**Day 1 (22.12.16)**

**GIT**

under C Drive, create gitworkspcae folder

**Configuring username and email-id in git:**

$ git config user.name --global "Vamsee1988"

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

Username and email is used by the git tool when we perform the commit.

cd into that folder

$ git clone <url of the repository>

Clone will completely copy the repository

**Exercise:**

Modify the files in local rep and pushing those changes to the remote rep

**steps:**

1)Create a new file like welcome.txt

2)Enter some details into the file above created and save it.

cd into local rep and devops3342

$vi welcome.txt

save it

$git status

$git add welcome.txt

$git status

**Committing changes in the staging area into local rep:**

$ git commit -m 'some meaningful message'

after this the changes are committed in the local rep

**Committing the changes in the local rep to remote rep:**

$git push origin master

->It will ask for username and password for the first time.

--global tells git use this email and name for all git rep's in the current computer.

Push command is rejected sometimes.

if local rep doesn't have the changes present in remote rep then our push operation is rejected.

$ git pull origin master

**This command does two things:**

1)Pull latest changes from the remote into local.

2)Merge remote changes with local changes.

**Day 2 (23.12.16)**

**Exercise:**

1)Create repo devops-prac in github

2)Include readme.md file while creating repo

3)Clone devops-prac rep into local machine

4)create a new text file mydetails.txt with some data

5)commit the modified rep file into local rep

6)push the changes to remote rep

---------------------------------------------------------

**Dealing with branches in git:**

When we create a new repo, by default we get master branch.

-->If there is any new functionality or a bug fix,it must happen on a separate branch.

We should not directly work on master branch.

**Creating a branch from github GUI console:**

branch-->enter any branch name <for searching a branch /for creating a new branch>-->click enter

**Creating a branch in local and push it to remote:**

$ git branch <branch name>

$ git branch (to list out all the branches in the repository)

**Switching a branch:**

$git checkout <branch name>

**Pushing a local branch to remote:**

$git push origin <branch name>

**Merging changes in devops-prctc1 into master:**

-->If we make any changes in a branch, then after committing it, we need to merge.

1)First we need to switch the branch to master.

2)$ git merge devops-prctc1

3)$ git push origin master

**Deleting a branch:**

$git branch -d <branch name>

$git push origin -d devops-prctc1

**Force delete:**

$ git branch -D <branch name>

We cannot delete a branch if it has new changes which is committed but not merged.

Still we want to delete the branch forcefully ,then we use -D instead of -d.

**Day 3 (26.12.16)**

**git stash**

It takes the changes in working and staging area, places them in temporary memory.nges in

For example, we are working on branch1, there are changes in working and staging area, suddenly we got a mail to fix a issue in production. we want to create a new branch and fix a issue in that. In working and staging area branch those of our core but

------------------------------

To solve this problem we can use stash

$ git stash save ---- it saves the changes in working and staging area

$ git stash clear ---- to removing changes from stash

**tags**

tags are used for releasing a particular version of software to the customers.

**Difference b/w a tag and a branch**

**Pull requests**

$ git init ---- we can convert the current folder in git local repository.

$ pwd

$ ..

$mkdir cloud

$ git init

**Build Tool**

We need to install

1. VirtualBox - to create a virtual machine - which acts as a Hypervisor

2. Vagrant -- Vagrant helps in simplifying the creation of virtual machine

3. Putty - it a software used on windows, this tool is helpful to login to remote unix/linux base servers

4. ssh

create a folder C:vms

github.com/javahome/devops-7am

**Starting a virtual machine using Vagrant**

--------------------------------------------------------------------------------------------

Vagrant.configure("2") do |config|

config.vm.define "server1" do |server1|

server1.vm.box = "bento/centos-7.2"

server1.vm.network "private\_network", ip: "192.168.50.10"

end

config.vm.define "server2" do |server2|

server2.vm.box = "bento/centos-7.2"

server2.vm.network "private\_network", ip: "192.168.50.11"

end

end

--------------------------------------------------------------------------------------------

$vagrant status -

$vagrant up server1 - to start

$vagrant halt server1 - to stop

$vagrant destroy server1

login into virtual machine

--------------------------------------

**Day 4 (28.12.16)**

**Maven**

Maven is a build and dependency management tool.

