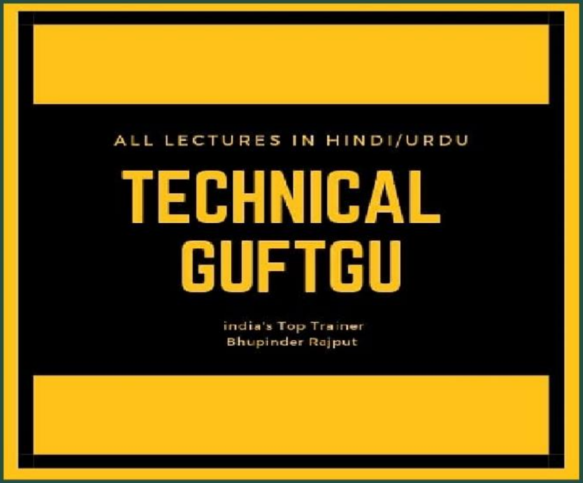
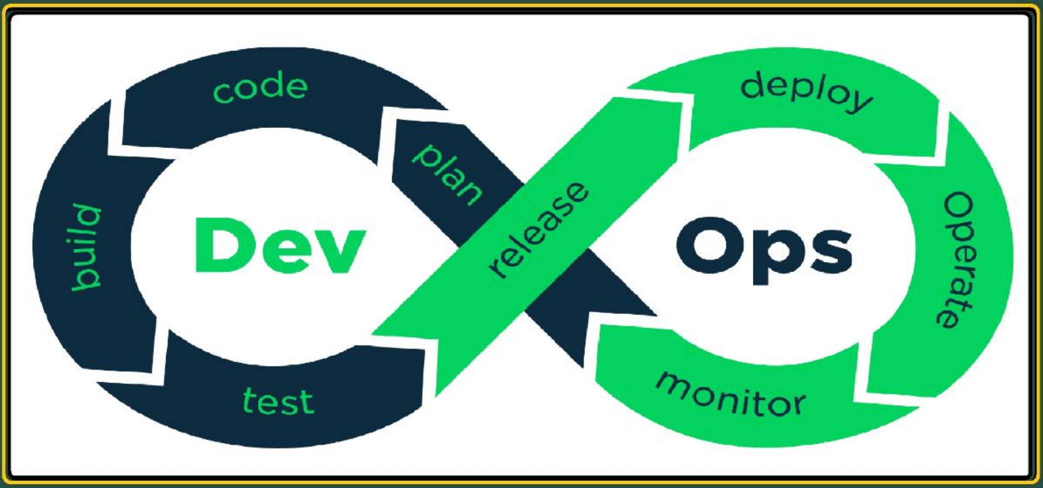




This DevOps Notes is Presented by . . . . .

Sir BHUPINDER RAJPUT

Good morning … Namaste ….As'Salamu Alaykum…. Dosto ☺

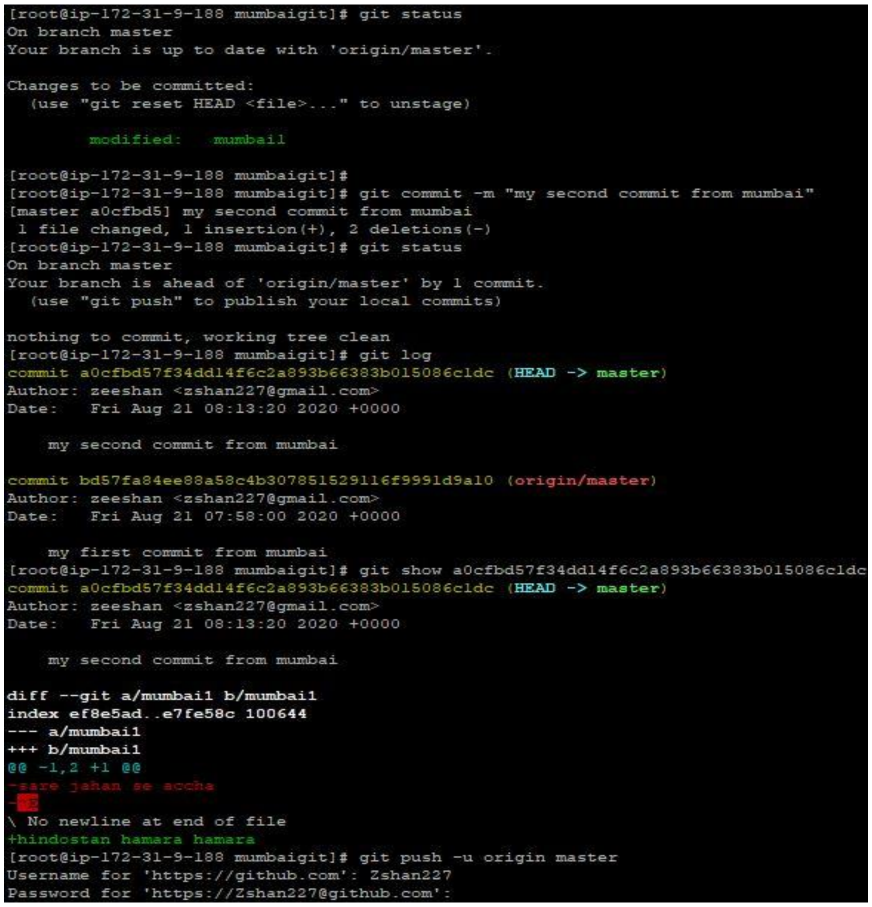
git + GitHub

Git has three stages of workflow -1. **Working aria 2. Staging aria 3. Local Repository**.
We send data or code from working aria to staging aria by **add command** and staging aria to Local repository by **commit command** and finally send data/code from Local repo to central repo by **push command**.

Update Linux operating system in working aria (Mumbai Ec2-user)

```
    # yum update -y
    # yum install git -y
    # which git
```
User /bin /git
```
    # git -version
2.23.3
    # git config --global user.name "Zeeshan"
    # git config --global user.email Zshan227@gmail.com
    # git config -list
```
(this command shows the all configurated list)

User.name=Zeeshan
User.email=Zshan227@gmail.com



```
[root@ip-172-31-9-188 mumbaigit]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   mumbai1

[root@ip-172-31-9-188 mumbaigit]#
[root@ip-172-31-9-188 mumbaigit]# git commit -m "my second commit from mumbai"
[master a0cfbd5] my second commit from mumbai
 1 file changed, 1 insertion(+), 2 deletions(-)
[root@ip-172-31-9-188 mumbaigit]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[root@ip-172-31-9-188 mumbaigit]# git log
commit a0cfbd57f34dd14f6c2a893b66383b015086c1dc (HEAD -> master)
Author: zeeshan <zshan227@gmail.com>
Date:   Fri Aug 21 08:13:20 2020 +0000

    my second commit from mumbai

commit bd57fa84ee88a58c4b3078515291l6f9991d9a10 (origin/master)
Author: zeeshan <zshan227@gmail.com>
Date:   Fri Aug 21 07:58:00 2020 +0000

    my first commit from mumbai
[root@ip-172-31-9-188 mumbaigit]# git show a0cfbd57f34dd14f6c2a893b66383b015086c1dc
commit a0cfbd57f34dd14f6c2a893b66383b015086c1dc (HEAD -> master)
Author: zeeshan <zshan227@gmail.com>
Date:   Fri Aug 21 08:13:20 2020 +0000

    my second commit from mumbai

diff --git a/mumbai1 b/mumbai1
index ef8e5ad..e7fe58c 100644
--- a/mumbai1
+++ b/mumbai1
@@ -1,2 +1 @@
-sare jahan se accha
-[illegible]
\ No newline at end of file
+hindostan hamara hamara
[root@ip-172-31-9-188 mumbaigit]# git push -u origin master
Username for 'https://github.com': Zshan227
Password for 'https://Zshan227@github.com':
```

Now work inside the Mumbai machine, create Directory and make file inside local-repo

```
[Ec2-user] # mkdir mumbaigit
[Ec2-user] # cd mumbaigit
[Mumbaigit]# git init                  ( init command turn Dir into local Repo)
[Mumbaigit]# cat >Mumbai1              (write inside the [Dir] local repo by cat > command)
        # cat Mumbai 1                 (to check the data/code what has been written in repo)
```

Put and write some code/data inside the file mumbai1, and come out by **Ctrl+d**

SARE JAHAN SE ACCHHA

```
        # git status
```

Untracked files: Mumbai1 (it's in red color means not added yet staging aria)

```
[mumbaigit]# git add .                 (Add command to add created file to staging aria)

      # git status
```

New file: Mumbai1 (it's in green color means added staging aria)

Now commit data from staging aria to Local repo

```
[Mumbaigit]# git commit -m "first commit from Mumbai"      (m=message)
             # git status
             # git log                 (to check what commits had done when and who did?)
```

You will see commit Id like 12345678KD458F4IW3E4 . Author, Mail id, Date, Time, message: first commit from Mumbai

```
Mumbaigit]# git show <commit-id>       (show command  the content of commit ID)
```

first commit from Mumbai
+ SARE JAHAN SE ACCHHA

If we run the git commit command again it will show nothing to commit, working tree clean means data has been committed.

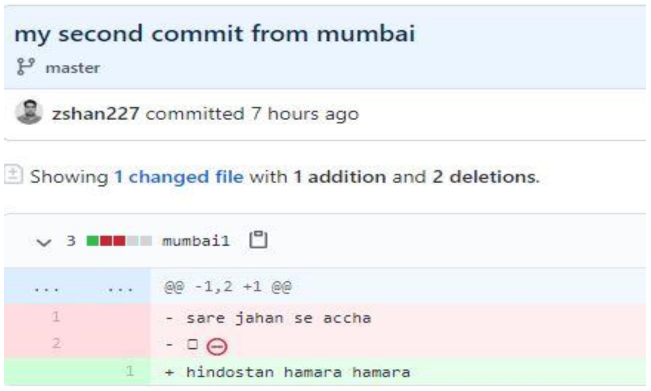
If want to send this code to my central repository, I have to connect local repo to central repo first, for this action I have to create a new repository (any name) and paste the URL of git repo and execute command as given below

```
[Mumbaigit] # git remote add origin https://github.com/zshan227/centralgit2.git
```

Now local repo has been connected to central repo, for pushing data to central repo execute this command

```
[Mumbaigit] # git push -u origin master        (push command local repo to central repo)
```

It will ask for **username and password** of your git hub account, after filling this and you can see all committed data/code inside central repo.



my second commit from mumbai
master
zshan227 committed 7 hours ago

Showing 1 changed file with 1 addition and 2 deletions.

```
3 mumbai1
... ... @@ -1,2 +1 @@
1       - sare jahan se accha
2       - □ ⊖
    1   + hindostan hamara hamara
```

Now create a machine in Singapore region and connect to git hub.

```
[ec2-user] # yum update -y
           # yum install get -y
           # git config -global username
           # git config --global user.name "Ahmad"
           # git config --global user.email flitz.power@gmail.com
           # git config --git remotelist
```

User.name=Ahmad
User.email=flitz.power@gmail.com

```
[Ec2-user] # mkdir singaporegit
           # cd singaporegit
[singaporgit]# git init

Initialized empty git repository in Home/ec2-user/singaporegit/.git/

[singaporgit]# ls -a
 .   ..    .git
[singapurgit]# git remote add origin
https://github.com/zshan227/centralgit2.git
```

Now local repo has been connected to central repo, for **Pulling data to central repo**, execute this command

```
[singaporgit]# git pull -u origin master
```

(you can execute without **-u** as well)

Now you can see it has pulled all data/code from remote directory central repo, all details and commits has been done by other Mumbai machine.

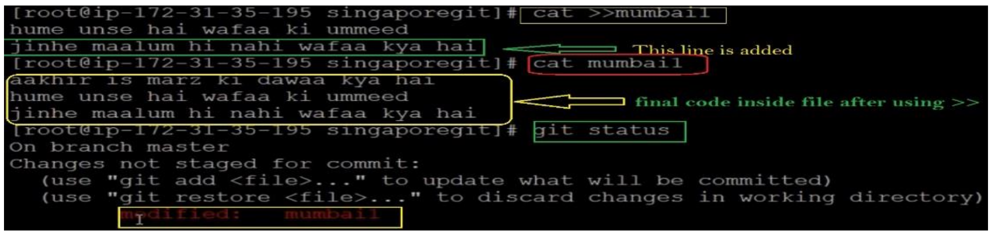
```
[singaporgit]# cat >mumbai1
HINDOSTAN HAMARA HAMARA
```

(> used to write and overwrite code inside mumbai1)

Ctrl+D

If you want to add lines or something on this code inside the file use command **# cat >>file**

```
          # git status
Modified: mumbai1
```



```
[root@ip-172-31-35-195 singaporegit]# cat >>mumbai1
hume unse hai wafaa ki ummeed
jinhe maalum hi nahi wafaa kya hai
[root@ip-172-31-35-195 singaporegit]# cat mumbai1
aakhir is marz ki dawaa kya hai
hume unse hai wafaa ki ummeed
jinhe maalum hi nahi wafaa kya hai
[root@ip-172-31-35-195 singaporegit]# git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   mumbai1
```

```
     # git add .
               # git status
Modified: mumbai1

               # git commit -m "first commit from singapore"
               # git log
```

Now it will show all messages **commits ids** and steps done by both Mumbai and Singapore machines

```
          # git show 12345678KD458F41W3E4
```

SARE JAHAN SE ACCHHA — Old commit
+ HINDOSTAN HAMARA HAMARA — new commit

Push data/code into central git from local repo

```
# git push -u origin master
```

(you can use **-f** instead of -u for **force** push)

Now Enter username and password of git hub account, after that you will see all new and old **commits updates** in central git, click **mumbai1** file you will get code "HINDOSTAN HAMARA HAMARA"

GITIGNORE-This command is used to ignore some specific file which we don't want to add & commit.

```
[mumbaigi1]# vi .gitignore
 * CSS
 * java
```

* **used to ignore particular file**

Esc+:wq

```
# git add .gitignore
# git commit -m "ignore file format"
# git status
```

can use single comma as well

Nothing to commit, working tree is clean, now create some files in different formats by using **touch command**

```
# touch file1.txt file4.java file3.css file5.java file2.txt
# ls
# git status
File1.txt
File2.txt
```

only showing 2 untracked files rest three have been ignored

```
# git add .
# git status
```

Now both files have been added and showing us in green color after status command

```
File1.txt
File2.txt
# git commit -m "IGNORE JAVA CSS FILES"
# git log
```

```
12345678KD458F41W3E4  (HEAD -> master)
12345678KD458F41W3E4  (HEAD -> master)
12345678KD458F41W3E4  (HEAD -> master)
```

So many commit Ids are showing

```
# git show 12345678KD458F41W3E4
IGNORE JAVA CSS FILES
# touch Zeeshan.java
# git status
```

Nothing to commit, working tree is clean, now create some files in different formats by using **touch command**

```
# touch Zeeshan.txt
# git status
Zeeshan.txt
```

(Again it showed text file, not java file means ignored)

If I want to latest commit, last 2 commits, last-n commits and all commits in one line.

```
# git log -1
# git log -2
# git log -oneline
```

```
12345678KD458F41W3E4 (HEAD -> master) message "1"
12345678KD458F41W3E4     message "2"
12345678KD458F41W3E4     message "3"
```

So many commits are showing in one column

If I want to find specific commit, Acton and file use grep command with specific name rest will be ignored.

```
[mumbaigi1] # git log --grep "XYZ"
XYZ=zee,ignore,filename,java,Hindostan
```

GIT BRANCHES:

- Each task has one separate branches, after done with code other branches merge with master.
- This concept is useful for parallel development. Master branch is default branch
- We make branches, one for little features and other one for longer running features.
- It allows keeps the main master branch free from error.
- Files created in workspace will be visible in any of the branch workspace, until you commit, once
- you commit then those files belong to that particular branch

How to create Branches:

```
 [ec2-user] # cd mumbaigit
[mumbaigit] # git log --oneline
            # git branch
        *master
            # git branch branch1
            # git branch
        *master
         Branch1
            # git checkout branch1
```
(switch to branch branch1)
```
       master
        *Branch1
            # git branch -d <branchname>
```
(to delete any branch)

Branches Working process:

```
       # git checkout branch1
       # cat >shanfile
```
(create shanfile and write anything inside **by** >)
```
     Nothings is better that something
```
If you want to add lines or something on this code inside file use command **# cat >>file**, for rewrite use **>**

```
       # ls
     branch1 shanfile
            # git checkout master
            # ls
     mumbai1 shanfile
```

Shanfile and code is **showing** inside master branch because it hasn't committed with any branch yet.

```
            # git commit -m "branch1 first commit"
            # git log -oneline
     Branch1 first commit
            # git checkout master
            # git log -oneline
```
Shanfile & code will **not show** inside master branch because that file has been committed with Branch1.

