**Ansible:**

**Source:**

**https://www.youtube.com/watch?v=XJpN8qpxWbA**

<https://www.youtube.com/watch?v=dCQpaTTTv98>

<https://www.youtube.com/watch?v=icR-df2Olm8>

**https://www.youtube.com/watch?v=eEvqke6PW2E**

1. It is used for configuration management.
2. Main advantage of configuration management is to base line all the servers, ie., separting webserver/app servers/DB servers, so that we can check only those servers when we get the related issue.
3. The other advantage of configuration management is to easily scale up and scale down the hardware. The existing configuration play books will help us immediately.
4. All the servers can configure in single go. So it reduces the effort and there is no wait for developers for configuration and deployment and testing.
5. With configuration management, all the older versions configurations of our applications, so if new version is not good, with zero downtime, we can revert it.
6. Ansible is a configuration management tool to do these things and it is push mechanism.

**Features:**

1. Easy to install and configure.
2. Very easy to learn
3. Agentless, no need of installing agent. Install ansible in server, do SSH connection with nodes and push the playbooks.
4. Very powerful, having more than 750 inbuilt modules.
5. Modules are extensible.

**Ansible case study - NASA:**

1. Ansible tower organizes, Credential Management, Users, Job Assignment, Inventory Management.

**Ansible Architecture:**

1. It contains an automation engine in which, Inventory(list of all IPs of host machines), playbooks (Modules, APIs(to support command line arguments like python), Plugns(used to execute modules, action plugins and connection plugins))are present.
2. CMDB(configuration management database), stores all configuration details. Also saves all the configuration details of all our cloud machines.
3. Hosts

**Writing a playbook :**

**Structure:**

* Start with **3 dashes**.
* List of **hosts** where do you want to run this playbook.
* **Variables** section
* **Tasks** in top to bottom order like install softwares.
* **Handlers** are also Tasks but it needs triggers to execute. Triggers will be writing inside task. Eg., notify on installation of software.

**How to use Ansible:**

1. Used for deployment, configuration management, security, orchestration, provisioning.

* Orchestration means, defining the order like, machines to be removed from LB before upgrade and installation of LAMP stack before deploying PHP and HTML files etc.
* Ansible is useful in deployment
* Ansible is useful for security by configuring firewalls in playbooks.

**Hands on with Ansible:**

* Play books contain plays, plays contains tasks, tasks contain modules.
* Example modules:

1. Apt -> installation
2. Copy -> copy files from controller to nodes
3. Fetch -> copy files from nodes to controller
4. File -> link files, directories, change mode, change owner etc
5. Service -> start, stop, restart services
6. We also have AD-HOC commands instead of writing whole play book as follows.

ansible all -s -n shell -a ‘uptime’

ansible all -s -n shell -a ‘mount’

ansible all -s -n shell -a ‘cat /etc/redhat-release’

**Steps to work with Ansible:**

1. Install ansible using #pip install ansible.
2. Check the version using #ansible - - version
3. Provide ssh key of controller in node. Follow the below steps.

* #ssh-keygen
* In the target machine edit below line in vi /etc/ssh/sshd\_config file.

# vi /etc/ssh/sshd\_config

#PermitRootLogin yes -> this has to be uncommented.

#service sshd restart

* Copy ssh key using the command #ssh-copy-id -i root@<ip-address>

Or else

* #vi /etc/.ssh/id\_rsa.pub
* Copy this and paste in target machine’s /etc/.ssh/authorization\_keys file.

1. Create the ansible folder and add our nodes in hosts file.

#mkdir -p etc/ansible

#vi hosts

[test-nodes]

<ip-address of node1>

<ip-address of node2>

**Wq!**

1. Check the connectivity using the below command.

**#ansible -m ping ‘test-nodes’**

1. **Write a play book now in an yml file and test it in yamllint.com and run it using below command.**

**#ansible-playbook <play book name>**

1. Various commands and plugins to run commands directly in nodes without playbooks as follows.

#ansible <hosts-name> -m **command** -a ‘df -h’

#ansible <hosts-name> -m **service** -a “name=httpd state=started”

#ansible <hosts-name> -m **command** -a “/sbin/reboot -t now”

#ansible <hosts-name/node IP> -m **ping**

#ansible <hosts-name/node IP> -m **copy** -a “src=/etc/ansible/apache.yml dest=/tmp”

#ansible <hosts-name/node IP> -m **yum** -a “name=tomcat7 state=absent”

**<<Repeat>>**

**13/10/2018:**

1. To know the host name #hostname
2. To know the fully qualified domain #hostname -f -> domain is a group of machines.
3. #ifconfig ->
4. Private IP is an IP which is specific to a specific data center, it won’t be exposed to internet. The public IP is the IP address of the router which will be connected to the internet.