**What happens when we build a project?**

1. Compiles the code i.e., takes the source code and converts it into a binary format or a machine understandable format.

2. Execute JUnit test cases return by the developers as part of this JUnit is a framework for automating test cases.

3. JUnit finds logic failures.

4. It creates an artifact. Artifact is nothing but jar, war, ear

JAR- Java Archive (collection of .class files)

WAR- Web application Archive (collection of jar files, html, css, javascript, php)

EAR- Enterprise Archive (related to EJB)

**Maven Dependency Management**

Maven is helpful in downloading required dependencies (jar) for a particular project.

**Installing and configuring maven on CentOS**

JDK is the prerequisite for using maven

**To install OpenJDK 7 JDK using yum, run this command:**

$ sudo yum install java-1.7.0-openjdk-devel

(https://www.digitalocean.com/community/tutorials/how-to-install-java-on-centos-and-fedora)

**Verifying JDK installation**

$ javac -version

**Installing and configuring maven on CentOS**

https://maven.apache.org/download.cgi

$ wget url(http://www-us.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz)

[root@localhost vagrant]# ls

[root@localhost vagrant]# pwd

[root@localhost vagrant]# mv apache-maven-3.3.9-bin.tar.gz /opt/

[root@localhost vagrant]# cd /opt/

[root@localhost opt]# ls

[root@localhost opt]# tar -xzf apache-maven-3.3.9-bin.tar.gz

[root@localhost opt]# ls

[root@localhost opt]# rm apache-maven-3.3.9-bin.tar.gz

[root@localhost opt]# cd apache-maven-3.3.9/

[root@localhost apache-maven-3.3.9]# ls

[root@localhost apache-maven-3.3.9]# cd /bin/

[root@localhost bin]# ls

[root@localhost bin]#

.........................

[root@localhost bin]# cd ..

[root@localhost ~]# pwd

/root

[root@localhost ~]# /opt/apache-maven-3.3.9/bin/mvn --version

==> Right now maven installed under /opt/apache-maven-3.3.9

[root@localhost ~]#

**Setting a path to the maven bin directory**

If we set a path, we can directly access maven command anywhere in the system.

[root@localhost ~]# exit

exit

[vagrant@localhost ~]# sudo vi ~/.bashrc

-----------------------------------------------------------------------------

insert below commands in .bashrc and save & quit

export M2\_HOME=/opt/apache-maven-3.3.9

export PATH=$M2\_HOME/bin:$PATH

-------------------------------------------------------------------------

[vagrant@localhost ~]# source ~/.bashrc

[vagrant@localhost ~]# mvn -version

**Installing git in centOS**

[vagrant@localhost ~]# sudo yum install git

[vagrant@localhost ~]# git --version

**Day 5 (29.12.16)**

**pom.xml**

**pom** - project object model

**groupId:** groupId represents clients reverse domain name

Example: in.javahome, com.icici, net.citi,

**artifactId:** artifactId represents project name

**version:** version of this software

0.0.1

0-Major

0-Minor

1-patch/bugfix

**packaging:** packaging represents either jar, war, ear

**dependencies:** dependencies are nothing but jars on which this project depends on

**Note:** dependencies are added by the development team

**properties:** properties

if there is any text which is repeated across pom.xml file instead of hard coding this value across all the places. it can create a property and put this property across all the places.