How to Merge Branches: we use pulling mechanism, we can't merge branches of different repositories

```
       # git checkout master
            # git merge branchA
```
(to verify the merge)

Executed checkout command before merge command means, you wanted to merge any branch with master branch

# git log –oneleine

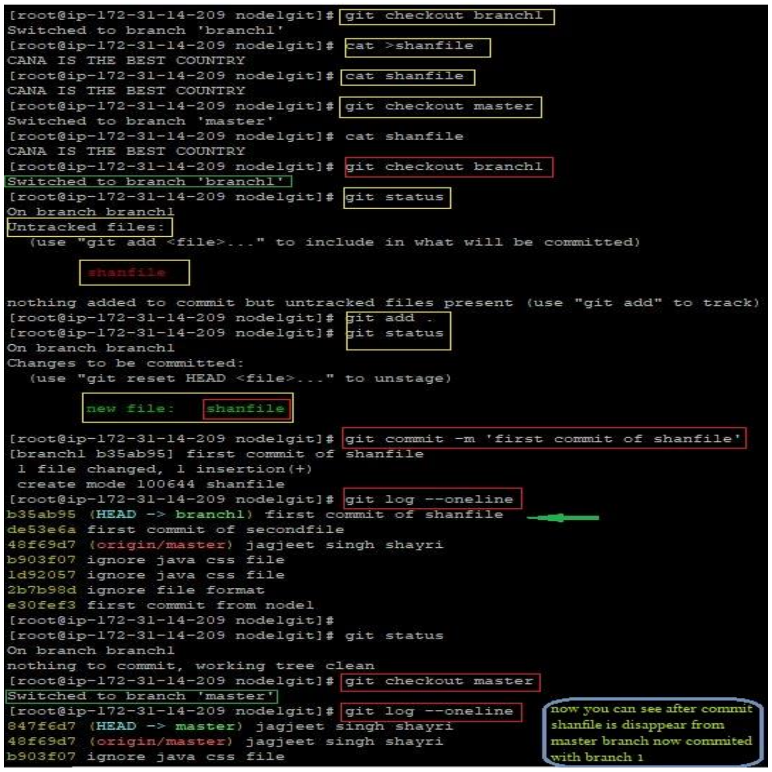
Now you can see **All commits** of both branches which have been merged together

# ls

Now you can see **All files** of both branches which have been merged together.

git **push** origin **master** (to push central repo lit git hub)

Enter username & password you can see merged data in central repository on git hub **.**



```
[root@ip-172-31-14-209 nodelgit]# git checkout branchl
Switched to branch 'branchl'
[root@ip-172-31-14-209 nodelgit]# cat >shanfile
CANA IS THE BEST COUNTRY
[root@ip-172-31-14-209 nodelgit]# cat shanfile
CANA IS THE BEST COUNTRY
[root@ip-172-31-14-209 nodelgit]# git checkout master
Switched to branch 'master'
[root@ip-172-31-14-209 nodelgit]# cat shanfile
CANA IS THE BEST COUNTRY
[root@ip-172-31-14-209 nodelgit]# git checkout branchl
Switched to branch 'branchl'
[root@ip-172-31-14-209 nodelgit]# git status
On branch branchl
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        shanfile

nothing added to commit but untracked files present (use "git add" to track)
[root@ip-172-31-14-209 nodelgit]# git add .
[root@ip-172-31-14-209 nodelgit]# git status
On branch branchl
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   shanfile

[root@ip-172-31-14-209 nodelgit]# git commit -m 'first commit of shanfile'
[branchl b35ab95] first commit of shanfile
 1 file changed, 1 insertion(+)
 create mode 100644 shanfile
[root@ip-172-31-14-209 nodelgit]# git log --oneline
b35ab95 (HEAD -> branchl) first commit of shanfile
de53e6a first commit of secondfile
48f69d7 (origin/master) jagjeet singh shayri
b903f07 ignore java css file
1d92057 ignore java css file
2b7b98d ignore file format
e30fef3 first commit from nodel
[root@ip-172-31-14-209 nodelgit]#
[root@ip-172-31-14-209 nodelgit]# git status
On branch branchl
nothing to commit, working tree clean
[root@ip-172-31-14-209 nodelgit]# git checkout master
Switched to branch 'master'
[root@ip-172-31-14-209 nodelgit]# git log --oneline
847f6d7 (HEAD -> master) jagjeet singh shayri
48f69d7 (origin/master) jagjeet singh shayri
b903f07 ignore java css file
```

now you can see after commit shanfile is disappear from master branch now commited with branch 1

GIT CONFLICT: When same files having different content in different branches, if you do merge conflict can occur. (Resolve conflict then add and commit)

# Cat >**shanfile**

hello zee ctrl+d

git add .

# git commit -m "commit before conflict"

git **checkout** branch1 switch to branch1

# Cat >**shanfile** create same file but write different code inside

hello shan ctrl+d

# git commit -m "commit from branch1"

git **checkout** master switch to branch1

# git merge branch1

Merge failed: fix conflict, then commit result

# vi shanfile **(update inside shanfile)**

```
<<<<<<HEAD                                          delete HEAD
Hello zee
============                                        delete =====
Hello shan
>>>>>> branch1                       Esc+:wq
```

You can change data according to yourself which you exactly needed before conflict do changes in file git will understand the change and execute data accordingly.

```
# git status
    # git add .
    # git commit -m "Resolve conflict"
    # git log --oneline
```

12h3a8g90 (HEAD → master) Resolve conflict

GIT BRANCH STASH: If your code is in progress and suddenly need changes through client escalation you have to **keep aside** current code and have to work on new features for some hours.
You can't commit your parallel code so you need some temporary storage to store partial changes and later on commit it. To stash an item only applied for **modifies files** not new files.

```
                # git checkout master
                # cat >zakfile
                # git commit -m "zakfile commit"
                # vi zakfile
My super zak code-1                 Esc+:wq        (Boss need other work so stash the data of zakfile)
        # git stash
                # Cat zakfile                 (zakfile empty, data stashed ,now you can do new work)
                # Git stash list
Stash (0) : WIP on master 1372ee7 .zakfile

                # vi zakfile
My super zak code-2                 Esc+:wq
                # cat zakfile
My super zak code-2
                # git stash
                # git stash list
Stash (0)
Stash (1)
                # cat zakfile                              (zakfile empty, data/code has been
stashed)
```

Now going to do old pending work

```
                # git stash apply stash@{1}

        # cat zakfile
My super zak code-2
                # git add.
                # git commit -m "zakfile commit done"

                # git stash apply stash@{0}
Auto merging zakfile; CONFLICT:Merge conflict in zakfile
```

# Vi zakfile **(update inside zakfile)**

```
<<<<< update stream
 My super zakcode-1
 ===================
 My super zakcode-2
>>>> stashed changes
```

final code would be my super zakcode-2
delete ========

Esc+:wq

```
# git add .
# git commit -m "zakfile commit done2"
# git status                         (empty)
# git log --oneline
Zakfile commit
Zakfile xommit done
Zakfile commit done2
# git stash list
Stash (0)
Stash (1)
```

(still available in stash list delete it by **# git stash clear**, recheck by **stash list** command)

GIT RESET: It is a powerful command that is used to udo local changes to the state of a git repository. It used to undo the add . command.

```
# cat <zeekile
Zee is the shan
# git add .

# git status
New file:zeefile
```

(now realized did mistake in data wanted to change)

- **To reset from staging aria**

```
# git reset Zeefile
# git reset .
```

(removed data from staging aria)

```
# git status
Zeefile
# git add .

# git status
Zeefile
```

- **To reset from staging aria**

```
# git reset -hard

# git status
One branch master nothing to commit: working tree clean
```

GIT REVERT:

Revert command helps you to undo the existing commit, it doesn't delete any data instead rather get creates a new commit with the included previous files reverted to the previous stat.
So, history moves forward while the stat of your file moves backward.

```
          # cat >fileZ
```
I LOVE MY INDIA
```
                # git add.
          # git commit -m "fileZ commit"          after commit I realized did wrong commit
          # git log –oneline
```

Now you can see so many commits copy previous commit id just before the mistake and paste on revert command

```
          # git revert 12h3a8g90
```

Wrong commit **undo** state moves to backward also write a mesage in this commit "please ignore previous commit"

How to remove untracked files

```
          # git clean -n                                          dry run
                # git clean -f
forcefully
```

Git Tags: Tag operation allows giving meaningful name to a specific version in the repository.

```
To Apply Tag      # git tag -a<tag-name> -m "message" commit-id
                   # git tag -a Zeeshan -m "love you India" 12h3a8g90g6k
To see tag       # git tag
                   # git show tag-name            to see particular commit content by using
Tag
To delete a tag    # git tag -d tag-name
```

Git Hub Clone: go to existing repo in git Hub copy the URL of central repo and paste with run command of Linux machine.

```
          # git clone <URL git hub repo>
[ec2-user]      # git clone https://github.com/zshan227/centralgit2.git
```

It creates a **local repo** automatically inside Linux machine with the same name of git hub account.

```
          # ls
Mumbai1git  zeefile centralgit node1git
```

Both repositories can connect together easily by master branch

Download & install Chef and create Cookbook, Recipes

- Wget <chef download link>
- Yum install <paste> chef -workstation downloaded file
- mkdir cookbooks
- cd cookbooks/
- chef generate cookbook **zee**-cookbook
- cd **zee**-cookbook
- yum install tree -y
- tree
- chef generate recipe **zee**-recipe
- **cd ..**
- vi **zee-cookbook/recipes/zee-recipe.rb**

  I + Enter then <paste code>

```
file '/myfile' do
content 'Welcome to Zeeshan Ahmad'
action :create
end
```

enter+esc+:wq

- Chef exec ruby -c **zee-cookbook/recipes/zee-recipe.rb** (check the syntax)
- Syntax **OK** (run the chef client)
- **Chef-client -zr "recipe[zee-cookbook::zee-recipe]"**
- **Cat** /myfile(xyz) (also try **ls** /) (to check inside the file)

Apache server:

```
[cookbooks]#chef generate cookbook apache-cookbook
#cd apache-cookbook
#chef generate recipe apache-recipe
#cd ..
#vi Apache-cookbook/recipes/apache-recipe.rb
```

I + Enter then <paste code>

```
package 'httpd' do
action :install
end

file '/var/www/html/index.html' do
content 'Welcome to Wafzee website'
action :create
end

service 'httpd' do
action [:enable, :start]
end                                                      esc+:wq
```

```
#Chef-client -zr "recipe[Apache-cookbook::Apache-recipe]"
```

Now ping public IP address to see content on apache website

ATTRIBUTES:

What is this: Attributes is a key value pair which represent a specific detail about node.

Who used? Chef client

Why used? To determine

- current state of node?
- what was the state of the node at the end of previous chef client run?
- What should be the state of the node at the end of current chef client will run?

Types:	**Priority**
1. Default	1st maximum
2. Force-default	2nd more
3. Normal	3rd may be
4. Override	4th less
5. Force override	5th very less
6. Automatic	6th minimum

Who defines Attributes?

Ans: (Node, Cookbooks, Roles, Environment**)** **(attribute defines by Ohai have highest priority)

```
# sudo su
# ohai
# ohai ipaddress
# ohai memory/total
# ohai cpu/0/mhz
# ls
# cd cookbooks
# cd Apache-cookbook
# Chef generate recipe recipe10
# cd ..
# vi apache-cookbook/recipes/recipe10.rb
```

I + Enter then <paste code>

```
File '/besicinfo' do
 Content "this id to get Attributes
 HOSTNAME: #{node['hostname']}
 IPADDRESS: #{node['ipaddress']}
 CPU: #{node['cpu']['0']['mhz']}
 MEMORY: #{node['memory']['total']}"
 owner 'root'
 group 'root'
action :create
end                                        Esc+:wq
```

```
#chef exec ruby -c apache-cookbook/recipes/recipe10.rb
# chef-client -zr "recipe[apache-cookbook::recipe10]"          (call the client)
```

SEE OUTPUT ATTRIBUTES

Insert Linux commands

```
[cookbooks]# vi zee-cookbook/recipes/ABC-recipe.rb
```

I + Enter then <paste code>

```
Execute "run a script" do
Command <<-EOH                  ----> EOH = End of here/hunk (now can write non ruby)
Mkdir /Zeeshandir
touch /Ahmadfile
EOH
End                                  Enter+Esc+:wq
```

Create user

```
#vi zee-cookbook/recipes/ABC-recipe.rb
```

```
User "Rockstar" do
Action :create
End                                  Enter+Esc+:wq          (now run the recipe)
#chef-client -zr "recipe[zee-cookbook::ABC-recipe]"
```

Create group

```
# vi zee-cookbook/recipes/ABC-recipe.rb
```

I + Enter then <paste code>

```
Group "Devops group" do
Action :create
Members 'shan'
Append true
End                                  Enter+Esc+:wq          (now check the recipe)
```

```
# Chef exec ruby -c zee-cookbook/recipes/ABC-recipe.rb
# syntax OK                                                  (now run the recipe)
# chef-client -zr "recipe[Zee-cookbook::abc-recipe]"
# cat /etc/group          (also try ls /)          (to check the group)
```

RUNLIST : To run the recipe in a sequence order that we mention in a run list. With this process we can run multiple recipes but the condition is, they must be only one recipe from one cookbook.

***(chef client calling default recipes from **Zee**-cookbook & **Apache**-cookbook **together**) ***

```
[cookbooks]# chef-client -zr
"recipe[zee-cookbook::default],recipe[Apache-cookbook::default]"
```

Include Recipe: To call recipes/recipe from another recipe with in same cookbook. To run **multiple recipes** from same cookbook. We can run any numbers of recipes with **include command** but all must be from same cookbook. Here including recipes with default recipe in **Zee-Cookbook.**

```
[cookbooks]# vi Zee-cookbook/recipes/default.rb
```

I + Enter then <paste code>

```
Include_recile "ABC-cookbook::ABC-recipe"
Include_recipe "ABC-cookbook::XYZ-recipe"
Include_recipe "ABC-cookbook::123-recipe"        Esc+:wq  (call the chef client)

#chef-client -zr "recipe[Zee-cookbook::default]"
```

Connect workstation to chef server to node using chef-repo, bootstrap

Chef server is going to mediator between the code and cookbooks.