Internet -> public ip -> firewall -> provides the private ips which can be accessed only in the specific location. If we want to connect to the private IP, we need to go via public IP and the public IP is dynamic IP.

Public IPs are limited and costly.

Loopback IP 🡪 is an IP that specifies local computer. Ranges from 127.0.0.1 to 127.255.255.255 on IPv4 and ::1 on IPv6.

DHCP -> Dynamic Host Control protocol.

1. #df -h
2. #w -> how many users logged into the system.
3. #history -> to know all the commands used till now
4. #cat /etc/passwd -> provides all the users available in the machine.
5. #date
6. #cal
7. #uptime

Load average present in uptime tells us the load avg of last 5, 10 and 15mins

1. #top -> windows task manager

Zombie -> for a main process, some child processes will be started to serve them. In any case the parent process stops, the child processes won’t stop and these will be zombie processes, we need to kill them.

Swap memory is the virtual memory from hard disk when all our RMA is completely used to avoid the machine in hang state. Swap will reduses the performance.

Buffer is the memory to interact with I/O devices because CPU is much faster than I/O devices. To maintain the difference CPU maintains buffer.

Cached memory is used to keep the output of a previous command to serve other users if they use same command.

1. #ps -ef | grep defunct -> to know the zombie processes.
2. #ps -e -> to know all the processes
3. #ps -ef -> all the processes with associated folders.
4. #touch abc.txt -> create a file
5. #mkdir -p
6. #more abc.txt -> display the content of the file page wise.
7. #yum repolist -> repositories available in the machine
8. #yum list -> list down all the softwares available in the above repositories
9. #yum list | grep httpd -> to see the s/ws related to httpd.
10. #find / -name abc.txt -> to find a file with name abc.txt in the root(/) directory
11. #chown ec2-user:ec2-user abc.txt -> user:group
12. The **total** details comes on ls -l command defines the total number of blocks the folder occupied on hard disk.
13. #uname -> OS type
14. #uname -a -> more details about the OS
15. #traceroute google.co.in -> will give the all route IPs from our IP to destination machine. If somewhere breaks, it will show \*\*\*. This is very useful in debugging when a website is not working.

**Go through linux cheat sheet and admin cheat sheets.**

**Installations:**

1. Binary installation:

* To make the binaries to be run we need to build this first ie., compilation.
* Download the binaries from apache website(source code)-> download GCC because httpd is written in C/C++ -> prepare binaries using make command -> Install it.

**Virtual hosting – deploying more applications on same web server:**

1. Name based -> based on DNS name
2. IP based ->

DNS1

DI website1

Same IP Address

For both DNSs

DNS2 Website2

**14/10/2018:**

1. Google -> Download httpd for linux -> select proper version -> copy link
2. In linux machine

#cd /home/ec2-user

#wget < <https://archive.apache.org/dist/httpd/httpd-2.2.3.tar.gz>> -> download file

# tar -xzvf httpd-2.2.3.tar.gz -> z is for gz format.(extraction)

# cd httpd-2.2.3

1. We nee gcc as this source code written in C/C++ and for compatibility with pearl we need pcre(pearl compatible regular expression)
2. Also we need openssl to create certificates.

# yum install make gcc pcre-devel openssl-devel

1. ./configure will check whether required files are present or not for compilation. It also create a **Makefile** which will be having instructions to compile.
2. **make** will compile and create binaries.
3. **make install** will install the binaries in a folder location /usr/local/apache2).

#./configure

#make

#make install

Bin -> tomcat engine

Conf -> config file, where we can change apache

Modules -> contains readily available modules like mode \_sl and mod\_jk

Logs -> access logs and error logs.

1. Start apache using

#cd bin

# ./apachectl start

#curl ifconfig.co -> to know the public ip. Paste this in browser to check.

* We can install more apaches on same machine in the separate location.
* To start the second one you can either change port number or we can request for one more IP for the same machine and change this IP address instead of changing port number.

