="java -version"
PAOC02PTAMJG8WP:~ amgonsal$ echo $x
java -version
PAOC02PTAMJG8WP:~ amgonsal$ eval $x
java version "1.8.0_77"
```

—— Vagrant

http://friendsofvagrant.github.io/v1/docs/vagrantfile.html

—— Ansible commands

Ansible is all about making SSH connection to other host and firing commands from main host. Like scp command, there are modules in Ansible to perform tasks on remote hosts once it is connected via ssh

https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-14-04

Handson video - https://www.youtube.com/watch?v=UfD0BASLcJ0

http://docs.ansible.com/ansible/list_of_commands_modules.html

Ansible works by configuring client machines from an computer with Ansible components installed and configured.It communicates over normal SSH channels in order to retrieve information from remote machines, issue commands, and copy files. Because of this, an Ansible system does not require any additional software to be installed on the client computers.

Configuration files are mainly written in the YAML data serialization format due to its expressive nature and its similarity to popular markup languages. Ansible can interact with clients through either command line tools or through its configuration scripts called Playbooks. Ansible uses python whereas Chef and Puppet uses ruby . Ansible uses push mechanism whereas Chef and puppet uses pull mechanism so agents need to pull config.

Writing a playbook using Ansible

https://www.digitalocean.com/community/tutorials/configuration-management-101-writing-ansible-playbooks

—— Puppet (in link below and above, two tools Ansible and Puppet are compared for same type of requirement) .

https://www.digitalocean.com/community/tutorials/configuration-management-101-writing-puppet-manifests

Hands-on

https://www.youtube.com/watch?v=Hiu_ui2nZa0

—— What is BASE64 encoding - Converting binary data into encoding (There are 2 types - Hex and Base64. Base64 uses mapping table so more powerful)

Great video https://www.youtube.com/watch?v=29Nd_sr6GYU

——————REST API using Spring boot (IntellJ - This uses jar file instead of war that other framework like jersey/others uses -)

https://www.youtube.com/watch?v=4A2SH0QS1vo

——————Java What is HashCode and Equal method does

This is applicable when we use contain or comparable interface in collections.

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

or use github https://gist.github.com/runeflobakk/1164848

———————

What is Floating IP concept - High available server

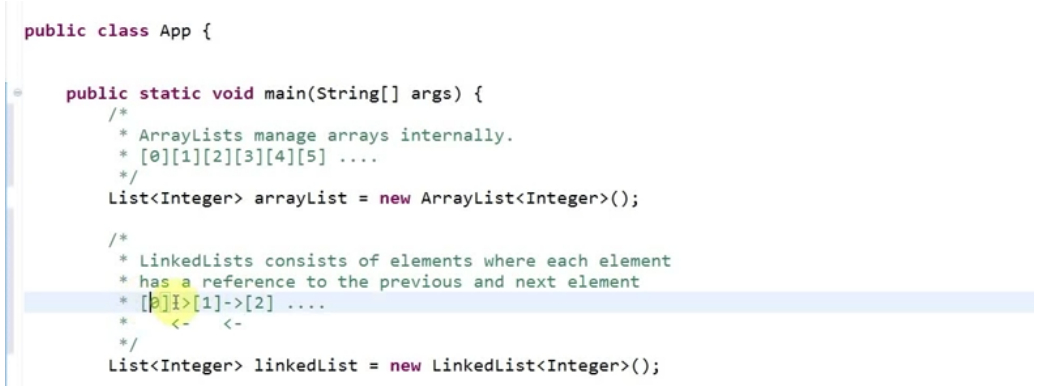
http://en.clouddesignpattern.org/index.php/CDP:Floating_IP_Pattern

A static IP address used in Amazon Web Services (AWS) is known as an "Elastic IP Address (EIP)." You can use this to reassign IP addresses. You can detach an EIP from an existing EC2 instance and attach it to another ECP instance, to swap the virtual server that provides the services.

———————

Iterator is internal class of List Interface .. that uses HasNext , Next method for defining our class level iterator whereas Iterable uses foreach loop for items that are generated by Iterator.

Java collection concept - https://www.youtube.com/watch?v=mkCTxtLe7XU&list=PLB841C370FAFB8EC7