Bootstrapping Attaching a node to chef server, by Bootstrap command, both workstation and node should be in same AZ. Two actions will be done while bootstrapping 1. adding node to chef server 2. installing chef package.

Chef-repo It would be the main directory inside it you have to run any commons, it is also having cookbooks).

Create **chef manage** account by "*manage.chef.io*" and download **starter kit**. Go to download and extract file **chef-repo**, after extracting we got more files inside chef-repo are (.chef ,cookbooks ,gitignore, README.md, roles)

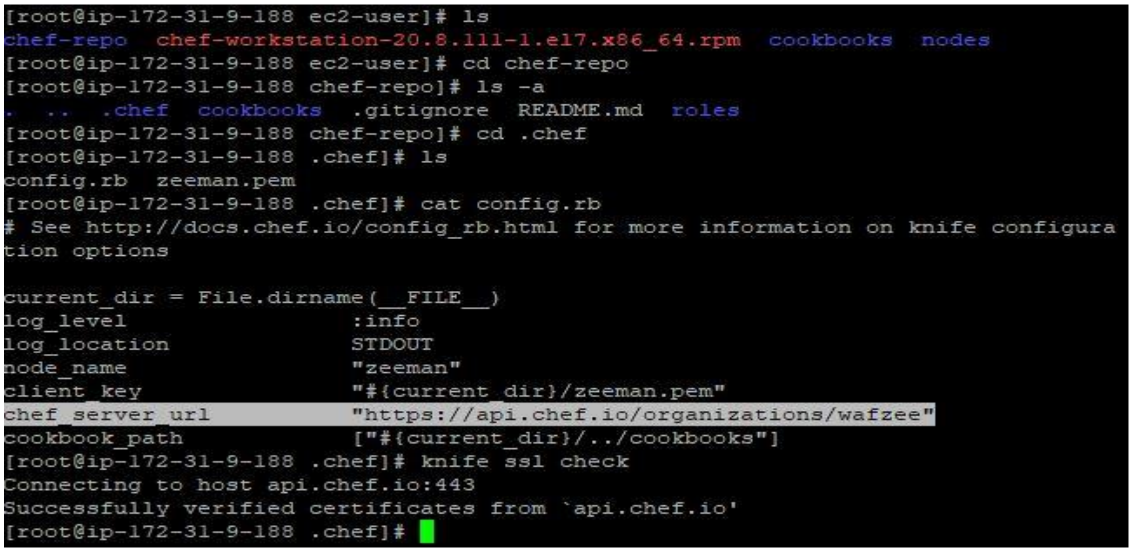
For sending chef-repo file to Linux machine we use the software called **WinSCP.** Drag Chef-repo from left window and drop to right Linux window. (by **ls command** in you can check Chef-repo is showing in your workstation or not)

```
 # Sudo su
 # ls
chef-repo chef-workstation-20.7.96-1.e17.x86_64.rpm cookbooks nodes
 # cd chef-repo
 # ls-a
.  ..  .chef cookbooks .gitignore README.md roles
 # cd .chef/
[.chef]# ls/
 # config.rb    Wafzee.pem                          (organization name is wafzee)
 # cat config.rb
```

Inside config.rb you will get `chef_server_url`   https://api.chef.io/organizations/wafzee

```
 # knife ssl check                    (to check workstation is connected with chef server ?)
```

Connecting to host api.chef.io:443 Successfully verified certificates from 'api.chef.io'



```
[root@ip-172-31-9-188 ec2-user]# ls
chef-repo  chef-workstation-20.8.111-1.el7.x86_64.rpm  cookbooks  nodes
[root@ip-172-31-9-188 ec2-user]# cd chef-repo
[root@ip-172-31-9-188 chef-repo]# ls -a
.  ..  .chef  cookbooks  .gitignore  README.md  roles
[root@ip-172-31-9-188 chef-repo]# cd .chef
[root@ip-172-31-9-188 .chef]# ls
config.rb  zeeman.pem
[root@ip-172-31-9-188 .chef]# cat config.rb
# See http://docs.chef.io/config_rb.html for more information on knife configura
tion options

current_dir = File.dirname(__FILE__)
log_level                :info
log_location             STDOUT
node_name                "zeeman"
client_key               "#{current_dir}/zeeman.pem"
chef_server_url          "https://api.chef.io/organizations/wafzee"
cookbook_path            ["#{current_dir}/../cookbooks"]
[root@ip-172-31-9-188 .chef]# knife ssl check
Connecting to host api.chef.io:443
Successfully verified certificates from `api.chef.io'
[root@ip-172-31-9-188 .chef]#
```

Create Linux machine **(Node1)** same **AZ** of workstation with new security group and new key pair name `node1-key`, save **`Private IP`** for further knife bootstrap commands.   (SSH & HTTPs)

Attach Advance details **`[ #!/bin/bash`**

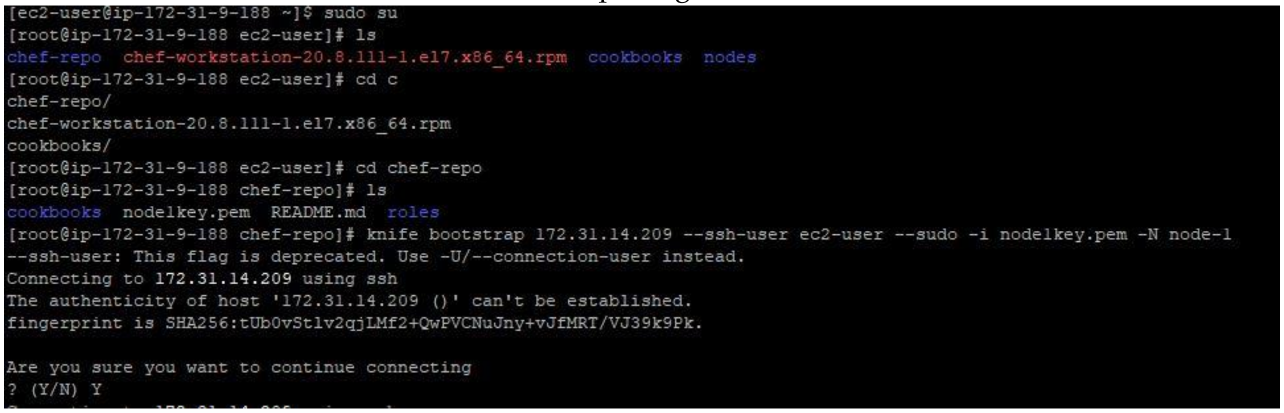
```
Sudo su
Yum update -y]
```

With the help of **WinSCP** please transfer downloaded `node1-key.pem` to Chef-repo for **bootstrap command**

*Now go to **chef workstation** and execute **Bootstrap command** to attach node1 to chef-server.*

`[chef-repo] # knife bootstrap` 172.31.10.120 **`--ssh-user ec2-user --sudo -i node-2key.pem -N node1`**   **(Y** for YES/NO**)**

Node has been **connected to server** and node package has installed



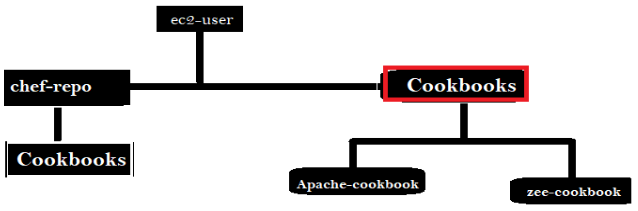
```
[ec2-user@ip-172-31-9-188 ~]$ sudo su
[root@ip-172-31-9-188 ec2-user]# ls
chef-repo  chef-workstation-20.8.111-1.el7.x86_64.rpm  cookbooks  nodes
[root@ip-172-31-9-188 ec2-user]# cd c
chef-repo/
chef-workstation-20.8.111-1.el7.x86_64.rpm
cookbooks/
[root@ip-172-31-9-188 ec2-user]# cd chef-repo
[root@ip-172-31-9-188 chef-repo]# ls
cookbooks  nodelkey.pem  README.md  roles
[root@ip-172-31-9-188 chef-repo]# knife bootstrap 172.31.14.209 --ssh-user ec2-user --sudo -i nodelkey.pem -N node-1
--ssh-user: This flag is deprecated. Use -U/--connection-user instead.
Connecting to 172.31.14.209 using ssh
The authenticity of host '172.31.14.209 ()' can't be established.
fingerprint is SHA256:tUb0vStlv2qjLMf2+QwPVCNuJny+vJfMRT/VJ39k9Pk.

Are you sure you want to continue connecting
? (Y/N) Y
```

(Thank you for installing Chef Infra client chef package)

`[chef-repo]` **`# knife node list`**

`node1` →(showing result **node1** means node1 has been connected to serve

Moving and delete cookbooks in chef-repo to avoid cookbooks confusion:



How move cookbooks to chef-repo's cookbooks

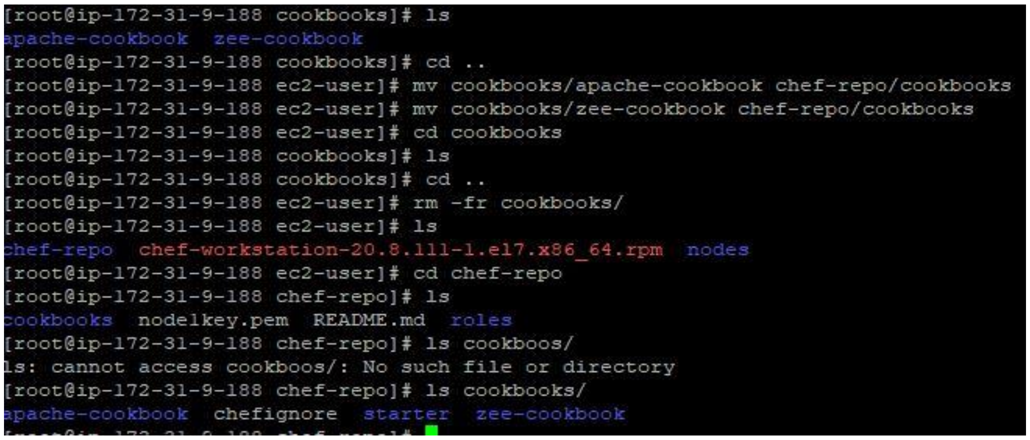
```
# mv cookbooks/apache-cookbook chef-repo/cookbooks
# mv cookbooks/Zee-cookbook chef-repo/cookbooks
# ls
```
didn't get any cookbook, all empty

```
# cd chef-repo
# ls
```
get (cookbooks node1-key.pem README.md roles)

```
# ls cookbooks/
```

```
apache-cookbook chefignore starter zee-cookbook
```
(got inside the chef-repo-cookboook)

It means both cookbooks have been moved to **Chef-repo** **Cookbooooks** from **Cookbooks**



```
[root@ip-172-31-9-188 cookbooks]# ls
apache-cookbook  zee-cookbook
[root@ip-172-31-9-188 cookbooks]# cd ..
[root@ip-172-31-9-188 ec2-user]# mv cookbooks/apache-cookbook chef-repo/cookbooks
[root@ip-172-31-9-188 ec2-user]# mv cookbooks/zee-cookbook chef-repo/cookbooks
[root@ip-172-31-9-188 ec2-user]# cd cookbooks
[root@ip-172-31-9-188 cookbooks]# ls
[root@ip-172-31-9-188 cookbooks]# cd ..
[root@ip-172-31-9-188 ec2-user]# rm -fr cookbooks/
[root@ip-172-31-9-188 ec2-user]# ls
chef-repo  chef-workstation-20.8.111-1.el7.x86_64.rpm  nodes
[root@ip-172-31-9-188 ec2-user]# cd chef-repo
[root@ip-172-31-9-188 chef-repo]# ls
cookbooks  node1key.pem  README.md  roles
[root@ip-172-31-9-188 chef-repo]# ls cookboos/
ls: cannot access cookboos/: No such file or directory
[root@ip-172-31-9-188 chef-repo]# ls cookbooks/
apache-cookbook  chefignore  starter  zee-cookbook
```

Upload apache-cookbook to chef-server:

```
[Chef-repo] # knife cookbook upload apache-cookbook
            # knife cookbook list
```
(confirm uploading?)

```
        apache-cookbook
```

Now we will attach the recipe on node1 which we would like to run on node1, by this command

```
[chef-repo]# knife node run_list set node1 "recipe[apache-cookbook::apache-recipe]
```

```
Node1:
  run_list: recipe[apache-cookbook::apache-recipe]
```

```
[chef-repo] # knife node show node1
```
(get so many info including recipes in run_list)

Now access the Node1

```
# sudo su
# chef-client
```

This chef-client implement the code (inside the recipe) on server Automatically

Now back to **workstation** and **edit the recipe:**

```
[Chef-repo]#vi cookbooks/apache-cookbook/recipes/apache-recipe.rb
```

"Update recipe" `Enter+Esc+:wq`

Upload apache-cookbook to chef-server

```
[chef-repo] # knife cookbook upload apache-cookbook
```

Now go to the node1 and call chef client **# chef-client**

You can see all updated content, also you can ping **node1**'s public IP and see change.

Now see how can we automate this process:

Go to node1

```
[ec2-user] # vi /etc/crontab
* * * * * root chef-client        Esc+:wq      */n (HR DAY MONTH YEAR WEEK)
```

With the help of this command automation will start no need to call the chef client again=2

Chef-client command execute periodically according to "**`*/n * * * * crontab method`**"

Now see full automation:

Create one more linux machine **Node2** *(we also can use existing key of node1 for node2 creation)

Attach Advance details
```
[#!/bin/bash
  Sudo su
  Yum update -y
  echo"* * * * *root chef-client">> etc/crontab]
```

Now back to ***workstation*** *and run Bootstrap command*

```
[chef-repo]# knife bootstrap 172.31.10.120 --ssh-user ec2-user --sudo -i node-
2key.pem -N node2
```

(Y for YES/NO**)**

Node has been connected to server and **node package** has been installed

Now Attach the Recipe to node2 run_list

```
[chef-repo]# knife node run_list set node2 "recipe[apache-cookbook::apache-recipe]"
```

then for check ping the IP of node2 and see webpage.

How to see Delete everything from inside chef-server:

To see list of client present inside chef-server
[chef-repo] #knife client list

To delete clients
knife client delete clientname -y

To see cookbook list
[chef-repo] # knife cookbook list

To delete cookbook
#knife cookbook delete cookbookName -y

To see Role list
[chef-repo] # knife role list

To delete Role
#knife cookbook delete roleName -y

To see Node list
[chef-repo] # knife node list

To delete Node
#knife cookbook delete nodeName -y

How to create ROLE:

```
[chef-repo]# ls
 .chef     roles
           # cd roles/
[roles]    #ls
           #starter.rb
[roles]    # vi Engineer.rb
      Name "Engineer"
           Description "webserver role"
           run_list "recipe[apache-cookbook::apache-recipe]"       ESC+:wq
```

*(this is the command to create role name Engineer)

Now back to chef-repo # **cd ..** and upload the role on chef server

```
[chef-repo] # knife role from file roles/Engineer.rb
```

```
If you want to see the created role
            # knife role list
    o/p:    Engineer
```

Now create 4 instances (1,2,3,4) by one IMA on same availability zone as of workstation with new security group sg-1 with SSH +HTTP.