**Apache – Multi Process Module (MPM):**

**Pre-fork MPM**

1. One apache parent process having 8 child processes.
2. Each child serves one request. No thread model
3. #httpd -V shows which MPM our apache using currently.

**Worker-MPM**

1. Thread based. For connecting server there will be connection created which is called as thread.
2. 4 child processes present in this MPM and each child can handle multiple threads.

* When our application is less wait app and getting server is too busy error, we need to use Worker-MPM. Because so many requests to apache but service processing time is very low as it is a low weight application. CPU and memory are under utilized and still getting Server is too busy error, we should go for Worker-MPM.

Next class:

Deploy more than one application on one apache.

SSL certificate

Mod\_jk

* How to check which process is running on port 80?
* # netstat -plan | grep 80

**20/10/2018: (Repeat)**

**Deploying two applications in same httpd:**

* **One option is** Binding IP addresses -> get one more IP address to the same machine and bind this in our second Apache. We can request our OS guys.
* **The other one is using named virtual hosts.**
* Add two virtual hosts, only server and root has to be changed.
* Enable “NameVirtualHost \*:80”
* Instead of being “aws.com” going to official site, we need to overwrite it in /etc/hosts, so that it reaches our website.
* Install #yum install elinks for testing
* Test it with #elinks aws.com

**Httpd.conf looks like below:**

NameVirtualHost \*:80

#

# NOTE: NameVirtualHost cannot be used without a port specifier

# (e.g. :80) if mod\_ssl is being used, due to the nature of the

# SSL protocol.

# VirtualHost example:

# Almost any Apache directive may go into a VirtualHost container.

# The first VirtualHost section is used for requests without a known

# server name.

#

#<VirtualHost \*:80>

# ServerAdmin webmaster@dummy-host.example.com

# DocumentRoot /www/docs/dummy-host.example.com

# ServerName dummy-host.example.com

# ErrorLog logs/dummy-host.example.com-error\_log

# CustomLog logs/dummy-host.example.com-access\_log common

#</VirtualHost>

<VirtualHost \*:80>

ServerAdmin webmaster@dummy-host.example.com

DocumentRoot /www/docs/aws

ServerName aws.com

ErrorLog logs/dummy-host.example.com-error\_log

CustomLog logs/dummy-host.example.com-access\_log common

</VirtualHost>

<VirtualHost \*:80>

ServerAdmin webmaster@dummy-host.example.com

DocumentRoot /www/docs/aws2

ServerName aws2.com

ErrorLog logs/dummy-host.example.com-error\_log

CustomLog logs/dummy-host.example.com-access\_log common

</VirtualHost>

**SSL(Secure Sockets Layer):**

* Instead of sending plain text over the internet (http-80), we can encrypt the data(https-443).
* This encryption should be authorized by browser, otherwise we will get, your site is not secured.
* We can generate our own SSL using **Openssl.**

OpenSSL

RSA alogorithm

1024

Creation of certificate: ???

* Conf**# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout test.key -out test.crt**
* Under conf folder you can find two extra files(test.crt, test.key) created.
* Configure virtual host with port number 443 inside http.conf as below.
* Add one more Name based virtual host with port number 443.

<VirtualHost \*:443>

DocumentRoot /var/www/html/

SSLEngine on

SSLCertificateFile /etc/httpd/conf/test.crt

SSLCertificateKeyFile /etc/httpd/conf/test.key

</VirtualHost>

* For this we also need openssl module under /etc/httpd/modules
* Get it from amazon repository using #yum install mod\_ssl
* Restart httpd and test it via browser.
* https://<ip address>
* Even http should be redirected to https. For this we need to follow below.
* Add the redirection virtual host in httpd.conf at top of all virtual hosts.
* Configure two SSL virtual hosts. Add below line in httpd.conf

NameVirtualHost \*:443

**Redirecting http to https:**

REDIRECTION

=====================

ONE TYPE OF REDIRECTION (IP ADDRESS REDIRECTION)

<VirtualHost \*:80>

ServerName example.com

Redirect / https://35.154.222.121

</VirtualHost>

SECOND TYPE OF REDIRECTION (MOD\_REWRITE)

<VirtualHost \*:80>

RewriteEngine On

RewriteCond %{HTTPS} off

RewriteRule (.\*) https://%{SERVER\_NAME}/$1 [R,L]

</VirtualHost>



**Application Servers:**

* Web servers hosts only static content but useful in creation of connection b/n internet and app server. To host dynamic content from DB, we should go for app servers.