<properties>

<spring.version>4.2.8.RELEASE</spring.version>

</properties>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-content</artifactId>

<version>${spring.version}</version>

</dependency>

**Plugins:**

By using Plugins we can add functionalities

**Maven repository**

Maven repository is a server/location where it maintain all dependencies

Maven supports three types of repositories

**1. Central Repository**

* It is repository maintained by Maven over internet
* When we build first time, it goes to central & downloads all those dependecies and it will create a package out of that dependency.
* When we run build first time, maven goes to central repository downloads all required dependencies for creating a package.

**2. Local Repository:**

This repository is maintain in the local machine.

In linux/unix, the loctation of local repository path is: $ ~/.m2/repository/

In linux/unix, the local repository path is: <user-hime>/.m2/repository/

**3. Remote Repository:**

For setting up remote repository we have various softwares

1) nexus

2) artifactory

**Versions**

1. RELEASE version

2. SNAPSHOT version

**1. RELEASE version**

* RELEASE meaning the current version of software was released. There won't be any modifications for this version.
* Maven downloads release versions first time from the central or remote repository, second time onwards it uses the jar file in the local repository.

**2. SNAPSHOT version**

If the dependency is SNAPSHOT version, meaning is this version is under development there might be new changes pushed by the developers very often so maven every time download SNAPSHOT version from the central or remote repository.

**NOTE:** if snapshot is not explicitly mentioned then it becomes release version

**Transitive dependencies**

for my project a.jar is dependency, b.jar and c.jar are transitive dependencies.

**Maven Build Life Cycle**

Step1: It validate pom.xml -->

Step2: Compile source code --> mvn compile

Step3: Execute JUnit test cases --> mvn test

Step4: Package: it creates a package. --> $ mvn package

Step5: Verify: It also can verifies integration test cases

Step6: Install: It installs artifact(jar/war/ear) in local repository. --> $ mvn install

Step7: Deploy: It places the artifact in the remote repository. --> $ mvn deploy

$ mvn clean package - delete target file generated by ......................

**Day 6 (30.12.16)**

**Continuous Integration(CI)**

It is a process where bunch of developers push their changes every day to the SCM(Source Code Manager) like Github, then take the latest code and run builds(maven) on this code & generate an artifact. This process called as Integration Process. If we repeat this process again and again it is called as Continuous Integration(CI).

--> The objective of CI is find syntactical and functional errors as early as possible which lets developers to fix them immediately. That is the idea behind CI.

--> To implement CI process, there are CI tools like Jenkins, Bamboo, Team City.

--> Lets begin with Jenkins

**Jenkins**

It is a popular open source CI tool which provides lots of inbuild functionalities.

**Steps to install Jenkins in CentOS or Linux**

For Java based applications, the prerequisites for using Jenkins server is

1. JDK

2. Maven

3. Git

**Installation of Jenkins**

**To install Jenkins,**

$ sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

**To verify the status of Jenkins,**

$ sudo service jenkins status

**To start the Jenkins**

$ sudo service jenkins start

**To restart jenkins automatically when VM restarts,**

$ sudo chkconfig jenkins on

By default Jenkins use 8080 port number

To access jenkins from web browser,

192.168.70.10:8080

**Day 7 (2.1.17)**

**Creating a first Jenkins Job**

* The objective is take the project source from github use maven to run a build on this project and generate artifact.
* When we ran this job the build failed saying mvn clean package ..........
* In our case mvn is in the path, Jenkins is not able to ............ we can configure maven installation location.

**Steps to configure maven installation location**

Jenkins Home -> Manage Jenkins -> Global Tool Configuration -> Add Maven

In a Jenkins, using jenkins job not only build it can perform any kind of activity. for example, executing script, creating & deleting a folder.

**Example2:**

Using previous job, instead of manual build, we can automated process ....... every hour

Open ims-dev project-> Click configure under Build Triggers -> Select Build periodically

\*/1\*\*\*\*

**Example3:**

Triggering a build only when there is a change in git repository.