```
public class App {

    public static void main(String[] args) {
        /*
         * ArrayLists manage arrays internally.
         * [0][1][2][3][4][5] ....
         */
        List<Integer> arrayList = new ArrayList<Integer>();

        /*
         * LinkedLists consists of elements where each element
         * has a reference to the previous and next element
         * [0]I>[1]->[2] ....
         *     <-   <-
         */
        List<Integer> linkedList = new LinkedList<Integer>();
```

Collection.sort — If you wish to use this then object should always have natural order. If you use List or Set of String or Integer , they have natural order but not object of any class. This natural order can be established on class when we implement Comparable interface on that class. Implementation of Comparable interface requires CompareTo method which will establish natural order. Check this video . There is issue with Treeset so check below video to understand better

https://www.youtube.com/watch?v=OlMw3ecL1ow&index=8&list=PLB841C370FAFB8EC7

————————————What is difference between AWS Elastic beanstalk , ECS and Docker ?

http://stackoverflow.com/questions/32220557/docker-on-aws-what-is-a-difference-between-elastic-beanstalk-and-ecs

———————

Network Manager

There are 2 utilities/commands for NM -

1. nuclei

2. nmtui

Introduction to Networking - https://www.digitalocean.com/community/tutorials/an-introduction-to-networking-terminology-interfaces-and-protocols#NetworkLayers

NAT: NAT stands for network address translation. It is a way to translate requests that are incoming into a routing server to the relevant devices or servers that it knows about in the LAN. This is usually implemented in physical LANs as a way to route requests through one IP address to the necessary backend servers.

VPN: VPN stands for virtual private network. It is a means of connecting separate LANs through the internet, while maintaining privacy. This is used as a means of connecting remote systems as if they were on a local network, often for security reasons.

Interfaces

Interfaces are networking communication points for your computer. Each interface is associated with a physical or virtual networking device.

Typically, your server will have one configurable network interface for each Ethernet or wireless internet card you have.