Static Content(web server)

Dynamic(app server)

DB

* Install tomcat7-> start it-> install sample application

#yum install tomcat7

#service tomcat7 start

# yum install -y tomcat-webapps

* It is developed on Java so java should be there in our machine. Java – version.
* Works on port number 8080, http protocol
* Also works on AJP(8009) and https(8443)

Internet -> web server -> Firewall -> App server -> DB

* Firewall will accept the traffic from dedicated webserver only.

Firewall(opens only for dedicated webserver requests.)

APP server

Data base

Web server

**Connecting from web server to app server:**

* **It can be done by two ways.**
* **1. Proxy pass 2. Mod\_JK**

1. **Proxy pass:**

**-> easy, module inbuilt. Cons -> one-one, straight forward.**

**-> Enter the below two lines inside httpd.conf to connect through proxy pass.**

ProxyPass /examples <http://172.31.45.203:8080/exmaples/>

ProxyPassReverse <http://172.31.45.203:8080/exmaples/> /examples

ProxyPass /sample <http://172.31.45.203:8080/sample/>

ProxyPassReverse <http://172.31.45.203:8080/sample/> /sample

**21/10/2018(Repeat):**

1. **Mod\_jk:**

|  |  |
| --- | --- |
| **Proxy Pass** | **Mod\_JK** |
| One to one | One to many |
| Easy to configure | Not easy |
| Default module available in httpd | Need to download and configure |
| Supports http/s | Supports AJP13 |
| Works on port numbers 80/443 | Works on port number 8009 |

**Google.co.in**

Nearest DNS

LB

Mod\_JK

Webserver3

Webserver2

Webserver1

Mod\_JK

App server3

App server2

App server1

1. **Take two EC2 servers**
2. **Install web server on one server**
3. **Install App servers on both**
4. **Download, load, configure mod\_jk inside webserver.**

**Mod\_JK configurations:**

1. **#cd /etc/httpd/modules**
2. **Download mod\_jk for specific version of httpd(#httpd -v).**
3. **#wget** <https://archive.apache.org/dist/tomcat/tomcat-connectors/jk/binaries/linux/jk-1.2.31/x86_64/mod_jk-1.2.31-httpd-2.2.x.so>
4. **#mv mod\_jk-1.2.31-httpd-2.2.x.so mod\_jk.so**
5. **Provide execute permissions for it.**
6. **#vi /etc/httpd/conf/httpd.conf**

* **Load the module(#LoadModule jk\_module modules/mod\_jk.so)**
* **Setup workers properties**
* **Log level**
* **Log format**
* **Mount the app**

**===================================**

**/etc/httpd/conf/httpd.conf**

**===================================**

**LoadModule jk\_module modules/mod\_jk.so**

**JkWorkersFile /etc/httpd/conf/workers.properties**

**JkShmFile /etc/httpd/conf/mod\_jk.shm**

**JkLogFile /etc/httpd/conf/mod\_jk.log**

**JkLogLevel info**

**JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "**

**JkMount /examples/\* worker1**

**===================================**

**/etc/httpd/conf/workers.properties**

**===================================**

**worker.list=worker1**

**worker.worker1.type=ajp13**

**worker.worker1.host=18.224.170.125 (IP address of app server installed machine)**

**worker.worker1.port=8009**

* **To change the console name to differentiate**

**#export PS1='appserver#'**

1. Test this by accessing the url <http://13.58.216.45/examples/> (this is the IP of webserver not the app server)

**Load Balancer – APP server:**

* **Install tomcat7 and tomcat7-webapps in another machine and add below lines in worker.properties.**

**worker.list=worker1, worker2, loadbalancer**

**worker.worker1.type=ajp13**

**worker.worker1.host=18.224.170.125**

**worker.worker1.port=8009**

**worker.worker2.type=ajp13**

**worker.worker2.host=localhost**

**worker.worker2.port=8009**

**worker.loadbalancer.type=lb**

**worker.loadbalancer.balance\_workers=worker1,worker2**

* **Also mount loadbalancer instead of worker1 in httpd.conf.**

**JkMount /examples/\* loadbalancer**

* **Stop tomcat7 in one of the servers and hit the url** <http://13.58.216.45/examples/>**. It should work. Do the same by stopping the other tomcat7 and starting the current one. It should work.**

**Important errors:**

This site can’t be reached -> when web server itself is down

Service Temporarily Unavailable -> webserver is runnnig but all app servers are down.