Under build triggers we must enabled "Build when a change is pushed to Github" and "Poll SCM"

**Jenkins Workspace**

It is a location in the Jenkins server where a project is clone and maven goal is executed.

**Day 8 (3.1.17)**

**Configuring Email notifications**

- For this configuration initially we need to set up SMTP server

**Configuring Gmail SMTP server**

- SMTP is a server used for sending & receiving emails

- In Jenkins Home, Manage Jenkins-->Configuration-->Email Notification

Q: How Jenkins find changes in SCM?

Ans: It maintain last revision number in log file.

Plugins are maintained by the Jenkins under Jenkins home/plugins

Jenkins--> Manage Jenkins-->

**Master-Slave Configurations in Jenkins**

The idea of Master-slave to distribute jobs across the multiple servers which decreases/reduces load on single server.

Jenkins Home--> Manage Jenkins --> Manage Nodes--> New Node

Q: How to configure a job to run under slave?

Ans: while configure a job select the option "restrict where this project can be run" and put label of the slave.

**Day 9 (4.1.17)**

**User Management**

Jenkins Home--> Manage Jenkins--> Manage Users--> Create a user

Jenkins Home--> Manage Jenkins--> Configure Global Security

**Web Servers**

Web servers are helpful for running web based applications.

Web apps are two kinds

1. Static web apps - which uses a plain html

2. Dynamic web apps -

For Dynamic web apps we can use php, servlets & JSP, ASP .net, python etc...

IQ: Why we need web servers & what functionalities it provides for us?

Ans: 1. Web servers provides implementation of http & https protocols.

2. Web servers provide in built multithreading support.

3. They do provide in-built security.

Different web servers available in the market.

1. Apache tomcat

2. Oracle weblogic

3. IBM Websphere

4. SUN Glassfish

5. JBoss

6. Apache

**1. Apache tomcat:**

**Installing tomcat:**

Latest version as of today is tomcat 9, but most of the applications used tomcat 7&8

**Pre-requisite:**

JDK is a prerequisite for tomcat.

Step 1: Change Directory into /opt

Step 2: use wget & downloading tomcat

**Starting a tomcat:**

]$ cd into tomcat home/bin

[root@localhost bin]# pwd

/opt/apache-tomcat-8.5.9/bin

[root@localhost bin]#./startup.sh

**Stopping a tomcat:**

]$ cd into tomcat home/bin

[root@localhost bin]# pwd

/opt/apache-tomcat-8.5.9/bin

[root@localhost bin]#./shutdown.sh

Tomcat users 8080 has a default port number

**Deploying Web application in tomcat**

There are different ways to deploy web application in tomcat

**Option 1:** take the war file & put it in tomcat home/webapps

- Take the war file from server1 & place it in server2 tomcat/webapps folder

**Day 10 (6.1.17)**

**To deploy a project in tomcat server**

[root@localhost webapps]# scp vagrant@192.168.70.10:/var/lib/jenkins/workspace/myweb-dev/target/myweb.war

Q: How to change the port number of tomcat?

Ans:

tomcat-home/conf/server.xml

------------------------------------------------------------------

Default port number for tomcat is: 8080

To change port number, edit server.xml in tomcat8/conf

[root@localhost conf]# pwd

/home/vagrant/tomcat8/conf

path:

-->

<Connector port="4321" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

<!-- A "Connector" using the shared thread pool-->

After change port number, have to restart the server

-----------------------------------------------------------------

\*\*\*The configure tomcat to start at the time of booting.

Q: How to check tomcat logs?

Ans: tomcat-home/logs

**Tomcat Manager Application**

Using this application you can monitor which projects are deployed on the tomcat along with that you also can deploy new application by uploading a war file.