Attach Advance details

```
[#!/bin/bash
      Sudo su
      Yum update -y
      echo"* * * *root chef-client">> etc/crontab]
```

Now Bootstraps the nodes 1,2,3,4 one by one

```
#[chef-repo]# knife bootstrap 172.31.10.121 --ssh-user ec2-user --sudo -i node-1key.pem -N node1
#[chef-repo]# knife bootstrap 172.31.10.122 --ssh-user ec2-user --sudo -i node-1key.pem -N node2
#[chef-repo]# knife bootstrap 172.31.10.123 --ssh-user ec2-user --sudo -i node-1key.pem -N node3
```

```
#[chef-repo]# knife bootstrap 172.31.10.124 --ssh-user ec2-user --sudo -i node-1key.pem -N node4
```

Now connect these nodes to roles one by one.

```
# knife node run-list set node1 "role[Engineer]"
Node1:
  Run_list:role[Engineer]
```

(similarly for rest 3 nodes)

```
# knife node run-list set node2 "role[Engineer]"
# knife node run-list set node3 "role[Engineer]"
# knife node run-list set node4 "role[Engineer]"
```

UPLOAD cookbook to server

```
# knife cookbook upload apache-cookbook
```

Now we can check public IP of any node on webserver, every node will behave like server cause, now cookbook has been uploaded despite of uploading different recipes, all recipes have uploaded together inside role by cookbok.

Now we are doing changes in recipe

```
# vi cookbooks/apache-cookbook/recipes/apache-recipe.rb
Content change to "I Love my India"                    ESC+:wq
```

Now see if Boss need changes, said do work on another recipe (recipe10)

```
#cat cookbooks/apache-cookbook/recipes/recipe10.rb
```

Paste code update recipe and go to the role in workstation

```
# vi roles/Engineer.rb
```

```
 vi Engineer.rb
Name "Engineer"
 Description "webserver role"
run_list "recipe[apache-cookbook::apache-recipe]"
run_list "recipe[apache-cookbook::recipe10]"
```

update apache-recipe to **recipe10** in role

ESC+:wq

*for update in recipe we can create user and file by these commands below

```
#user"zee"
 #file "shanfile"
```

now upload role to server

```
[chef-repo] # knife role from file roles/Engineer.rb
```

Again, go to the workstation

```
# vi roles/Engineer.rb
```

```
Name "Engineer"
  Description "webserver role"
  run_list "recipe[apache-cookbook]"
```

(change last line only apache-cookbook in role)

ESC+:wq

now upload role to server

```
[chef-repo] # knife role from file roles/Engineer.rb
```

Do not mention any recipe just upload only cookbook for all recipes, will update automatically on server

```
# # knife cookbook upload apache-cookboo
```

Now we are adding 2 cookbooks in roles

```
vi roles/Engineer.rb
```

```
  Name "Engineer"
  Description "webserver role"
  run_list "recipe[apache-cookbook]","recipe[zee-cookbook]"            esc+:wq
```

now upload role to server

```
[chef-repo] # knife role from file roles/Engineer.rb
```

Do not forget to upload zee-cookbook on server otherwise role will not perform properly

```
# knife cookbook upload zee-cookbook
```

Boss need changes again but this time in zee-recipe

```
Chef-repo]# vi cookbooks/apache-cookbook/recipes/zee-recipe.rb
```

```
 %W (httpd mariadb-server unzip git vim) .each do |p|
Package p do
Action :install
end
end                                                        esc+:wq
# knife cookbook upload zee-cookbook
```

Go to inside any node and search git by using command

```
# which git
```

after 1 minute execute again same command and you will see output

```
/bin/git
```

it means working properly

- It is an advance version of virtualization. It Design to create, deploy and run application. Docker Engine runs natively on Linux distributions,
- Docker uses container on the host OS to run applications. It allows applications to use the same Linux kernel as a system on the host computer, rather than creating a whole virtual OS. Docker written in GO language.
- The tool performs OS level virtualization also known as containerization.
- Docker is a set of PAAS that uses OS level virtualization whereas VMware uses hardware level virtualization

Advantages: layered file system, no pre-allocation of RAM, light weight
CI-efficiency: Docker enables you to build container image and use the same image across every step of the deployment process.
Disadvantages: Difficult to manage large number of containers. Cross platform compatibility not possible. Docker is suitable when deployment OS and testing O S are same. No solutions for data recovery and backups. Not good for rich GUI.

Architecture: Docker-Client Docker-Engine Server-Daemen
Docker-Hub Docker-image Docker-Compose

Image: Docker image are the read only binary templates used to create containers, or single file with all dependencies and configuration required to run a program.

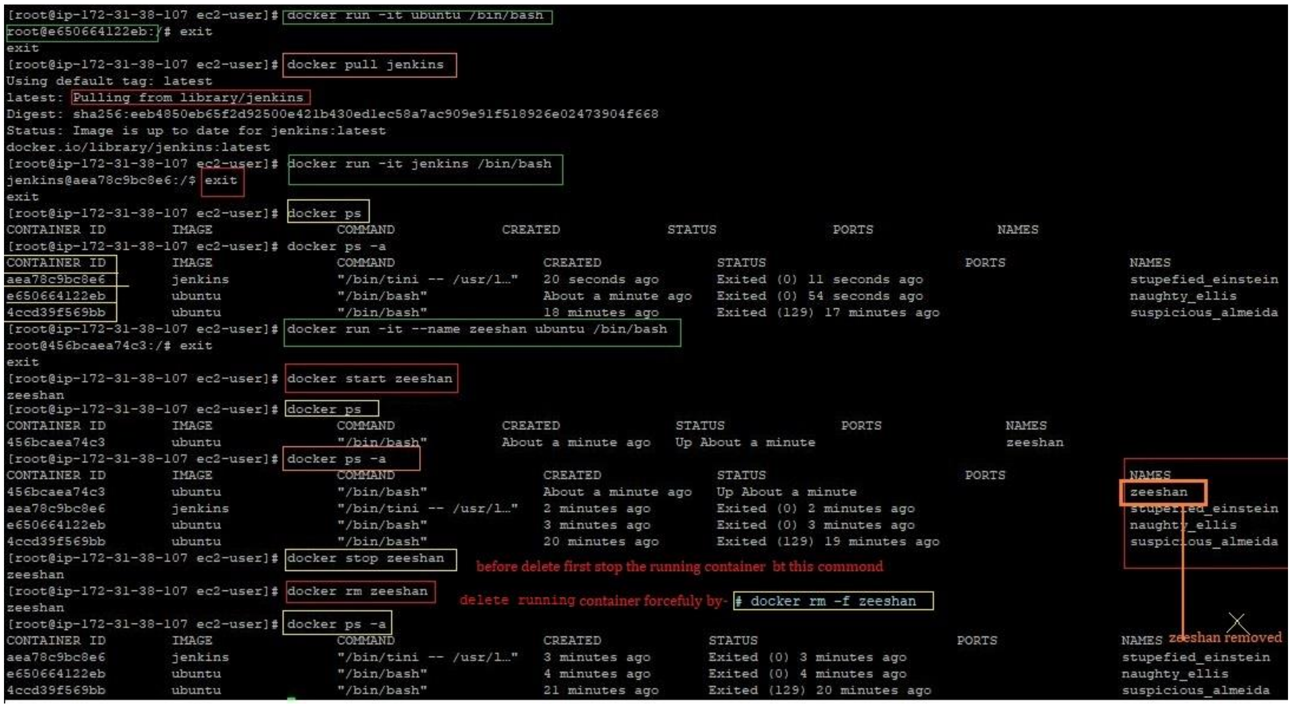
Container: It holds the entire packages that needed to run the application.

Basic Docker Commands:

[ ]	# yum install docker -Y	install docker
[ ]	# remove docker -Y	uninstall docker
[ ]	# docker images	See all images present in your local
[ ]	# docker **search** Jenkins	To find out images in Docker hub
[ ]	# docker **pull** Jenkins	To download image from docker hub to local machines
[ ]	#***systemctl*** start docker	To start docker service on terminal
[ ]	# **service** docker **start**	to start docker functioning
[ ]	# **docker run -it ubuntu /bin/bash**	To create a **container**
[ ]	# docker run **-it** **--name Zeeshan** ubuntu **/bin/bash**	To give name to container
[ ]	#docker **container** run **-it --name Zeeshan -p 8000:80** ubuntu /bin/bash	
[ ]	# service docker **status/** (docker info)	To check service is **start or not**
[ ]	# docker **start** container	To start container
[ ]	# docker **attach** container	To go **inside** container
[ ]	# docker ps **-a**	To **see all** containers
[ ]	# docker ps (PS=Process Status)	to see **only running** containers

```
[      ] # docker network inspect Zeeshan          To check the network status inside container    [
] # docker run -d Zeeshan                        To running a container in the background
[      ] # docker stop Zeeshan                              always stop container before delete
[      ] # docker rm Zeeshan                                 to remove container
[      ] # docker rm -f Zeeshan                                  To remove running containers
[      ] # docker container prune                                  To remove all containers
```

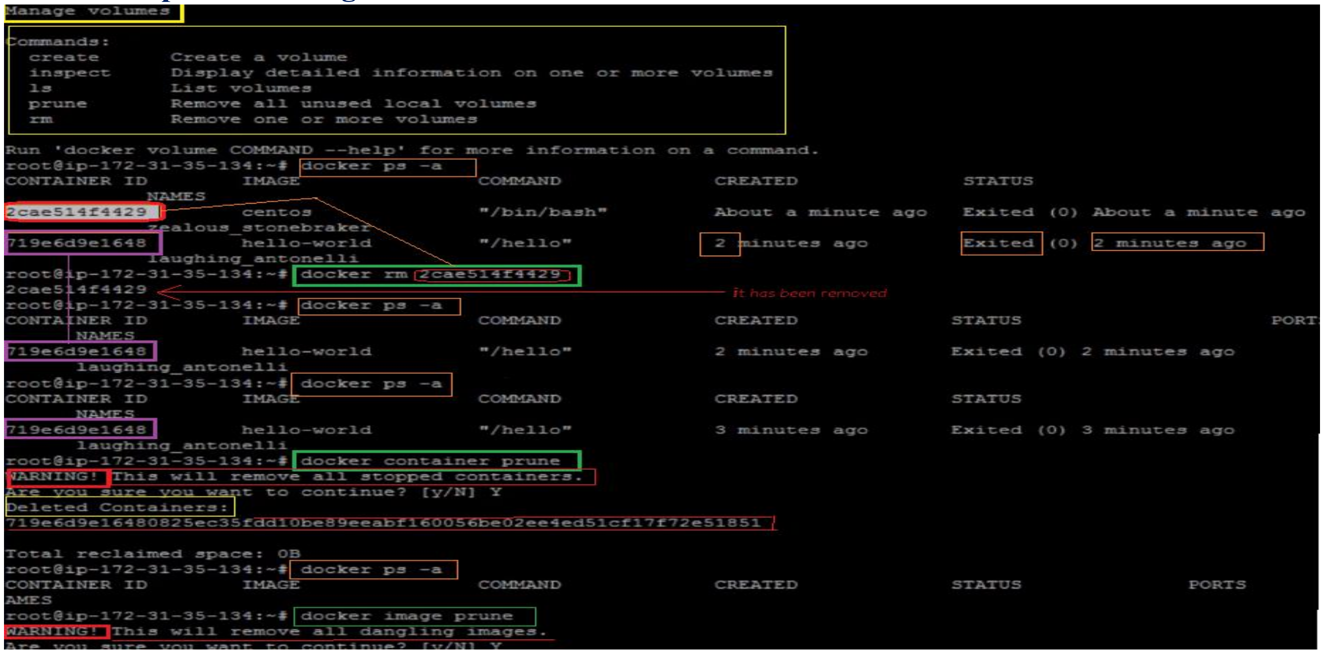
Docker Installation by ubuntu image



```
[root@ip-172-31-38-107 ec2-user]# docker run -it ubuntu /bin/bash
root@e650664122eb:/# exit
exit
[root@ip-172-31-38-107 ec2-user]# docker pull jenkins
Using default tag: latest
latest: Pulling from library/jenkins
Digest: sha256:eeb4850eb65f2d92500e421b430ed1ec58a7ac909e91f518926e02473904f668
Status: Image is up to date for jenkins:latest
docker.io/library/jenkins:latest
[root@ip-172-31-38-107 ec2-user]# docker run -it jenkins /bin/bash
jenkins@aea78c9bc8e6:/$ exit
exit
[root@ip-172-31-38-107 ec2-user]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@ip-172-31-38-107 ec2-user]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS                        PORTS               NAMES
aea78c9bc8e6        jenkins             "/bin/tini -- /usr/l…"   20 seconds ago       Exited (0) 11 seconds ago                         stupefied_einstein
e650664122eb        ubuntu              "/bin/bash"              About a minute ago   Exited (0) 54 seconds ago                         naughty_ellis
4ccd39f569bb        ubuntu              "/bin/bash"              18 minutes ago       Exited (129) 17 minutes ago                       suspicious_almeida
[root@ip-172-31-38-107 ec2-user]# docker run -it --name zeeshan ubuntu /bin/bash
root@456bcaea74c3:/# exit
exit
[root@ip-172-31-38-107 ec2-user]# docker start zeeshan
zeeshan
[root@ip-172-31-38-107 ec2-user]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED              STATUS              PORTS               NAMES
456bcaea74c3        ubuntu              "/bin/bash"         About a minute ago   Up About a minute                       zeeshan
[root@ip-172-31-38-107 ec2-user]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS                        PORTS               NAMES
456bcaea74c3        ubuntu              "/bin/bash"              About a minute ago   Up About a minute                                 zeeshan
aea78c9bc8e6        jenkins             "/bin/tini -- /usr/l…"   2 minutes ago        Exited (0) 2 minutes ago                          stupefied_einstein
e650664122eb        ubuntu              "/bin/bash"              3 minutes ago        Exited (0) 3 minutes ago                          naughty_ellis
4ccd39f569bb        ubuntu              "/bin/bash"              20 minutes ago       Exited (129) 19 minutes ago                       suspicious_almeida
[root@ip-172-31-38-107 ec2-user]# docker stop zeeshan
zeeshan
[root@ip-172-31-38-107 ec2-user]# docker rm zeeshan
zeeshan
[root@ip-172-31-38-107 ec2-user]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                        PORTS               NAMES
aea78c9bc8e6        jenkins             "/bin/tini -- /usr/l…"   3 minutes ago       Exited (0) 3 minutes ago                          stupefied_einstein
e650664122eb        ubuntu              "/bin/bash"              4 minutes ago       Exited (0) 4 minutes ago                          naughty_ellis
4ccd39f569bb        ubuntu              "/bin/bash"              21 minutes ago      Exited (129) 20 minutes ago                       suspicious_almeida
```

before delete first stop the running container bt this commond
delete running container forcefuly by- # docker rm -f zeeshan
zeeshan removed