Not Found -> web server is running, but the application is not mount at loadbalancer. Need to mount this application in httpd.conf

# JkMount /sample/\* loadbalancer

**Load Balancer – Web server:**

# Install httpd in other machine and do all mod\_jk set up here.

# Go to load balance in AWS site.

# Create classic load balancer.

# While creation select the security group which is already assigned to our machines which you want to put in LB.

# Also select those two machines.

# Test it using the load balancer DNS name (webLB-882584192.us-east-2.elb.amazonaws.com).

# Stop services in various combinations and test.

# Note: To avoid installing all the softwares, you can simply select one machine which is already having all the softwares and select Actions -> Image -> create image.

**27/10/2018(weekend):**

1. Health string in AWS will check the index.html under root directory. We can also change this in Health Check tab of LB.

**ASG(Auto Scaling Group):**

* Spinning of new machines can be automated based on conditions.
* Eg., the existing machine CPU reaches more than 70% or memory reaches 80% -> spin up a new machine to handle the load.
* Go to AWS page -> left side under Auto Scaling -> Launch Configurations
* Before this first we need to create Auto Scaling Group by selecting an IAM image.
* Here you can create ASG. In b/n you need to select all available subnets.
* Provide how many initial machines you need.
* Once it is ready create policies according to your requirement.

**EBS(Elastic Block Storage):**

There are 3 types of storages capacity from low to high as below, but cost is reverse:

1. S3 -> simple storage solution -> small storage
2. EBS -> Elastic Block Storage -> moderate storage
3. Glacier -> too high storage

* Glacier is cheaper, but IO operations are very slow, so it will be used mainly for storage.
* EBS and Glacier are region specific but S3 is global.
* Ie., if we have data in EBS in Mumbai, we can’t attach this data to the server in Hyderabad.
* S3 data can be accessed via internet using http protocol.
* What if your up and running machine is not accessible via putty, we should detach the volume and attach this to another machine(spin up new machine) and terminate this machine.(IQ)
* We need to replicate our data in various location to protect our data.

**Create a new volume and attach to our linux machine also mount it using fdisk:**

* Create a new volume of 5 GB in AWS in the same zone of the machine and click on actions and attach it to our machine.
* Inside the linux machine please do these.

#df -h -> you can’t see your 5 GB attached data

#fdisk l -> fdisk is a module used to partition the disk. You can see your extra 5GB data in MB’s with name /dev/xvdf(may differ)

#fdisk /dev/xvdf

#n -> press “n” for new partition.

* We can create maximum of 4 partitions internally. (IQ)

#press enter until you reaches one more help.

# w -> Press “w” for write/save this.

#df -h -> you can’t see the 5 GB as you haven’t mounted.

#fdisk -l -> you can see the partition

# mkfs.ext3 /dev/xvdf1 -> to make file system for our 5GB volume

# mkdir /data -> create a new directory to mount our volume

# mount /dev/xvdf1 /data -> mount our new volume to this new directory

# df -h -> now you can see your 5GB data.

**Fiddler is used to check where the connection breaks. So that we can inform LB team in case the step where the connection failed matches with the LB IP.**

**28/11/2018(weekend):**

**Roadmap to become a DevOps engineer:**

* For CI/CD, continuous testing and continuous monitoring, number of tools needed. Having knowledge on these is DevOps work.
* Linux admin including scripting
* Scripting knowledge (bash, power shell, python)
* Configuration management tools(Ansieble, Chef, puppet)
* GIT/GIT HUB
* JENKINS
* Monitoring
* Cloud providers

**Nagios:**

* Active check and passive check->If Nagios is having its own plugin to check it is active check but if some other tool sending the status and Nagios is just showing on its dashboard then it is a passive check.
* Nagios knows what is down and what is unavailable.

1. Destination not reachable -> when within the TTL(time to live), if source machine unable to find the destination IP is Destination not reachable
2. Host down -> when source finds out the destination IP in the n/w with in TTL but it didn’t get the response within TTL, then host is down.
3. Timedout -> within TTL, source didn’t get any response it is timedout. It may be b/c of n/w traffic busy.