In addition, it will define a virtual network interface called the "loopback" or localhost interface. This is used as an interface to connect applications and processes on a single computer to other applications and processes. You can see this referenced as the "lo" interface in many tools.

Many times, administrators configure one interface to service traffic to the internet and another interface for a LAN or private network.

In DigitalOcean, in datacenters with private networking enabled, your VPS will have two networking interfaces (in addition to the local interface). The "eth0" interface will be configured to handle traffic from the internet, while the "eth1" interface will operate to communicate with the private network.

——— Service and ports

80 - HHTP

433 - HTTPS

22 - SSH

80 - Apache

RAID 1 offers redundancy through mirroring, i.e., data is written identically to two drives. RAID 0 offers no redundancy and instead uses striping, i.e., data is split across all the drives. This means RAID 0 offers no fault tolerance; if any of the constituent drives fails, the RAID unit fails.
——————

DNS Client - Server

https://www.digitalocean.com/community/tutorials/an-introduction-to-dns-terminology-components-and-concepts

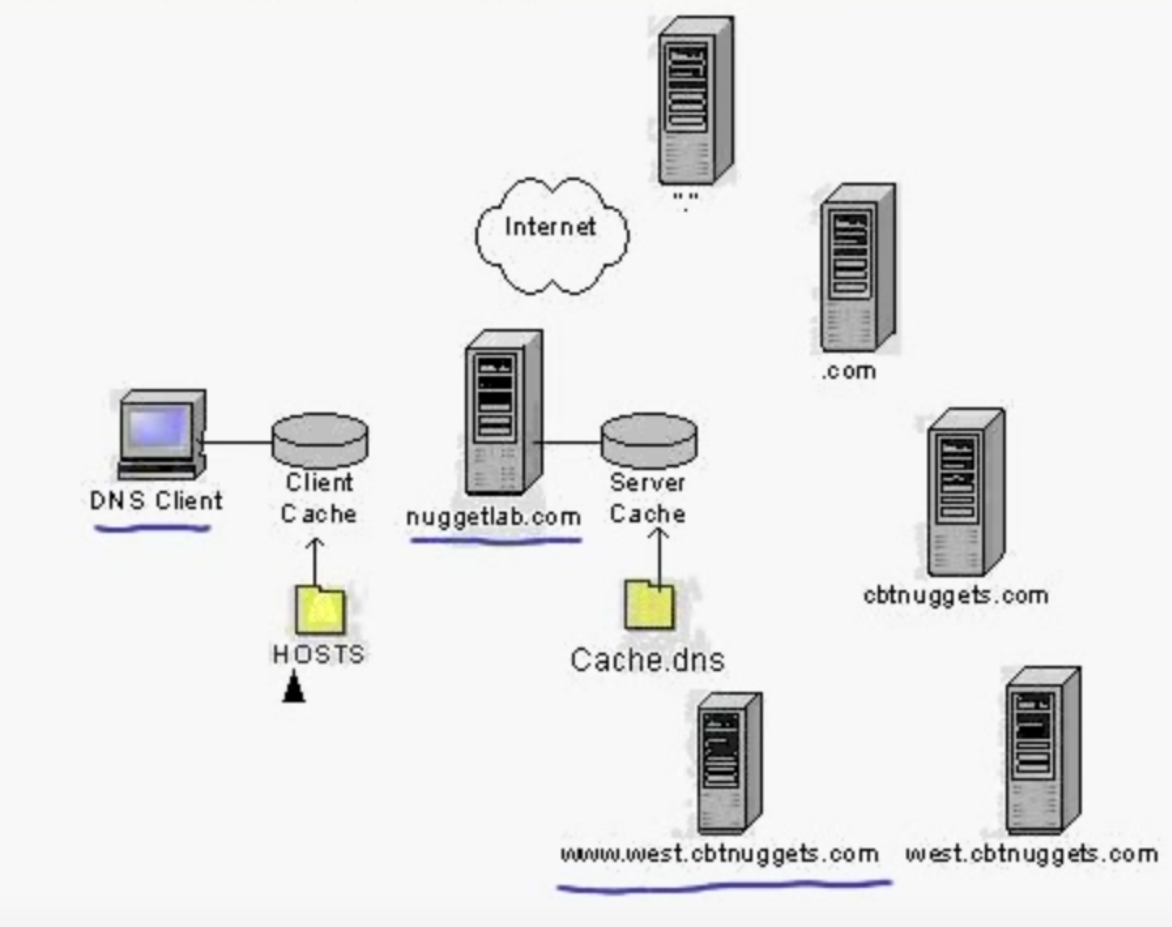
https://technet.microsoft.com/en-us/library/cc787920%28v=ws.10%29.aspx

and https://www.youtube.com/watch?v=ChJSEJ91XWg

The difference between a host name and a subdomain is that a host defines a computer or resource, while a subdomain extends the parent domain. It is a method of subdividing the domain itself.

A proper FQDN ends with a dot, indicating the root of the DNS hierarchy. An example of a FQDN is "mail.google.com.". Sometimes software that calls for FQDN does not require the ending dot, but the trailing dot is required to conform to ICANN standards.

A name server is a computer designated to translate domain names into IP addresses. These servers do most of the work in the DNS system. Since the total number of domain translations is too much for any one server, each server may redirect request to other name servers or delegate responsibility for a subset of subdomains they are responsible for.



———————

Audit Events(process roll up data (clone, fork, vfork, execve, exit_group) - Crowdstrike

———

Great article on **Gradle** - http://www.drdobbs.com/jvm/why-build-your-java-projects-with-gradle/240168608?pgno=2