Remove stop and running containers



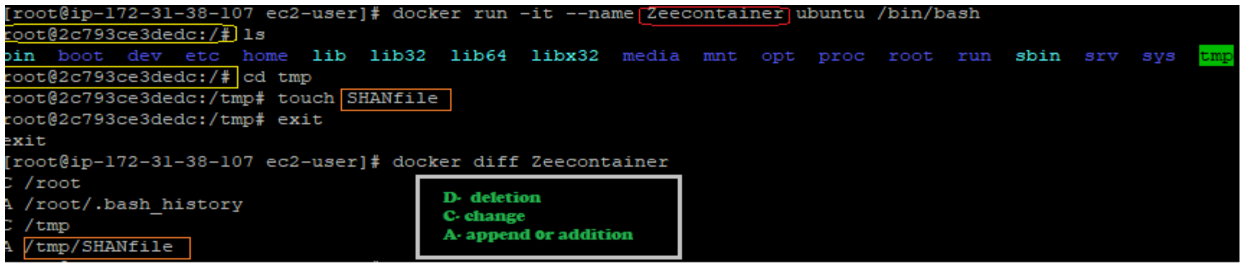
```
Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
root@ip-172-31-35-134:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED              STATUS
          NAMES
2cae514f4429        centos              "/bin/bash"         About a minute ago   Exited (0) About a minute ago
          zealous_stonebraker
719e6d9e1648        hello-world         "/hello"            2 minutes ago        Exited (0) 2 minutes ago
          laughing_antonelli
root@ip-172-31-35-134:~# docker rm 2cae514f4429
2cae514f4429
root@ip-172-31-35-134:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                     PORT
     NAMES
719e6d9e1648        hello-world         "/hello"            2 minutes ago       Exited (0) 2 minutes ago
     laughing_antonelli
root@ip-172-31-35-134:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
     NAMES
719e6d9e1648        hello-world         "/hello"            3 minutes ago       Exited (0) 3 minutes ago
     laughing_antonelli
root@ip-172-31-35-134:~# docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] Y
Deleted Containers:
719e6d9e16480825ec35fdd10be89eeabf160056be02ee4ed51cf17f72e51851

Total reclaimed space: 0B
root@ip-172-31-35-134:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
AMES
root@ip-172-31-35-134:~# docker image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] Y
```

Create file inside container

```
[                ]# docker run -it --name Zeecontainer ubuntu /bin/bash
root@2d793ce3dd:/#ls
                 #cd tmp
                 #touch SHANfile                    (create file inside temp
directory)
                 #exit
```

If you want to see the difference between base image and changes on it use **diff** command then.

```
[                ]# docker diff Zeecontainer

C /root                                      D=detection C=Change A=Append
A /root/.bash_history
C /tmp
A /tmp/SHANfile                   (we can see the changes-file created inside
root)
```



```
[root@ip-172-31-38-107 ec2-user]# docker run -it --name Zeecontainer ubuntu /bin/bash
root@2c793ce3dedc:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp
root@2c793ce3dedc:/# cd tmp
root@2c793ce3dedc:/tmp# touch SHANfile
root@2c793ce3dedc:/tmp# exit
exit
[root@ip-172-31-38-107 ec2-user]# docker diff Zeecontainer
C /root
A /root/.bash_history
C /tmp
A /tmp/SHANfile
```

D- deletion
C- change
A- append or addition

Create image from container

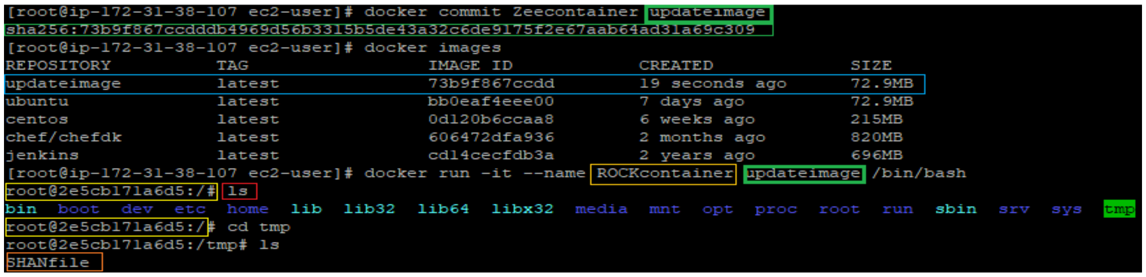
```
[      ]# docker commit ZEEcontainer updateimage

Sha256:hh33h4hh47shdudu79fkfk954low7gd56sv04k5757jrjr74urjjr4    ←updateimage

[      ]# docker images
```

we got so many images ubuntu Jenkins chef & centOS also updateimage,

Create new container by image(updateimage) created by other container

```
[      ]# docker run -it --name ROCKcontainer updateimage /bin/bash
root@2e5cb171a6d5:/# ls
                    # cd tmp/
                    # ls
                    # SHANfile
```

you will get all files back inside new container because it is created by old image.



```
[root@ip-172-31-38-107 ec2-user]# docker commit Zeecontainer updateimage
sha256:73b9f867ccdddb4969d56b3315b5de43a32c6de9175f2e67aab64ad31a69c309
[root@ip-172-31-38-107 ec2-user]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
updateimage         latest              73b9f867ccdd        19 seconds ago      72.9MB
ubuntu              latest              bb0eaf4eee00        7 days ago          72.9MB
centos              latest              0d120b6ccaa8        6 weeks ago         215MB
chef/chefdk         latest              606472dfa936        2 months ago        820MB
jenkins             latest              cd14cecfdb3a        2 years ago         696MB
[root@ip-172-31-38-107 ec2-user]# docker run -it --name ROCKcontainer updateimage /bin/bash
root@2e5cb171a6d5:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp
root@2e5cb171a6d5:/# cd tmp
root@2e5cb171a6d5:/tmp# ls
SHANfile
```

Docker file creation - steps:

1. Create a file name **D**ockerfile *remember D always capital letter
2. Add instructions inside **D**ockerfile
3. Build **D**ockerfile to create an image
4. Run image to create container.

Step 1 use command **# vi Dockerfile**

Step 2 use command go inside the **D**ockerfile

press i

```
# FROM ubuntu
# RUN echo "Love the Zeeman show" > /tmp/starfile
```

Esc:wq!

Step 3 to create image out of docker file

```
# docker build -t Shanu .
```

[.] > all stuff present inside the dockerfile build into this ***new image***

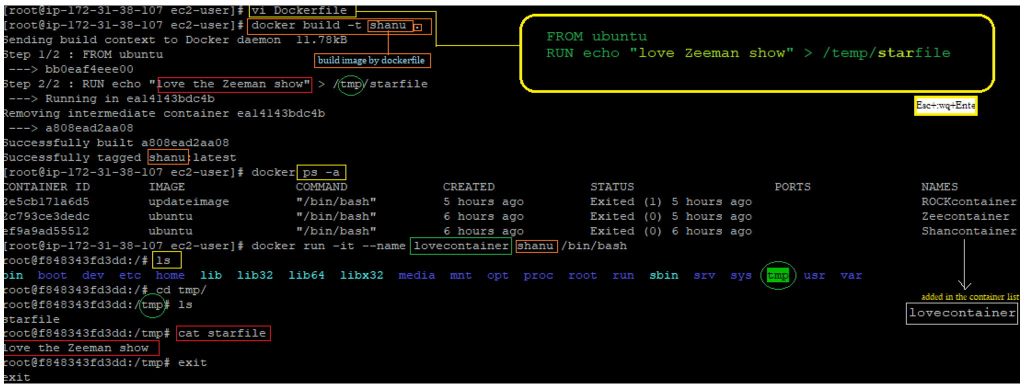
Step 4 now create container my newly created image (Shanu)

```
[    ]# docker run -it --name lovecontainer Shanu /bin/bash
root@2e5cb171a6d5:/# ls
root@2e5cb171a6d5:/# cd tmp/
root@2e5cb171a6d5:/tmp# ls
```

you will get starfile, use cat command to see inside the starfile

```
root@2e5cb171a6d5:/tmp# cat /starfile
```

love the Zeeman show **# exit**



```
[root@ip-172-31-38-107 ec2-user]# vi Dockerfile
[root@ip-172-31-38-107 ec2-user]# docker build -t shanu .
Sending build context to Docker daemon  11.78kB
Step 1/2 : FROM ubuntu
 ---> bb0eaf4eee00
Step 2/2 : RUN echo "love the Zeeman show" > /tmp/starfile
 ---> Running in ea14143bdc4b
Removing intermediate container ea14143bdc4b
 ---> a808ead2aa08
Successfully built a808ead2aa08
Successfully tagged shanu:latest
[root@ip-172-31-38-107 ec2-user]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                   PORTS               NAMES
2e5cb171a6d5        updateimage         "/bin/bash"         5 hours ago         Exited (1) 5 hours ago                       ROCKcontainer
2c793ce3dedc        ubuntu              "/bin/bash"         6 hours ago         Exited (0) 5 hours ago                       Zeecontainer
ef9a9ad55512        ubuntu              "/bin/bash"         6 hours ago         Exited (0) 6 hours ago                       Shancontainer
[root@ip-172-31-38-107 ec2-user]# docker run -it --name lovecontainer shanu /bin/bash
root@f848343fd3dd:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@f848343fd3dd:/# cd tmp/
root@f848343fd3dd:/tmp# ls
starfile
root@f848343fd3dd:/tmp# cat starfile
love the Zeeman show
root@f848343fd3dd:/tmp# exit
exit
```

```
FROM ubuntu
RUN echo "love Zeeman show" > /temp/starfile
```

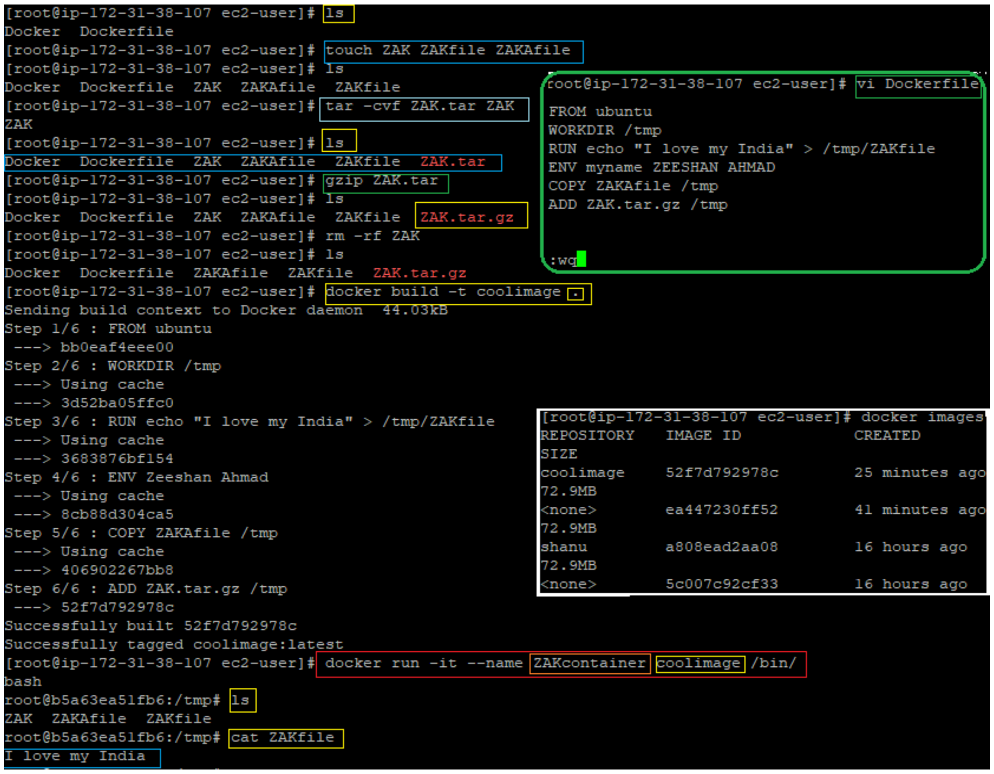
Dockerfile: - It is basically a test file it contains some set of instructions
Automation of Docker image creation

Docker Components: FROM, RUN, WORKDIR, MAINTAINER, COPY
ADD, ENV, EXPOSE, CMD ENTRYPOINT

Means execution of different type of files inside the Dockerfile

Create new files by touch command and make zip and unzip using tar commands

```
# tar -cvf ZAK.tar Zak                    # gzip ZAK.tar
```



```
[root@ip-172-31-38-107 ec2-user]# ls
Docker  Dockerfile
[root@ip-172-31-38-107 ec2-user]# touch ZAK ZAKfile ZAKAfile
[root@ip-172-31-38-107 ec2-user]# ls
Docker  Dockerfile  ZAK  ZAKAfile  ZAKfile
[root@ip-172-31-38-107 ec2-user]# tar -cvf ZAK.tar ZAK
ZAK
[root@ip-172-31-38-107 ec2-user]# ls
Docker  Dockerfile  ZAK  ZAKAfile  ZAKfile  ZAK.tar
[root@ip-172-31-38-107 ec2-user]# gzip ZAK.tar
[root@ip-172-31-38-107 ec2-user]# ls
Docker  Dockerfile  ZAK  ZAKAfile  ZAKfile  ZAK.tar.gz
[root@ip-172-31-38-107 ec2-user]# rm -rf ZAK
[root@ip-172-31-38-107 ec2-user]# ls
Docker  Dockerfile  ZAKAfile  ZAKfile  ZAK.tar.gz
[root@ip-172-31-38-107 ec2-user]# docker build -t coolimage .
Sending build context to Docker daemon  44.03kB
Step 1/6 : FROM ubuntu
 ---> bb0eaf4eee00
Step 2/6 : WORKDIR /tmp
 ---> Using cache
 ---> 3d52ba05ffc0
Step 3/6 : RUN echo "I love my India" > /tmp/ZAKfile
 ---> Using cache
 ---> 3683876bf154
Step 4/6 : ENV Zeeshan Ahmad
 ---> Using cache
 ---> 8cb88d304ca5
Step 5/6 : COPY ZAKAfile /tmp
 ---> Using cache
 ---> 406902267bb8
Step 6/6 : ADD ZAK.tar.gz /tmp
 ---> 52f7d792978c
Successfully built 52f7d792978c
Successfully tagged coolimage:latest
[root@ip-172-31-38-107 ec2-user]# docker run -it --name ZAKcontainer coolimage /bin/
bash
root@b5a63ea51fb6:/tmp# ls
ZAK  ZAKAfile  ZAKfile
root@b5a63ea51fb6:/tmp# cat ZAKfile
I love my India
```

```
[root@ip-172-31-38-107 ec2-user]# vi Dockerfile

FROM ubuntu
WORKDIR /tmp
RUN echo "I love my India" > /tmp/ZAKfile
ENV myname ZEESHAN AHMAD
COPY ZAKAfile /tmp
ADD ZAK.tar.gz /tmp


:wq
```

```
[root@ip-172-31-38-107 ec2-user]# docker images
REPOSITORY    IMAGE ID            CREATED
SIZE
coolimage     52f7d792978c        25 minutes ago
72.9MB
<none>        ea447230ff52        41 minutes ago
72.9MB
shanu         a808ead2aa08        16 hours ago
72.9MB
<none>        5c007c92cf33        16 hours ago
```

Docker Volumes: uses of docker volumes-

- Decoupling container from storage
- Share volume among different containers
- Attach volume to containers
- On deleting Container, Volume doesn't delete.

Create volume from Dockerfile: (Method 1)

Create a Dockerfile **# vi DockerfileZ**

```
FROM ubuntu
      VOLUME ["/myvolume"]                                Esc:Wq
```

Then create image from is **DockerfileZ**

```
# Docker build -t superimage .
```

- **Now create a container from this image.**

```
# docker run -it --name containerZ superimage /bin/bash
```

```
containerZ:/#ls
```
you can see so many files including myvolume

Go inside myvolume and create files Amar Akbar Anthony by touch commands

- **Now share this volume (*myvolume*) with another container**

```
# docker run -it --name containerZS --privilleged=true --volumes-from
containerZ ubuntu /bin/bash
ContainerZS:/#
```

After creating **ContainerZS** **myvolume** is visible inside it; whatever you do in one volume you can see from other volumes.

```
:/#cd myvolume        myvolume# ls
```
you will get Amar Akbar Anthony

Create volume using command: (Method 2)

```
# docker run -it --name containerM -v /volumeX ubuntu /bin/bash
```

Create 3 files (fileX) (fileY) (fileZ) inside volumeX.