**Features ->** can configure to whom we need to send alerts. On which services we need to monitor.

**Network viewpoint concept ->** for distributed configuration Nagios Fusion is useful.

**Configuration files -> check all the configuration files and work???**

* **Object.cache vs objects.precache**

**Check various commands by running in the command prompt???**

**Install SMTP package and configure mail id in contacts and check for alerts???**

**#yum install mailx**

**Localhost -> add one more service, change the command name as you expected.**

**Under commands config configure this check\_service command. Take the help of /etc/lib64/plugins/httpd plugin help page.**

**Check the log files**

**Flagpping/Flap detection ->** Nagios will check for last 22 times if there are any fluctuation and not sending mails because it becomes actual state before checking 5 minimum times and send alerts.

**Take one more machine, configure one more host under localhost.cfg file also under services, add this host name as well and check. Or else you can create a group for both localhost and this new server and configure group directly???**

**Set up Nagios for windows:**

* **Go to Nagios.org -> install it -> plugins ……. ???**

**Nagios exchange for plugins. Take a plugin, configure it, set it up in our Nagios for servers. ???**

**Number of apache connections -> create a new plugin under plugins folder, give execute permission, give this command in localhost to check. Configure this command in commands config file.**

**04/11/2018:**

**Configuration Management Tools:**

* Platform independent configuration, easy English, extra facilities than the shell script or power shell.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Salt** | **Puppet** | **Chef** | **Ansible** |
| **OS** | All | All | All | All |
| **Developed using** | Python(P-DSL) | Ruby(P-DSL) | Ruby(Chef-DSL) and Erlang | Python(YML) |
| **Mechanism** | PUSH | PULL | PULL | PUSH |
| **Agent** | Agentless | Agent Based | Agent Based | Agentless |
| **Market** | Deprecated | Going down | Stable | Emerging |

**CHEF(WE-Repeat):**

* Official version release on Jan 2009.
* OPSCode community manages CHEF.
* Opensource and Enterprise versions are available.
* CHEF-Framework includes work station, server and nodes (VM node or physical node or Cloud instance)

**Workstation:**

* Under **workstation**, we need to

1. install ChefDK,
2. download starter kit
3. chef-repo(comes along with starter kit)
4. Bootstraping -> installing chef client in a node remotely and registering with chef server.

* Tools we used under CHEF are –

1. Knife
2. Test Kitchen
3. FoodCritic

* Work station can be on windows or Linux machine. It can be a separate machine or can be included in chef server or in a laptop.

**Chef-server:**

* 3 types of servers are there.

1. Opensource -> no support, we need to install everything on our machine
2. Hosted chef server -> chef people will do a set up for us and 5 nodes support is free. It can support up to 100 nodes with license cost.
3. Enterprise CS -> unlimited number of nodes

**Node:**

* We need to

1. Install chef client
2. Register this node with chef server
3. Communication b/n server and node will be by using SSL.

**Chef Resources:**

1. Service - > used to start a service in the client
2. Package -> used to install/remove/upgrade of a s/w in client
3. File -> can be used for creation/deletion/edition of files
4. Template -> can be used when u want to supply some attributes

For every resource, you will be having attributes.

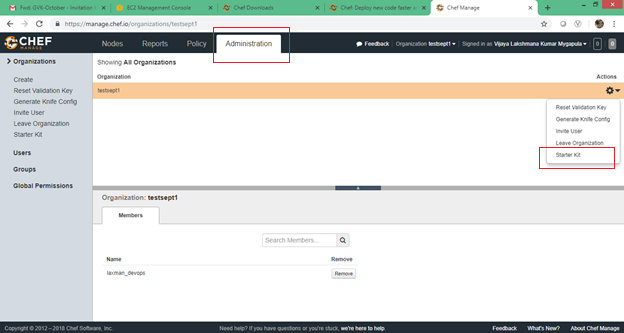
**Recipe vs Cookbook:**

* Collection of all resources is called recipe and collection of all recipes is called as Cookbook.
* The extension of the recipe is .rb.
* Under a cookbook the below folders will be created.

1. Template
2. Recipe -> contains default.rb
3. File

**Hands on:**

* Create an AWS instance
* Google -> download chefDK
* Go to /home/ec2-user -> # wget <https://packages.chef.io/files/stable/chefdk/3.4.38/el/7/chefdk-3.4.38-1.el7.x86_64.rpm>
* # rpm -Uvh chefdk-3.4.38-1.el7.x86\_64.rpm -> installs the checkDK rpm file.
* Set up a chef server now. Company is already giving a server from chef as first option in chef.io website.
* Open chef.io -> right top -> log in -> manage console/server
* Create a login and enter inside.
* Account details: laxman\_devops, laxman123
* Create an organization with unique name.
* Create different organizations based on either location or based on environment. It would be basically useful to apply rules for a set of machines grouped under one organization.
* Download starter kit as shown.