Ant's first official version was released in 2000. Each element of work (a *target* in Ant's lingo) can be combined and reused. Multiple targets can be chained to combine single units of work into full workflows. For example, you might have one target for compiling Java source code and another one for creating a JAR file that packages the class files. Building a JAR file only makes sense if you first compiled the source code. In Ant, you make the JAR target depend on the compile target. Ant doesn't give any guidance on how to structure your project. Though it allows for maximum flexibility, Ant makes each build script unique and hard to understand. External libraries required by your project are usually checked into version control, because there is no automated mechanism to pull them from a central location. Early versions of Ant required a lot

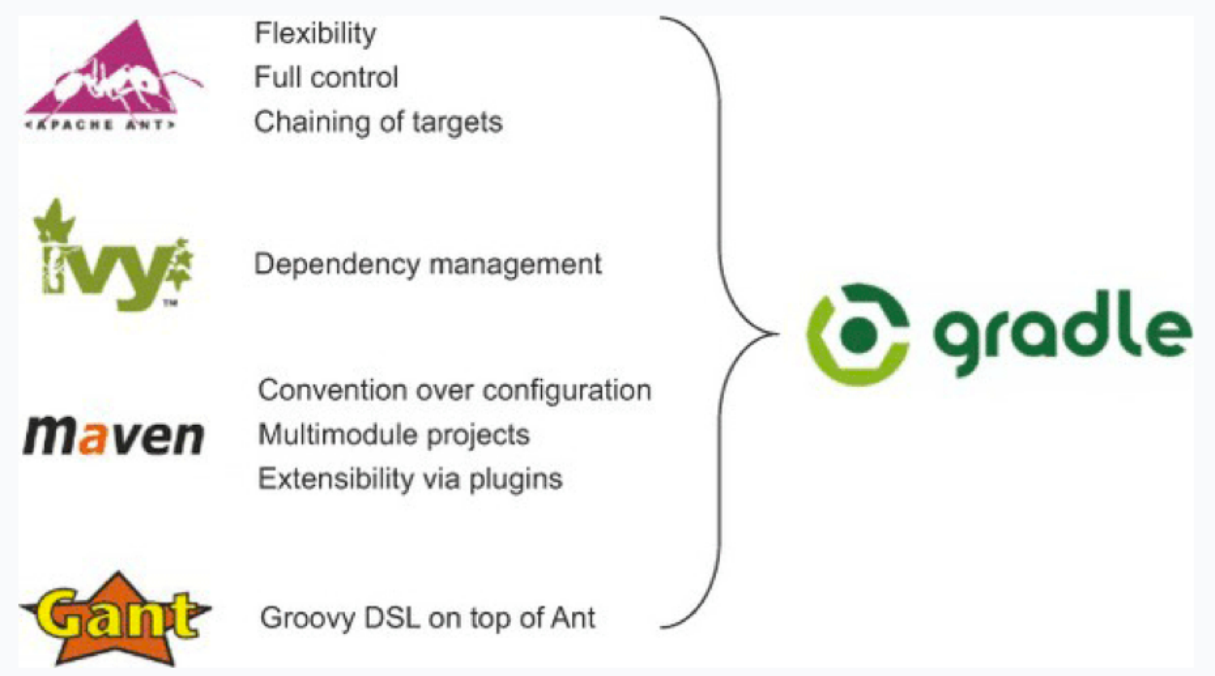
of discipline to avoid repetitive code. Its extension mechanism was simply too weak. As a result, the bad coding practice of copying and pasting code was the only viable option. To unify project layouts, enterprises needed to impose standards.

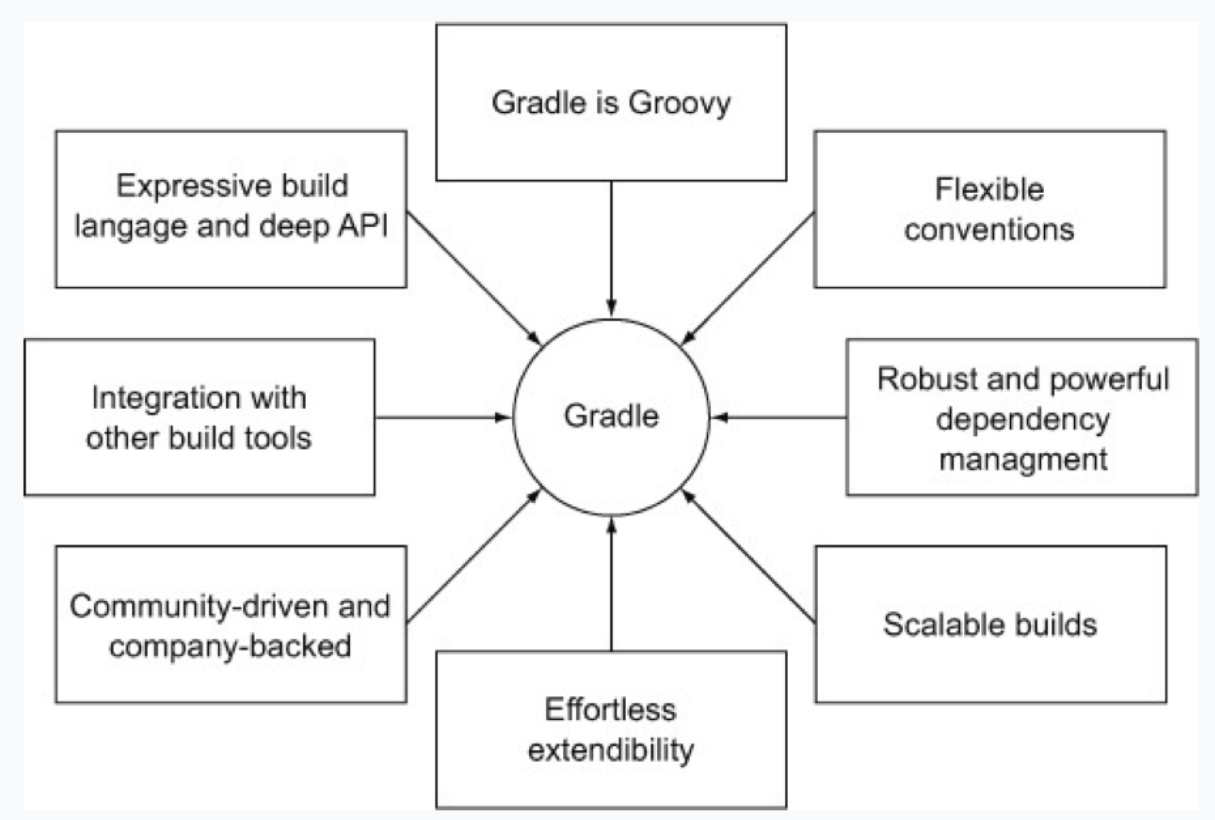
Maven 1, released in July 2004, tried to ease that process. It provided a standardized project and directory structure, as well as dependency management. Unfortunately, custom logic is hard to implement. If you want to break out of Maven's conventions, writing a plugin, called a *Mojo*, is usually the only solution. The name Mojo might imply a straightforward, easy, and sexy way to extend Maven; in reality, writing a plugin in Maven is cumbersome and overly complex.

These days a lot of people are looking for alternatives to established build tools. We see a shift from using XML to a more expressive and readable language to define builds. A build tool that carries on this idea is Gant, a DSL on top of Ant written in Groovy(domain-specific language (**DSL**) instead of the XML). Using Gant, users can now combine Groovy features with their existing knowledge of Ant without having to write XML. Even though it wasn't part of the core Maven project, a similar approach was proposed by the project Maven Polyglot that enables you to write your build definition logic, which is the project object model (POM) file, in Groovy, Ruby, Scala, or Clojure.