- Make new container >> **ContainerT** by using the Volume of **ContainerM**.

```
# docker run -it --name ContainerT --privileged=true --volumeX-from
ContainerM ubuntu /bin/bash
```

```
containerM:/#ls                # cd volumeX/              # ls

you will get fileX fileY fileZ.
```

Volume (Host-container) mapping:

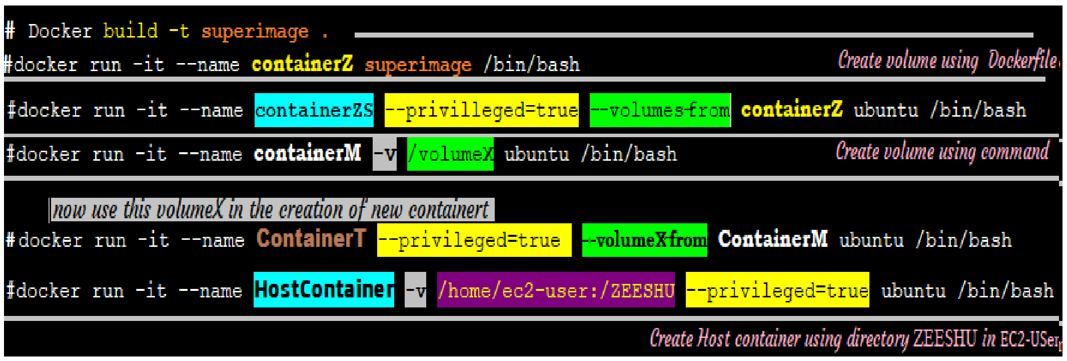
```
#cd ec2-user
# ls            >>                    # Dockerfile  file1   file2  file 3
Create host container

# docker run -it --name HostContainer -v /home/ec2-user:/ZEESHU -
-privileged=true ubuntu /bin/bash

HostContainer:/# ls                >>             # cd ZEESHU (Directory)
# Dockerfile file1 file2 file3
```

Mapping done between host and (myfile)**ZEESHU** (Directory)

```
# Cd ZEESHU                       # touch file444 file222 file333
# exit
# Cd ec2-user                      # ls
```

You can see same files inside host machine ***ec2-user***

```
# Dockerfile file1 file2 file3 file444 file333 file222
```



```
# Docker build -t superimage .
#docker run -it --name containerZ superimage /bin/bash
```

Create volume using Dockerfile

```
#docker run -it --name containerZS --privileged=true --volumes-from containerZ ubuntu /bin/bash
#docker run -it --name containerM -v /volumeX ubuntu /bin/bash
```

Create volume using command

now use this volumeX in the creation of new containert

```
#docker run -it --name ContainerT --privileged=true --volumeXfrom ContainerM ubuntu /bin/bash
#docker run -it --name HostContainer -v /home/ec2-user:/ZEESHU --privileged=true ubuntu /bin/bash
```

Create Host container using directory ZEESHU in EC2-USer

Docker exec :- It creates a new process in the container's environment, specially used for running new things in an already started container be in a shell or some other process.
Docker Attach: - It just connected to the standard input/output of the main process inside the container to corresponding standard input/output error of current terminal.

Expose:- When you expose a port the service inside a container will not be accessible from outside, but accessible from inside other container, it's good for inter-communication container
Publish:- If you do publish **-p**(used for port mapping) but do but do not Expose, docker doesn't implicit expose, If a port is open to the public it is automatically open to the other containers.

When you Expose and **-p** a port the service in the container is accessible from anywhere even outside docker container.

Three options are: -

*[**-p includes Expose**(open)]*

1. *Neither **Specify** nor **-p***
2. *Only **Specify** not **-p***
3. *Both **Specify** and **-p***

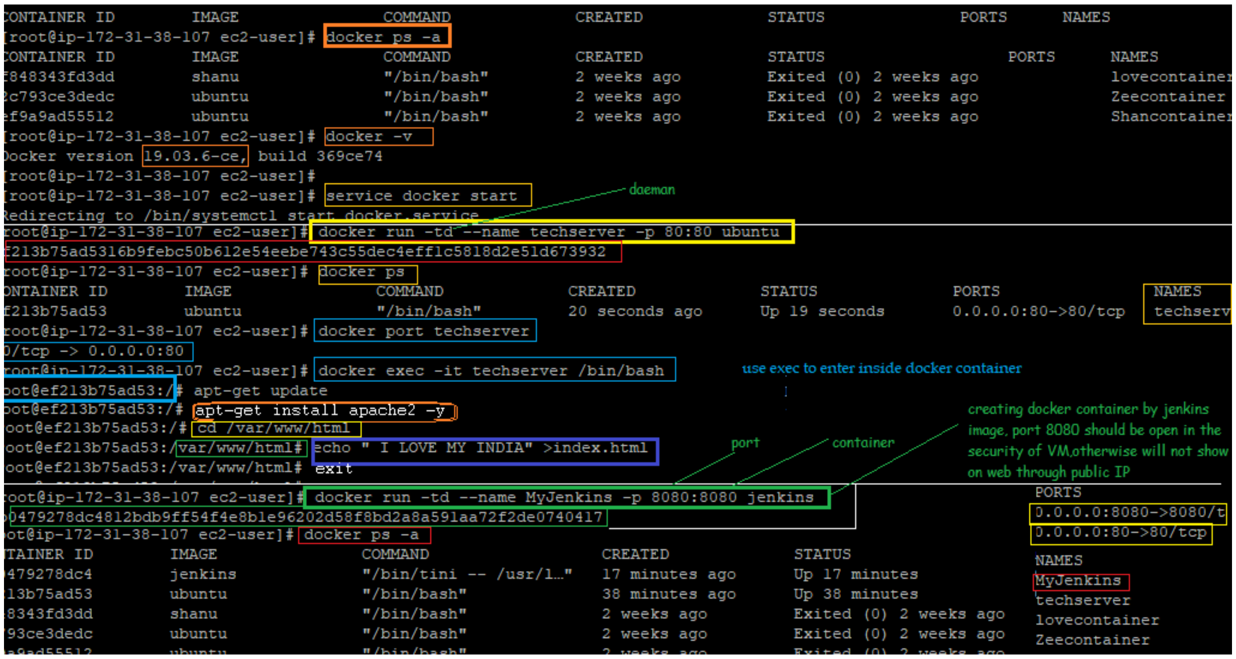
```
# yum install docker
# service docker start
# docker run -td -name techserver -p 80:80 ubuntu
```
(80 for port 80 for container)

```
Eh6ebdf6jf9fmf7rmf8tr9r0fkf8mfd8md7jd7d7dyupo09wtldu
# docker port techserver
  80/Tcp  ------> 0.0.0.0 /80
# docker exec -it techserver
# apt-get update
# apt-get install apache2-y
# cd /var/www/html
# var/www/html # echo "I love my India" >index.html
# exit
# service apache2 start
```

Now you can put public IP on browser can see easily "I love my India" which was deployed on apache server.
Same thing you can do with Jenkins by using port 8080:8080 and publish -p

```
#docker run -td -name Myjenkins -p 8080:8080 jenkins
Eh6ebdf6jf9fmf7rmf8tr9r0fkf8mfd8md7jd7d7dyupo09wtldu
```
d=daemon

Before pasting public Ip of Jenkins container please enable 8080:8080 port inside the security of you virtual Machine, Now you can see Jenkins website on Brower.


daeman
use exec to enter inside docker container
creating docker container by jenkins image, port 8080 should be open in the security of VM,otherwise will not show on web through public IP
port
container

```
ONTAINER ID         IMAGE               COMMAND             CREATED             STATUS              PORTS     NAMES
[root@ip-172-31-38-107 ec2-user]# docker ps -a
ONTAINER ID         IMAGE               COMMAND             CREATED             STATUS                   PORTS     NAMES
848343fd3dd         shanu               "/bin/bash"         2 weeks ago         Exited (0) 2 weeks ago             lovecontaine
c793ce3dedc         ubuntu              "/bin/bash"         2 weeks ago         Exited (0) 2 weeks ago             Zeecontainer
f9a9ad55512         ubuntu              "/bin/bash"         2 weeks ago         Exited (0) 2 weeks ago             Shancontaine
[root@ip-172-31-38-107 ec2-user]# docker -v
ocker version 19.03.6-ce, build 369ce74
[root@ip-172-31-38-107 ec2-user]#
[root@ip-172-31-38-107 ec2-user]# service docker start
edirecting to /bin/systemctl start docker.service
root@ip-172-31-38-107 ec2-user]# docker run -td --name techserver -p 80:80 ubuntu
f213b75ad5316b9febc50b612e54eebe743c55dec4eff1c5818d2e51d673932
root@ip-172-31-38-107 ec2-user]# docker ps
ONTAINER ID         IMAGE               COMMAND             CREATED             STATUS              PORTS                NAMES
f213b75ad53         ubuntu              "/bin/bash"         20 seconds ago      Up 19 seconds       0.0.0.0:80->80/tcp   techserv
root@ip-172-31-38-107 ec2-user]# docker port techserver
0/tcp -> 0.0.0.0:80
root@ip-172-31-38-107 ec2-user]# docker exec -it techserver /bin/bash
oot@ef213b75ad53:/# apt-get update
oot@ef213b75ad53:/# apt-get install apache2 -y
oot@ef213b75ad53:/# cd /var/www/html
oot@ef213b75ad53:/var/www/html# echo " I LOVE MY INDIA" >index.html
oot@ef213b75ad53:/var/www/html# exit
oot@ip-172-31-38-107 ec2-user]# docker run -td --name MyJenkins -p 8080:8080 jenkins
0479278dc4812bdb9ff54f4e8b1e96202d58f8bd2a8a591aa72f2de0740417
ot@ip-172-31-38-107 ec2-user]# docker ps -a
TAINER ID         IMAGE            COMMAND                  CREATED            STATUS                   PORTS
479278dc4         jenkins          "/bin/tini -- /usr/l…"   17 minutes ago     Up 17 minutes            0.0.0.0:8080->8080/t
13b75ad53         ubuntu           "/bin/bash"              38 minutes ago     Up 38 minutes            0.0.0.0:80->80/tcp
8343fd3dd         shanu            "/bin/bash"              2 weeks ago        Exited (0) 2 weeks ago
93ce3dedc         ubuntu           "/bin/bash"              2 weeks ago        Exited (0) 2 weeks ago
                                                                                                        NAMES
                                                                                                        MyJenkins
                                                                                                        techserver
                                                                                                        lovecontainer
                                                                                                        Zeecontainer
```

Docker hub Explanation: push and pull images

```
# Service docker start
# docker run -it ubuntu bin/bash
```

Automatic generated container name **interesting _bond.**
Create some files inside this container inside tmp file, by touch fileshan, fileZan ,filekhan.
Using commit command create new image by this **interesting _bond** container.

```
# docker commit interesting _bond image1
```

Now create docker hub account by hub.docker.com and go to ec2 host machine and login username pass

`# docker login` insert username(zshan227) and password **[username =dockerID]**

`# docker tag image1` **zshan227/project1** Now give tag to your image **[project1 = newimage]**

`# docker` **push** `zshan227/project1` push docker id/new image to docker hub

Now go to the docker hub account see repositories we will get this image `(project1)` with docker id.
Now create one instance from Tokyo reason and pull this image from docker hub.

```
# service docker start
# docker pull zshan227/project1
# docker images
Zshan227/project1
```

Pull new image from docker hub

only one image showing

Now create new container by using this image docker id /new image.

```
# docker run -it -name mycontainer zshan227/project1 /bin/bash
Hdyh947846rhfhf7rhdhdeh:/#  ls
```

When you check inside this mycontainer You will get so many files including tmp file.

```
# Hdyh947846rhfhf7rhdhdeh:/# cd tmp/
```
go inside tmp ,you will get files was created in mumbai region

(fileshan, fileZan ,filekhan)

Make private your project by setting of docker hub account. After making private it denied access, and required to login docker if wanted to pull any image.

Some more important commands:

```
# docker stop $(docker ps -a -q)
```
stop all running containers

```
# docker rm $(docker ps -a -q)
```
delete all stop containers

```
# docker rmi -f $(docker image -q)
```
delete all images

"$ sign used as a script"

Ansible Server: The machine where Ansible is installed and from which all task and playbook will be run. **Host:** Nodes, which are automated by Ansible

Module: Basically, it is a command or set of similar commands meant to be executed on the client-side.

Role: A way of organizing tasks and related files, to be later called playbook.

Fact: Information fetched from the client system from the global variables with the gather-facts operation.

Inventory: File containing data about the Ansible client servers.

Notifier: Section attributed to a task which calls a handler if the output is changed.

Handler: task which is called only if a notifier is present.

Playbook: It consist code in YAML format, which describes task to be executed.

Create 3 EC2 Instances on same availability zone, Ansible Server, Node 1and node 2.

With Advanced detail #!/bin/bash

```
sudo su
yum update-y
```

Go to inside Ansible server and download Ansible Package

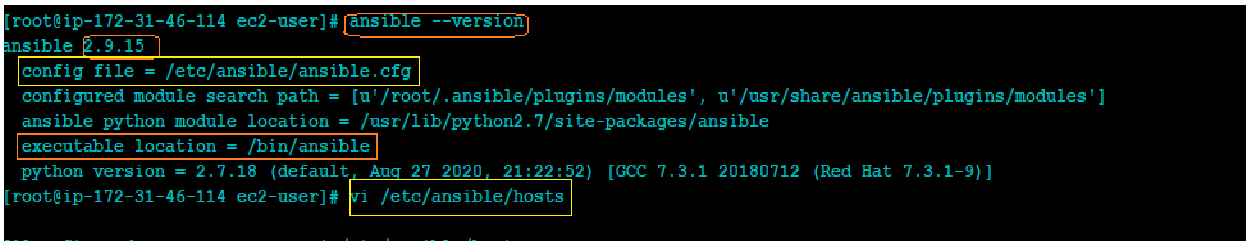
```
[] # sudo su
[] # wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
[] # ls
[] # epel-releas-latest-7.noarch.rpm
[] # yum install epel-releas-latest-7.noarch.rpm
[] # yum update -y
```

** epel= extra package for enterprises Linux*

Now we have install all the packages one by one

```
# yum install git python python-level python-pip openssl ansible -y
```

```
[] # ansible --version
ansible 2.9.25
```



```
[root@ip-172-31-46-114 ec2-user]# ansible --version
ansible 2.9.15
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /bin/ansible
  python version = 2.7.18 (default, Aug 27 2020, 21:22:52) [GCC 7.3.1 20180712 (Red Hat 7.3.1-9)]
[root@ip-172-31-46-114 ec2-user]# vi /etc/ansible/hosts
```

For any kind of update of packages or files we created group first then update the GROUP individually.

```
[] # vi /etc/ansible/hosts
```