In tomcat latest version, Tomcat Manager Application is disabled by default. In according to enable

**Change1:**

Modify tomcat-users.xml which is under tomcat's conf folder. We need to add two roles & one user as follows

------------------------------------------------------------------

To create user for tomcat, go to tomcat8/conf edit tomcat-users.xml & add below script

<!--

<role rolename="tomcat"/>

<role rolename="role1"/>

<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>

<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>

<user username="role1" password="<must-be-changed>" roles="role1"/>

-->

<role rolename="tomcat"/>

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<user username="jvk" password="password" roles="tomcat,manager-gui,manager-script"/>

After add roles & user, have to restart the server

------------------------------------------------------------------

**Change2:**

Modify context.xml under webapps/manager/META-INF

-----------------------------------------------------------------------

[root@localhost META-INF]# pwd

/home/vagrant/tomcat8/webapps/manager/META-INF

Edit context.xml in /home/vagrant/tomcat8/webapps/manager/META-INF

- Delete the value in between context

Have to restart the server

------------------------------------------------------------------------

**Deploying war file in tomcat using automation process**

It can achieve this

1. By writing a shell script

I upload the WAR to my home directory, cd to /usr/local/tomcat, then run the following commands:

bin/shutdown.sh

rm webapps/ROOT.war

rm -rf webapps/ROOT

cp ~/ROOT.war webapps

bin/startup.sh

As I use maven to generate my builds in tomcat inside a ubuntu box, I have a script called

install\_wars.sh

mvn clean install

service tomcat7 stop

find /var/lib/tomcat7/webapps/ -mindepth 1 -maxdepth 1 -type d -exec rm -rf {} \;

find . -name \*.war -exec cp {} /var/lib/tomcat7/webapps/ \;

service tomcat7 start

2. using Jenkins deploy to container plugin

**Step1:**

Manage Jenkins--> Available & search for deploy to container plugin & Install without restart

SonarQube - code Analyzer

**Day 11 (10.1.17)**

**Chef**

- Chef is Configuration Management Tool. Its main purpose is to automate software installation and configuration.

- Some people use chef for deployments as well.

Chef has mainly 3 components

1. Servers/Nodes

2. Chef Server

3. Chef workstation

**Chef Workstation:**

Chef Workstation is a PC or where DevOps engineers develop cookbooks & recipes etc

**Chef Server:**

It is server where we maintain our cookbooks

Ex: JDK Cookbook(installs JDK), Jenkins Cookbook(installs Jenkins & Configure Jenkins)

**Node:**

Node is machine (can VM or physical server or AWS EC2 instance) which is used for installing & configuring softwares using Cookbooks.

In our Tutorial workstation is our laptop, Chef server is hosted chef server in cloud and a node is a VM created using vagrant.

**Setting up Chef server**

Step1: Signup for chef server

Step2: Login to chef server

Step3: Create Organization in the chef server

**Setting up Chef Workstation using which we can communicate with chef server**

Step1: Download Starter Kit under Organization

Step2: Copy Starter Kit under C Drive

Step3: Extract Starter Kit

**Verifying chef server connectivity:**

As part of Starter Kit under chef-repo/.chef, we can find two files.

1) knife.rb - it has the details about chef server url

2) username.pem - it has private key

Q: How workstation/knife knows where is the chef server?

Ans: It uses knife.rb which has the details about chef server url & private key for authentication

**Day 12 (11.1.17)**

metadata.rb

**Creating a cookbook**

**To generate cookbook**

cookbooks$ chef generate cookbook cookbook name

**Cookbook:** cookbook is collection of recipes, files, templates etc...

**Recipe:** It is a collection of chef resources

**Chef resources:**

For Example:

file resource

to create file,

**-> under default.rb**

-------------------------------------------

1) file '/tmp/welcome.txt' do

action :create

end

(or)

2) file '/tmp/welcome.txt'

---------------------------------------------

both are same

**-> Using this cookbook we want to create welcome.txt file on a node.**

**Step1:** First we must upload chef cookbook into chef server.

$ knife upload cookbook cookbook name

$ knife upload cookbook hello-world

**Step2:** Configuring a node using hello-world cookbook

- Bootstrapping a node

$ knife bootstrap <IP Address> -x <USERNAME> -P <PASSWORD> --run-list 'recipe[hello-world::default]'

$ knife bootstrap 192.168.70.12 -x root -P vagrant --run-list

'recipe[hello-world::default]'

$ knife bootstrap 192.168.70.12 -x root -P vagrant -N Node-1

'recipe[hello-world::default]'

**Knife Bootstrap process:**

Step1: knife will ssh into the node.