We're on the cusp of a new era of application development: polyglot programming. Many applications today incorporate multiple programming languages, each of which is best suited to implement a specific problem domain. It's not uncommon to face projects that use client-side languages like JavaScript that communicate with a mixed, multilingual back end like Java, Groovy, and Scala, which in turn makes calls to a C++ legacy application. It's all about the right tool for the job. Despite the benefits of combining multiple programming languages, your build tool needs to fluently support this infrastructure as well. JavaScript needs to be merged, minified, and zipped, and your server-side and legacy code needs to be compiled, packaged, and deployed.

Gradle fits right into that generation of build tools and satisfies many requirements of modern build tools (Figure 1). It provides an expressive DSL, a convention over configuration approach, and powerful dependency management. It makes the right move to abandon XML and introduce the dynamic language Groovy to define your build logic. Sounds compelling, doesn't it?





Gradle is Groovy

Prominent build tools like Ant and Maven define their build logic through XML. As we all know, XML is easy to read and write, but can become a maintenance nightmare if used in large

quantities. XML isn't very expressive. It makes it hard to define complex custom logic. Gradle takes a different approach. Under the hood, Gradle's DSL is written with Groovy providing syntactic sugar on top of Java. The result is a readable and expressive build language. All your scripts are written in Groovy as well. Being able to use a programming language to express your build needs is a major plus. You don't need to be a Groovy expert to get started. Because Groovy is written on top of Java, you can migrate gradually by trying out its language features. You could even write your custom logic in plain Java — Gradle couldn't care less. Groovy veterans will assure you that using Groovy instead of Java will boost your productivity significantly. A great reference guide is the book *Groovy in Action, Second Edition* by Dirk Koenig et al. (Manning, 2009).

———
visudoer — command gives all sudo level access to normal user

———
iptables - https://www.youtube.com/watch?v=XKfhOQWrUVw

SERVER_IP = “192.168.2.12”

iptables -I INPUT -i eth0 -p tcp -s 0/0 -d $SERVER_IP - - deport 21 -j DROP

-I stands for insert , -A for append
-i - interface eth0
-p - protocol
-s source ip
-d destination
- - dport - destination port
-j stands for target is to ACCEPT that incoming packet

If we wish to accept incoming packets only from local network then use below