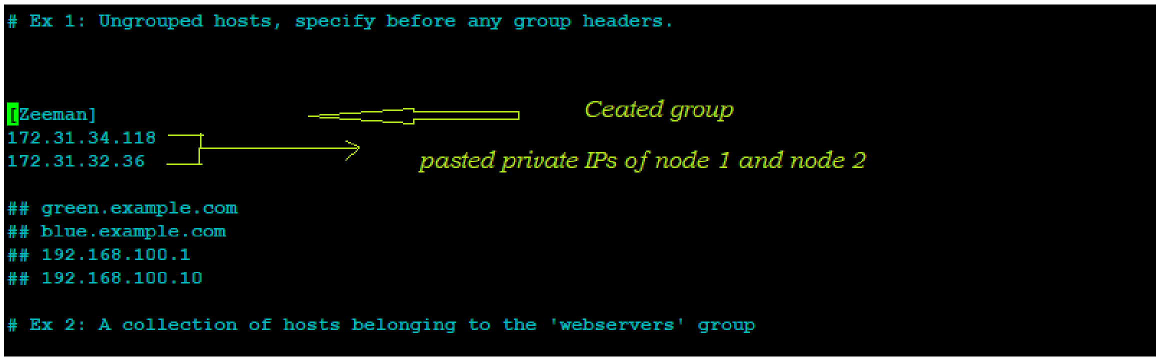
Press I to insert

Go to… **Ex 1: ungrouped hosts**

Please create a group by name Zeeman (zeko) and paste the Pvt. IP address of Node 1 & Node 2

```
[Zeeman]
172.31.34.118
172.31.32.36
```

:wq



```
# Ex 1: Ungrouped hosts, specify before any group headers.


[Zeeman]
172.31.34.118
172.31.32.36

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group
```

This will not work until we will do any change in configuration file. (Path is almost same)

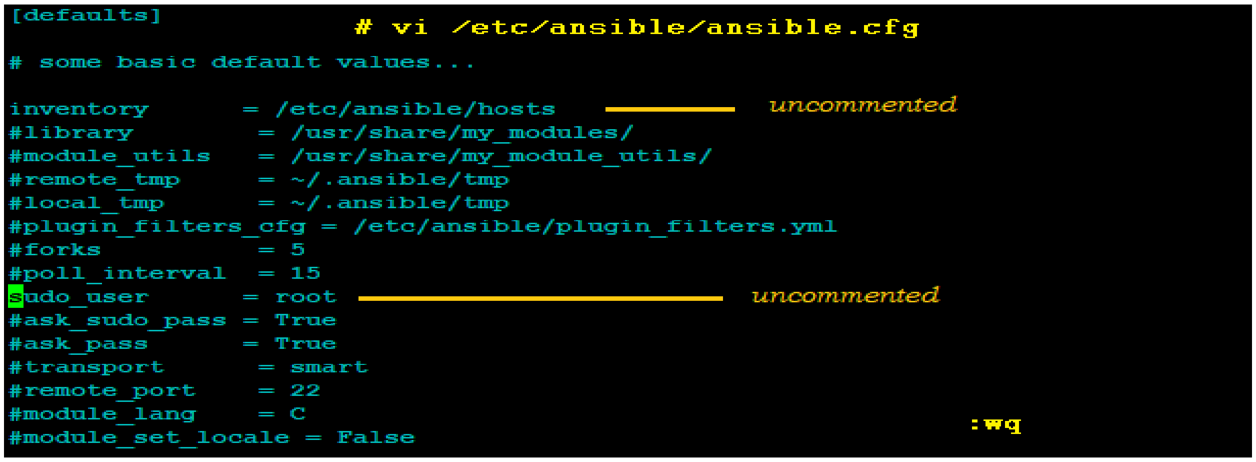
```
[] # vi /etc/ansible/ansible.cfg
```

Press I to insert

Active the Inventory and sudo_user by removing #, means Uncommented them.

```
#Inventory       = /etc/ansible/hosts
#sudo_user       = root
```

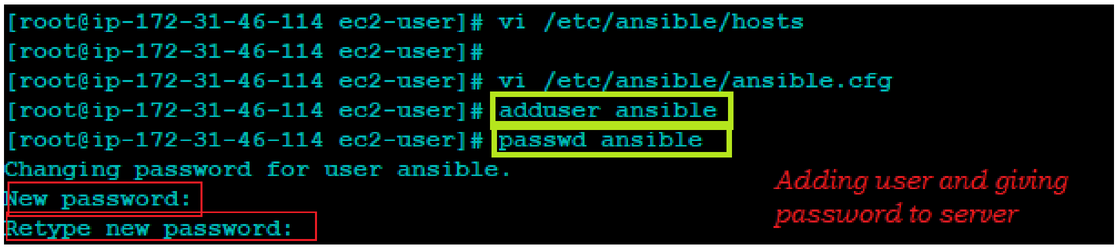
```
:wq
```



```
[defaults]
                    # vi /etc/ansible/ansible.cfg
# some basic default values...

inventory       = /etc/ansible/hosts
#library         = /usr/share/my_modules/
#module_utils    = /usr/share/my_module_utils/
#remote_tmp      = ~/.ansible/tmp
#local_tmp       = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
#forks           = 5
#poll_interval   = 15
sudo_user       = root
#ask_sudo_pass = True
#ask_pass       = True
#transport       = smart
#remote_port     = 22
#module_lang     = C
#module_set_locale = False
```

*Create user for safe communication between nodes to server   *(Do same for all machines)*

```
[] # adduser ansible
[] # passwd ansible
[] # TECHNICAL
[] # New password: TECHNICAL
```



```
[root@ip-172-31-46-114 ec2-user]# vi /etc/ansible/hosts
[root@ip-172-31-46-114 ec2-user]#
[root@ip-172-31-46-114 ec2-user]# vi /etc/ansible/ansible.cfg
[root@ip-172-31-46-114 ec2-user]# adduser ansible
[root@ip-172-31-46-114 ec2-user]# passwd ansible
Changing password for user ansible.
New password:
Retype new password:
```

If wanted to login any Node through switch user (Ansible user), use this command (for all machines)

```
[] # su - ansible
```

Suppose this for Ansible server, check all commands of creating files and directory are working?

```
[] $ touch fileA
[] $ ls
[] $ fileA
[] $ yum install httpd -y
```
you need to be root to perform this command

```
[] $ sudo yum install httpd -y
```
Asking password for Ansible

```
[sudo] password for ansible: TECHNICAL
```

Ansible is not in the sudoers file. For this we have to give sudo privileges & rights.

```
[] # Exit
```
*get out from Ansible user and execute **visudo***

```
[] # visudo
```
*Edit inside the **Allow root rights**, give privileges (all machines)*

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

```
:wq!
```

Here we are adding ansible, means giving it same permission as root have, and No password required

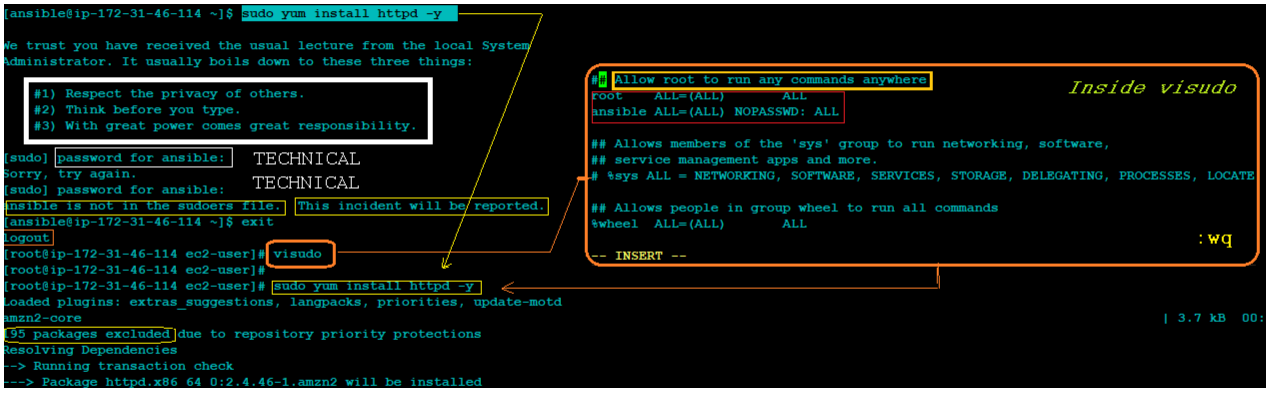
Again Check all commands of creating files and directory are working or not in Ansible server?

```
[] # su - ansible
[] $ yum install httpd -y
```
*you need to be **root** to perform this command*

```
[] $ sudo yum install httpd -y
```
*after adding **sudo** this is **executed*** perfectly*

**Ansible user got the privileges to work as SUDO USER by su – ansible*

Check the communication has stablished between nodes and server by login ansible user (su-ansible) in machines, means- -
Check, if do something on node and push to the server and create something on server and update on node.



```
[ansible@ip-172-31-46-114 ~]$ sudo yum install httpd -y

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for ansible:     TECHNICAL
Sorry, try again.
[sudo] password for ansible:     TECHNICAL
ansible is not in the sudoers file.  This incident will be reported.
[ansible@ip-172-31-46-114 ~]$ exit
logout
[root@ip-172-31-46-114 ec2-user]# visudo
[root@ip-172-31-46-114 ec2-user]#
[root@ip-172-31-46-114 ec2-user]# sudo yum install httpd -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                    | 3.7 kB  00:
195 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86 64 0:2.4.46-1.amzn2 will be installed
```

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

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

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

-- INSERT --
```

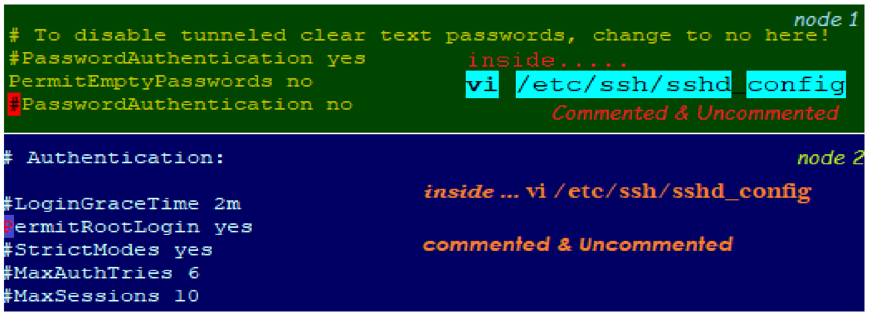
For connection establishment of ansible server + Node, execute SSH with Pvt IP of that Node.

[] # ssh 172.31.34.118 *Permission **denied** because public key is not updated*

[] # Exit

[] # vi /etc/ssh/sshd_config *Go inside and do 3 important changes (all machines)*

Uncommented PermitRootLogin yes
Uncommented PasswordAuthentication yes
Commented PasswordAuthentication no **:wq**



```
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
PermitEmptyPasswords no
#PasswordAuthentication no
```

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

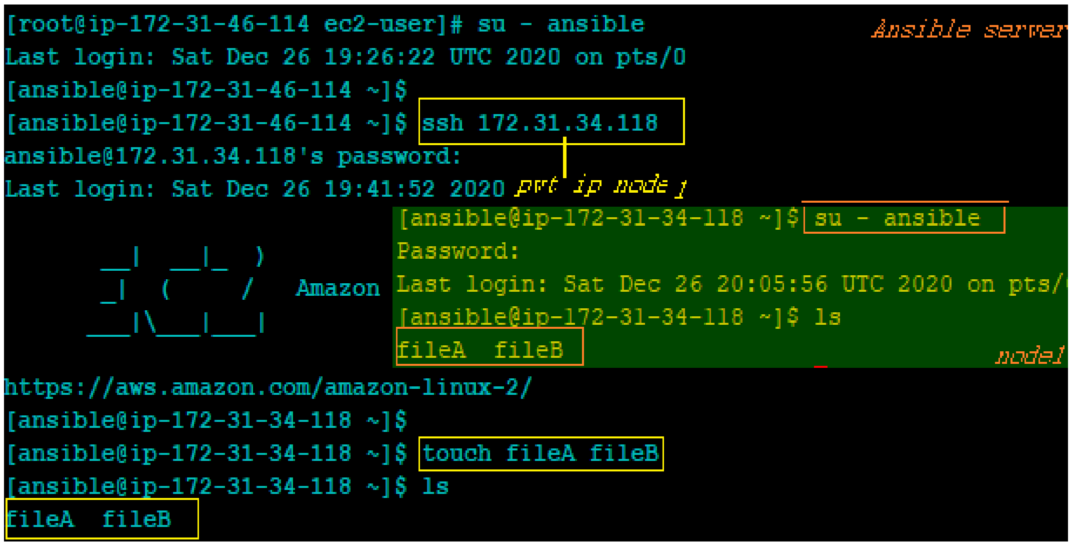
[] # *service sshd restart* *for better Implementation*

***For checking the communication* 'su - ansible'** *execute in All 3 machines*

Check server can access the node1 and node2

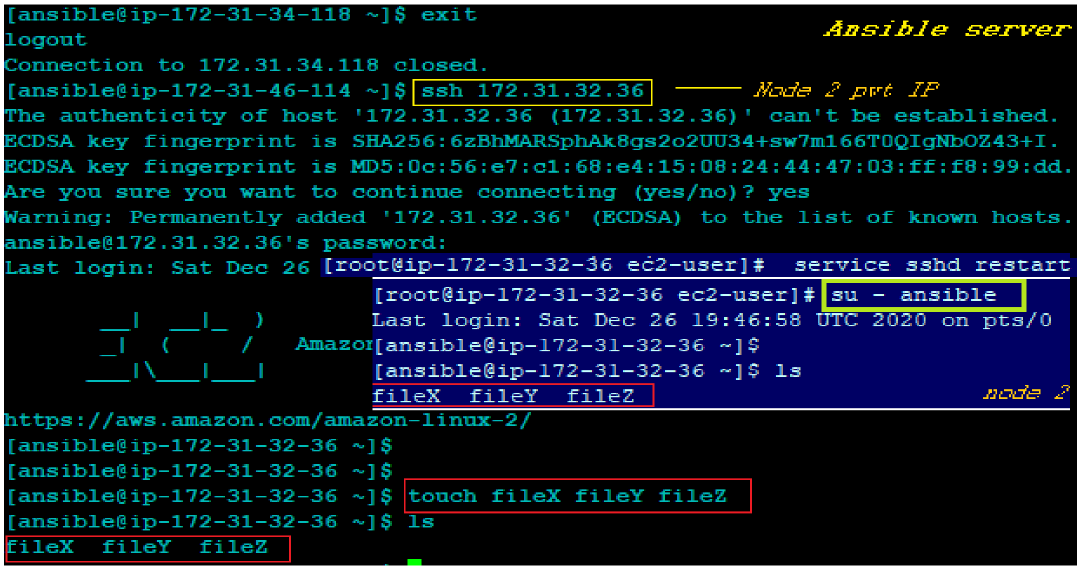
[] $ ssh 172.31.34.118 *After execution you can access **node 1** through server*
[] $ ssh 172.31.32.36 *After execution you can access **node 2** through server*

Create files and directory on server accessing node1 and node 2, and same files you will get inside node 1 and node 2 means, communication between server and nodes is perfect.