Step2: It installs chef client on the node.

Step3: Chef client on the node talks to chef server.

Step4: Chef client downloads the cookbook mention in the run-list

Step5: It executes default recipe according to our run-list

**Note:** Bootstrapping a node must be done only once.

**Day 13(18.1.17)**

**Cookbook\_file**

This provides

Steps:

1. Create files directory under a cookbook

2. Under files, create index.html and put the code inside the file

cookbook\_file '/var/www/html/index.html' do

source 'index.html'

action :create

end

-> cookbook\_file resource takes the file specified in source attribute(source 'index.html') from files directory and copies the content in specifies path('/var/www/html/index.html')

-> it deals with static content only

**Template Resource**

This is same as cookbook\_file resource but it deals with dynamic content

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

source 'index.html.erb'

end

-----------------------------

<%=

node ['time']['timezone']

%>

--------------------------------

package 'Install Apache' do

case node[:platform]

when 'redhat', 'centos'

package\_name 'httpd'

when 'ubuntu', 'debian'

package\_name 'apache2'

end

end

**Day 14 (19.1.17)**

**Custom Attribute:**

- We can create our own custom attributes which is specific to our need.

1. Create attributes folder under a cookbook

2. Under attributes create default.rb

**Declaration of attributes**

default['hello-world']['author'] = 'Hari'

default['hello-world']['company'] = 'Java Home'

**Using attributes in the recipes**

file '/tmp/att-demo.sh' do

content node['hello-world']['company']

end

**Note:** Attribute is specific to the node

file '/tmp/att-demo.sh' do

content "This is created by #{node['hello-world']['company']}"

end

**Using local variables**

#local variable

package\_name = 'httpd'

# Initialization local variable from platform

case node[:platform]

when 'redhat', 'centos'

package\_name = 'httpd'

when 'ubuntu', 'debian'

package\_name = 'httpd'

end

package package\_name do

action :install

end

**Splitting Chef recipes into multiple files**

**Day 15 (23.1.17)**

**Data Bags**

- Data bags holds global attributes

- Data bags are created & stored

**Databag structure**

E:\Desktop\Workspace\Databag structure.png

Users is a Data Bag & tomcat.json, mysql.json are items

**- Creating data bag**

$ knife data bag create <databag name>

$ knife data bag create users

**- listing out databags**

$ knife data bag list

**- creating data bag item**

$ export EDITOR=vi

$ knife data bag create <databag name> <item name>

$ knife data bag create users tomcat

$vi tomcat

----------------------------

{

"id": "tomcat",

"userid": "tomcatadmin",

"password": "admin123"

}

---------------------------

JSON- Java Script Object Notation

**Sample JSON structure**

{

"name":"Vamsee",

"age":28,

"isManager":true,

"phone":["9999999999","8888888888"],

"address":{

"city":"Bangalore",

"country":"India"

"pin":560037

"state":"Karnataka"

}

}

**- Deleting a item in data bag**

$ knife data bag delete <databag name> <item name>

$ knife data bag delete users tomcat

**- Editing a item in data bag**

$ knife data bag edit <databag name> <item name>

$ knife data bag edit users tomcat

**- Accessing data bag attributes in our chef code**

----------------------------------------

# Syntax to accessing data bag item in chef recipe

#'users' is a data bag name & 'tomcat' is data bag item

tomcat = data\_bag\_item('users', 'tomcat')

file '/tmp/databagdemo.txt' do

content tomcat['password']

end

----------------------------------------

**- creating data bag item from file. This is recommended approach**

Step1: create data\_bags under chef-repo

Step2: Under this directory create a folder 'Users'