```
NETWORK=“192.168.0.0/24”
iptables -I INPUT -i eth0 -p tcp -s $NETWORK -d
  $SERVER_IP --dport 22 -j ACCEPT
```

—Stateful firewall rule.. State defines if connection is stateful or not . State is stored in stateful table

Enable or allow ICMP ping incoming client request

```
iptables -I INPUT -i eth0 -p icmp –icmp-type 8 –s
  0/0 -d $SERVER_IP –m state –state NEW,
  ESTABLISHED, RELATED -j ACCEPT
```

```
iptables -I OUTPUT -i eth0 -p icmp –icmp-type 0 –s
  $SERVER_IP –d 0/0 –m state –state
  ESTABLISHED, RELATED -j ACCEPT
```

what is ICMP protocol - https://support.microsoft.com/en-us/kb/170292

To disable outgoing ICMP request , we can do it following two ways -

iptables -A OUTPUT -p icmp - - icmp-type echo-request -j DROP

or

iptables -A OUTPUT -p icmp - - icmp-type 8 -j DROP

StandAlone Firewall

```
# Remove any existing rules from all chains
iptables –flush

# Unlimited traffic on the loopback interface
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Set the default policy to drop
iptables --policy INPUT   DROP
iptables --policy OUTPUT  DROP
iptables --policy FORWARD DROP
```

StandAlone Firewall