**To configure FileZilla to use a private key for SFTP access, follow these steps:**

1. Transfer the private key **file** to the client computer.
2. Download **FileZilla** and install it on the client computer.
3. Start **FileZilla**.
4. On the Edit menu, click **Settings**. ...
5. Under Select Page, click **Connection**, and then click SFTP. ...
6. Click Add keyfile.

* Copy this startkit to chef server using winscp or filezilla.
* # unzip chef-starter.zip -> unzip it.
* # cd /chef-repo/.chef -> go inside of this folder
* # vi chef.rb -> you can see different settings like log settings….
* # cd ..
* # knife ssl check -> to check SSL validity and connectivity is there with chef server or not from chef work station.
* Create a new EC2 machine for chef node.
* This node should be listed on chef server.
* Create an SSH connection from chef workstation to chef client
* Once connection is established, change to sudo user and go to chef-repo folder.
* Now do the bootstrap from workstation to client.

**10/11/2018:**

**Class miss….**

**11/11/2018:**

* Google -> chef resources -> open first link
* Bootstrap the client from work station from ec2-user chef-repo folder

# knife bootstrap 3.16.28.164 --ssh-user ec2-user --sudo --identity-file /home/ec2-user/chef-starter/chef-repo/.chef/laxman\_devops.pem --node-name cheffnode

**Create individual recipes and test in work station:**

* Go to chef-repo file.
* Create a test.rb file with some resources with actions.
* # chef-client - - local-mode test.rb -> to run this in work station for testing.
* You can also use # chef-cliet -z test.rb

**Create cookbooks and export to chef server and run in chef client:**

* Go to cookbooks folder.
* # chef generate cookbook laxmancookbook
* # cd /cookbooks/laxmancookbook/recipes
* # vi default.rb
* Go to cookbooks
* # knife upload laxmancookbook
* # knife node list -> to know the nodes list
* Open chef server, add this cookbook into run list.
* Open chef client and run below command.
* # chef-client

**Default.rb:**

#

# Cookbook:: laxmancookbook

# Recipe:: default

#

# Copyright:: 2018, The Authors, All Rights Reserved.

package 'httpd' do

action :install

end

file '/var/www/html/index.html' do

content 'This is my own cookbook to install httpd'

end

service 'httpd' do

action :restart

end

**Creation of templates:**

# chef generate template index.html

# cd templates/

# vi index.html.erb

**How to run cook book directly in local machine:**

chef-client -z --runlist recipe[<cookbookName>]

**Exercise1:**

* Install two jdk’s two tomcats, two httpd on different ports.
* Set up proxy pass
* Set up virtual hosts

**Exercise2:**

* Write a cookbook to install source apache install.
* Replace the lines using unix commands for jdk version and proxy passes. Use **sed** and **cat** commands to do this.

**$ sed -I .bak / ->**

**To include a new rb file other than default.rb:**

include\_recipe ‘<other rb file in default.rb file>’

node[:ipaddress] -> to get the ip address of a machine.

**BERKS:**

* Under cookbook you can find a metadaga.rb. Under which you can add depends tag.

**Vi metadata.rb**

name ‘<author\_name>’

version ‘<version\_number>’ -> provide latest version not to override the previous one

depends ‘<cookbookName>’, ‘~>5.0’ -> download the cookbook from super market having version greater than 5.0

**ATOM:**

* Open source useful in providing syntax error while writing. Advanced editor.
* yum install atom

**24/11/2018:**

**PUPPET**

* Developed using Ruby.
* Puppet master, node/puppet. Communication happens by SSL identification.
* Works on pull mechanism.
* Idempotency -> if the s/w or configuration is already there, it won’t install again.
* puppet.conf -> provide the configuration to which server this node has to connect.
* fileserve.conf -> under puppet master saves the configurations
* autosign.conf -> if u change this, any client request is automatically signed off. Dangerous.
* **Facter ->** gives the clients details.
* **Module ->** Collection of manifests.

**Hands on:**

* Google -> puppet.com
* Site.pp is a default singe file in which you can write the code without creating the modules. It is a run list. But it is recommended to create modules and write code in init.pp file. List this in site.pp.