Step3: Create mysql.json file under Users

$ vi mysql.json

--------------------------

{

"id": "mysql",

"userid": "root",

"dbpassword": "db1234"

}

----------------------------

$ knife data bag from file <databag name> <item name>

**Encrypted Data bags**

we can encrypt attributes in a data bag. it is called as Encrypted data bag. This is imp when we store sensitive data like passwords.

**Day 16 (24.1.17)**

**Encrypted Data bags**

- we can encrypt attribues in a data bag. it is called as Encrypted data bag. This is imp when we store sensitive data like passwords.

- Encryption & Decryption happens based on a key.

- Creating a key file which is used for Encryption & Decryption.

]$ openssl rand -base64 512 | tr -d '\r\n' > <key file>

]$ openssl rand -base64 512 | tr -d '\r\n' > javahome-databags-key

**- Path of the key file**

]$ knife data bag create users passwords --secret-file javahome-databags-key

**- To see the content of key file**

]$ knife data bag show users passwords --secret-file javahome-databags-key

- we need to copy a secret file to the node such that node uses this secret file for decrypting the data

**Roles**

- in our infrastructure we see web servers, database servers, load balancers etc.

- The benifit of using role is we can group similar servers into one category. for example webservers role.

- the benifit is easy to manage a runlist i.e., we can create web servers role in this role we can mention the run list and we can add all web servers under this role. Down the line we want to change the runlist of all webservers, instead of modifying all the webservers runlist, we can modify the runlist of role

**Creating a role**

]$ knife role create <role name>

]$ knife role create webservers

**Environments**

In S/w development we have different environments. for example, Dev, QA, UAT & Prod

For Ex all the environments are having cookbook with a version 0.0.1. & lets say we are developing cookbook for next release which is 0.0.2. First we need deploy the latest version in Dev. i.e., the cookbook version in Dev must be 0.0.2 & in other environments the version 0.0.1. this is achieved using Environment

IQ: In chef, I want version of cookbook as 0.0.2 in one environment & 0.0.1 in other environment. Is this possible.

Ans. Yes. Using Environment

IQ: In a Dev Environment we want to install Apache server & Database server on single node. In Prod we want install them on different nodes.

Ans:

**Test Kitchen**

Using Test Kitchen we can speed up the development of our chef cookbooks.

**Development approach without Test Kitchen**

Step1: Write cookbook

Step2: Upload cookbooks to chef server

Step3: Maintain a separate VM for testing it

**Development approach with Test Kitchen**

]$ kitchen converge

**- How to login to the node**

]$ kitchen login

**Dependency Management in chef**

Some times in our cookbook we use the other cookbooks which is written by community

Ex: we want to install Java and tomcat. for this we can create a cookbook, in this cookbook we can mention the dependencies java & tomcat cookbooks

If java & tomcat depends on other cookbooks, we can call them as transitive dependencies.

we should manually download our dependencies & their transitive dependencies follwed by upload all those cookboos to chef server & use them for configuring our nodes.

]$ berks --version

**berks**

It is dependency management tool which takes away all our complex.

**Specify dependency in our cookbook**

Under metadata.rb

depends 'java', '~> 1.46.0'

**berks install**

]$ berks install

this command downloads dependencies & transitive dependencies.

**Uploading dependencies & transitive dependencies to the chef server**

]$ berks upload

IQ: I want to stop the upload of a cookbook with specific version to the chef server

Ans: we can use --freeze flag while uploading a cookbook such that no one can upload cookbook with same version

IQ: What is wrapper cookbook?

Ans: It is a cookbook developed by us which wraps community cookbooks.