```
SERVER_IP="192.168.0.3"
INTERNET="eth0"                 # Internet-connected interface
LOOPBACK_INTERFACE="lo"   # however your system names it
SUBNET_BASE="192.168.0/24"
LOOPBACK="127.0.0.0/8"          # reserved loopback address range
CLASS_A="10.0.0.0/8"            # class A private networks
CLASS_B="172.16.0.0/12"         # class B private networks
CLASS_C="192.168.0.0/16"        # class C private networks
CLASS_D_MULTICAST="224.0.0.0/4"     # class D multicast addresses
CLASS_E_RESERVED_NET="240.0.0.0/5"   # class E reserved addresses
```

As shown below, 3rd rule - if INTERNET (which is eth0) and our IP same then drop those as shown else it will cause denial of service attack
On rule 4 till 6 on CLASS_A,B,C are for our own private ip… so drop those as shown else another denial of service attack

Last rule , loopback IP should not be sending packets to our internet eth0 so if that comes then drop it as shown

StandAlone Firewall

```
iptables -A INPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# Source Address Spoofing and Other Bad Addresses

# Refuse spoofed packets pretending to be from
# the external interface's IP address
iptables -A INPUT  -i $INTERNET -s $IPADDR -j DROP

# Refuse packets claiming to be from a Class A private network
iptables -A INPUT  -i $INTERNET -s $CLASS_A -j DROP

# Refuse packets claiming to be from a Class B private network
iptables -A INPUT  -i $INTERNET -s $CLASS_B -j DROP

# Refuse packets claiming to be from a Class C private network
iptables -A INPUT  -i $INTERNET -s $CLASS_C -j DROP

# Refuse packets claiming to be from the loopback interface
iptables -A INPUT  -i $INTERNET -s $LOOPBACK -j DROP
```

Here is example how to set up firewall that has one host and 2 client network hosts. So our policies should be as follows -
So there are various protocols below
1. Standalone connection - using STATE option
2. SSH connection in network using $SERVER_IP option
3. Ping protocol - which is ICMP option (Internet controlled message protocol (ping))

1. Medial access control
2. IP
3. ICMP
4. TCP
5. UDP
6. HTTP
7. FTP

8. DNS
9. SSH

Some low level protocols are TCP, UDP, IP, and ICMP. Some familiar examples of application layer protocols, built on these lower protocols, are HTTP (for accessing web content), SSH, TLS/SSL, and FTP

```
Policies
Default INPUT = DROP
Default OUTPUT = ACCEPT
Default FORWARD = DROP

Stateful firewall
Server Ubuntu IP=192.168.1.130
Client Mint9 IP = 192.168.1.128 Allow FTP access, Allow ICMP request
Client Ubuntu10 IP = 192.168.1.129 Allow SSH access (later), Reject ICMP request
```

```
#!/bin/sh

SERVER_IP="192.168.1.130"
MINT="192.168.1.128"
UBUNTU="192.168.1.129"

iptables --flush
iptables --policy INPUT DROP
iptables --policy OUTPUT ACCEPT
iptables --policy FORWARD DROP

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISH,RELATED -j ACCEPT
```

```
iptables -A INPUT -i eth0 -p tcp -s $MINT -d $SERVER_IP --dport 21 -j ACCEPT

#iptables -A INPUT -i eth0 -p tcp -s $UBUNTU -d $SERVER_IP --dport 22 -j ACCEPT

iptables -A INPUT -i eth0 -p icmp --icmp-type 8 -s $UBUNTU -d $SERVER_IP -j ACCE
PT

iptables -A INPUT -i eth0 -p icmp --icmp-type 8 -s $MINT -d $SERVER_IP -j REJECT
```

Elastic Beanstalk FAQ - https://aws.amazon.com/elasticbeanstalk/faqs/

Continuous Integration/ Continuous Delivery(CI/CD)