```
[root@ip-172-31-46-114 ec2-user]# su - ansible
Last login: Sat Dec 26 19:26:22 UTC 2020 on pts/0
[ansible@ip-172-31-46-114 ~]$
[ansible@ip-172-31-46-114 ~]$ ssh 172.31.34.118
ansible@172.31.34.118's password:
Last login: Sat Dec 26 19:41:52 2020

       __|  __|_  )
       _|  (     /   Amazon
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ansible@ip-172-31-34-118 ~]$
[ansible@ip-172-31-34-118 ~]$ touch fileA fileB
[ansible@ip-172-31-34-118 ~]$ ls
fileA  fileB
```

```
[ansible@ip-172-31-34-118 ~]$ su - ansible
Password:
Last login: Sat Dec 26 20:05:56 UTC 2020 on pts/
[ansible@ip-172-31-34-118 ~]$ ls
fileA  fileB
```



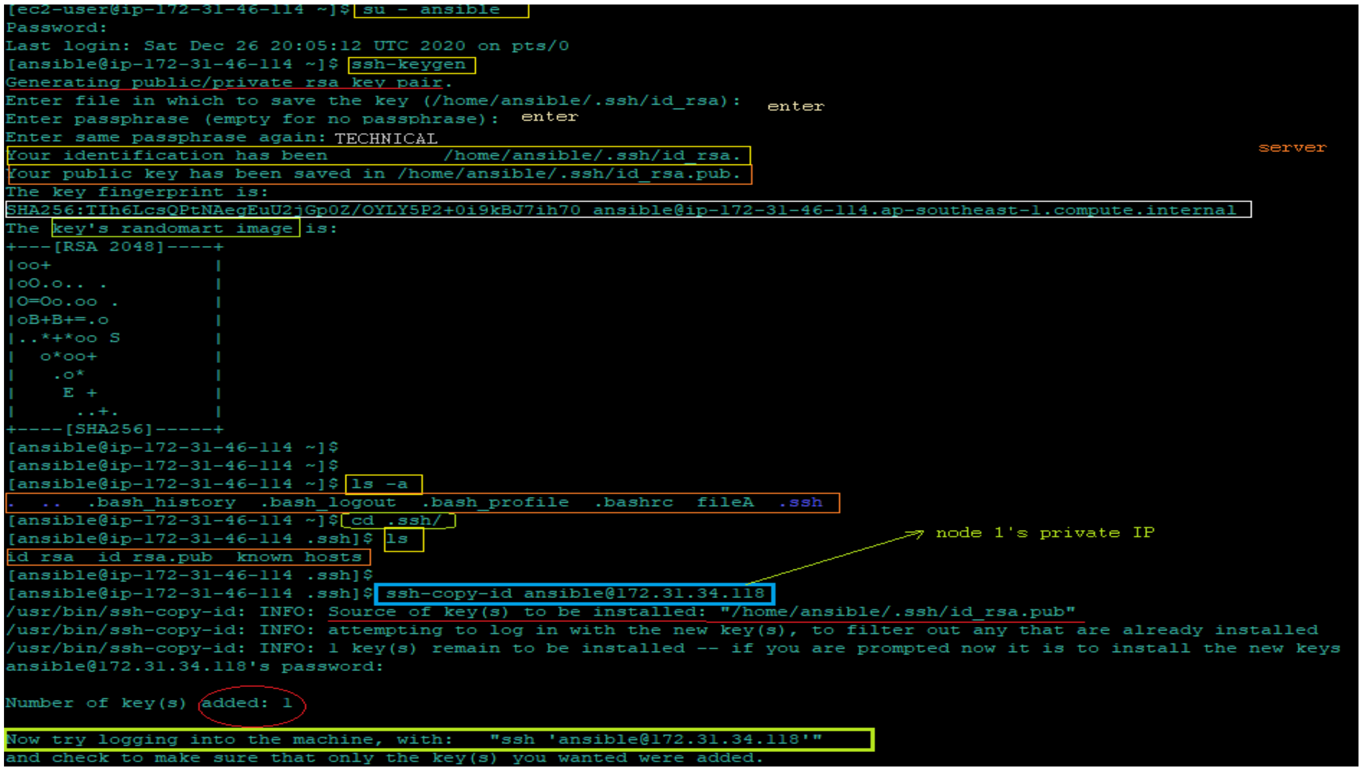
```
[ansible@ip-172-31-34-118 ~]$ exit
logout
Connection to 172.31.34.118 closed.
[ansible@ip-172-31-46-114 ~]$ ssh 172.31.32.36
The authenticity of host '172.31.32.36 (172.31.32.36)' can't be established.
ECDSA key fingerprint is SHA256:6zBhMARSphAk8gs2o2UU34+sw7m166T0QIgNbOZ43+I.
ECDSA key fingerprint is MD5:0c:56:e7:c1:68:e4:15:08:24:44:47:03:ff:f8:99:dd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.32.36' (ECDSA) to the list of known hosts.
ansible@172.31.32.36's password:
Last login: Sat Dec 26

       __|  __|_  )
       _|  (     /   Amazon
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ansible@ip-172-31-32-36 ~]$
[ansible@ip-172-31-32-36 ~]$
[ansible@ip-172-31-32-36 ~]$ touch fileX fileY fileZ
[ansible@ip-172-31-32-36 ~]$ ls
fileX  fileY  fileZ
```

```
[root@ip-172-31-32-36 ec2-user]#  service sshd restart
[root@ip-172-31-32-36 ec2-user]# su - ansible
Last login: Sat Dec 26 19:46:58 UTC 2020 on pts/0
[ansible@ip-172-31-32-36 ~]$
[ansible@ip-172-31-32-36 ~]$ ls
fileX  fileY  fileZ
```

For connection every time asked for the password this is not good for good performance & accuracy that's why we created the TRUST RELATIONSHIP. This can be happen between root to root and user to user.

```
[        ] $ exit
[ansible] $ ssh-keygen
```
*Generation Pub/pvt rsa key pair… Press **Enter** Enter **Enter**,*

```
[ansible] $ ls -a
       .bas history .bash logout .bash profile .bashrc file1 .ssh

[ansible] $ cd .ssh/
[    .ssh] $ ls
>--------->id_rsa id_rsa.pub known hosts
```



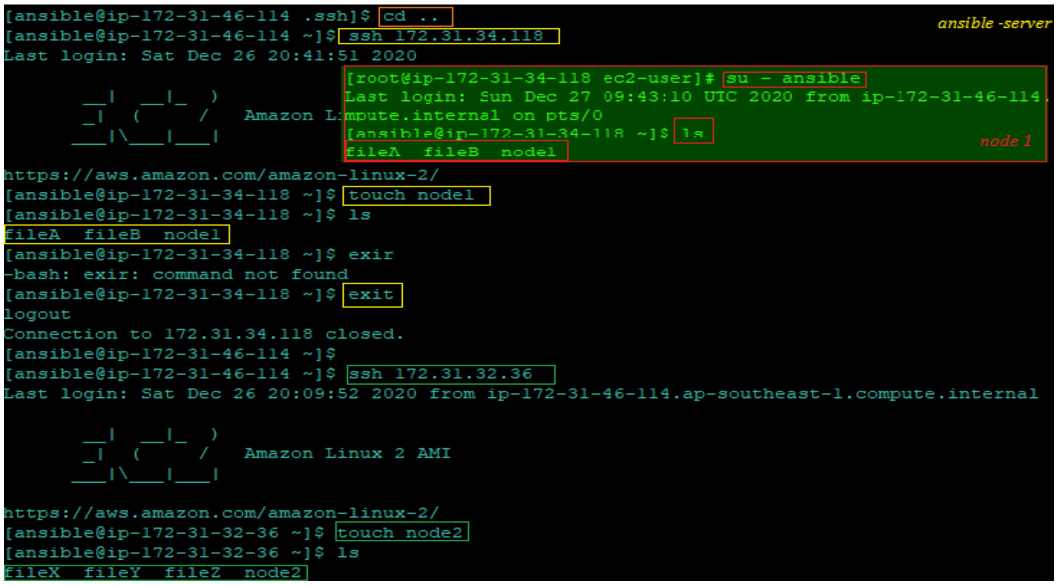
```
[ec2-user@ip-172-31-46-114 ~]$ su - ansible
Password:
Last login: Sat Dec 26 20:05:12 UTC 2020 on pts/0
[ansible@ip-172-31-46-114 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):  enter
Enter passphrase (empty for no passphrase):  enter
Enter same passphrase again: TECHNICAL
Your identification has been          /home/ansible/.ssh/id_rsa.
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:TIh6LcsQPtNAegEuU2jGp0Z/OYLY5P2+0i9kBJ7ih70 ansible@ip-172-31-46-114.ap-southeast-1.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
|oo+              |
|oO.o.. .         |
|O=Oo.oo .        |
|oB+B+=.o         |
|..*+*oo S        |
|  o*oo+          |
|   .o*           |
|    E +          |
|      ..+.       |
+----[SHA256]-----+
[ansible@ip-172-31-46-114 ~]$
[ansible@ip-172-31-46-114 ~]$
[ansible@ip-172-31-46-114 ~]$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  fileA  .ssh
[ansible@ip-172-31-46-114 ~]$ cd .ssh/
[ansible@ip-172-31-46-114 .ssh]$ ls
id_rsa  id_rsa.pub  known_hosts
[ansible@ip-172-31-46-114 .ssh]$
[ansible@ip-172-31-46-114 .ssh]$ ssh-copy-id ansible@172.31.34.118
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@172.31.34.118's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansible@172.31.34.118'"
and check to make sure that only the key(s) you wanted were added.
```

Copy servers' public key (id_rsa.pub) in all the nodes to remind the nodes, not to ask for password all the time just give the permissions.

```
[   .ssh] $ ssh-copy-id ansible@172.31.34.118
ansible@172.31.41.240's password: TECHNICAL
```

Last time asked for Password

```
[   .ssh] $ ssh-copy-id ansible@172.31.32.36
```



```
[ansible@ip-172-31-46-114 .ssh]$ cd ..
[ansible@ip-172-31-46-114 ~]$ ssh 172.31.34.118
Last login: Sat Dec 26 20:41:51 2020

       __|  __|_  )
       _|  (     /   Amazon L
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ansible@ip-172-31-34-118 ~]$ touch node1
[ansible@ip-172-31-34-118 ~]$ ls
fileA  fileB  node1
[ansible@ip-172-31-34-118 ~]$ exir
-bash: exir: command not found
[ansible@ip-172-31-34-118 ~]$ exit
logout
Connection to 172.31.34.118 closed.
[ansible@ip-172-31-46-114 ~]$
[ansible@ip-172-31-46-114 ~]$ ssh 172.31.32.36
Last login: Sat Dec 26 20:09:52 2020 from ip-172-31-46-114.ap-southeast-1.compute.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ansible@ip-172-31-32-36 ~]$ touch node2
[ansible@ip-172-31-32-36 ~]$ ls
fileX  fileY  fileZ  node2
```

```
[root@ip-172-31-34-118 ec2-user]# su - ansible
Last login: Sun Dec 27 09:43:10 UTC 2020 from ip-172-31-46-114.
mpute.internal on pts/0
[ansible@ip-172-31-34-118 ~]$ ls
fileA  fileB  node1
```

Host pattern is helpful for huge numbers of connected nodes.

"All" pattern refer to all the machines in an inventory

```
[   .ssh] $ cd ..
```

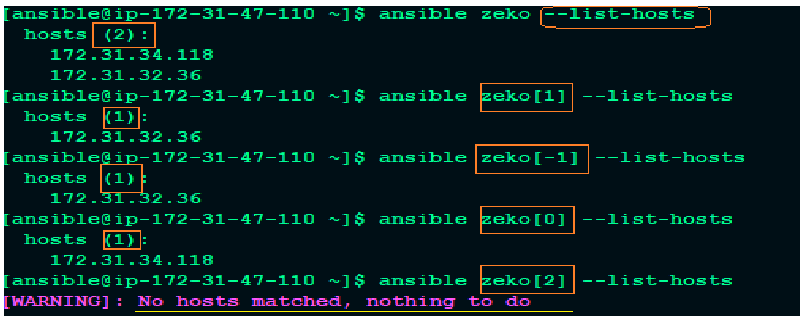
```
[ansible] $ ansible all --list-hosts
hosts (2):
172.31.41.240
172.31.41.248
```

*check no of all nodes, **hosts***

```
[ansible] $ ansible Zeeman --list-hosts
hosts (2):
172.31.41.240
172.31.41.248
```

*check nodes inside **group**=zeko,Zeeman*

```
[ansible] $ ansible Zeeman(0) --list-hosts
hosts (1):
172.31.41.240
[ansible] $ ansible Zeeman(1) --list-hosts
hosts (1):
172.31.41.248
[ansible] $ ansible Zeeman(-1) --list-hosts
hosts (1):
172.31.41.248
[ansible] $ ansible Zeeman(3) --list-host
[WARNING]: No host matched, nothing to do
```



```
[ansible@ip-172-31-47-110 ~]$ ansible zeko --list-hosts
  hosts (2):
    172.31.34.118
    172.31.32.36
[ansible@ip-172-31-47-110 ~]$ ansible zeko[1] --list-hosts
  hosts (1):
    172.31.32.36
[ansible@ip-172-31-47-110 ~]$ ansible zeko[-1] --list-hosts
  hosts (1):
    172.31.32.36
[ansible@ip-172-31-47-110 ~]$ ansible zeko[0] --list-hosts
  hosts (1):
    172.31.34.118
[ansible@ip-172-31-47-110 ~]$ ansible zeko[2] --list-hosts
[WARNING]: No hosts matched, nothing to do
```

Ad-hoc command:

It is individual running commands, which can be run individually to perform quick functions.
It's not use for configuration management and deployment because the commands are of one time usage.
Ad-hoc commands uses the /**user/bin/ansible** command line tool to automate the signal task.

Important Ad-hoc commands:

```
[ansible@ip]$ ansible zeko -a "ls"
```

***zeko**= group*
***a**=argument*

```
[ansible@ip]$ ansible zeko[0] -a "touch filezz"
```

update file to particular node

```
[ansible@ip]$ ansible all -a "touch filek"
```

update file to all nodes

```
[ansible@ip]$ ansible zeko -a "ls-al"
```

*show total list with all details of **groups***

```
[ansible@ip]$ ansible zeko -a "sudo yum install httpd -y"
[ansible@ip]$ ansible zeko -ba "yum install httpd -y"
```

***b**=become sudo*

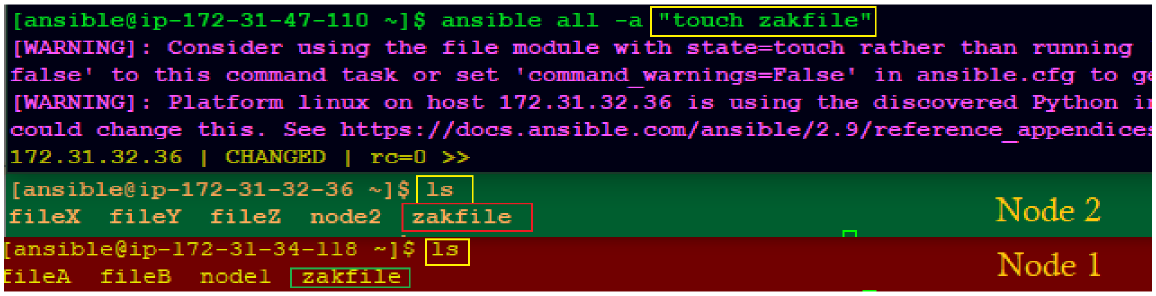
```
[ansible@ip]$ ansible zeko -ba "yum remove httpd -y"
```

Now create a file name zakfile on server to check Idempotency and Push Mechanism

```
[ansible@ip]$ ansible all -a "touch zakfile"
```

Changed | rc=0 >> showing change status but nothing happed new
Changed | rc=0 >> showing change status but not new work done

Now zakfile has created you can also find these files inside the nodes by ls command, that is called - Push Mechanism
It will overwrite again=2 without letting us know that, zakfile already has been created in past called -No Idempotency



```
[ansible@ip-172-31-47-110 ~]$ ansible all -a "touch zakfile"
[WARNING]: Consider using the file module with state=touch rather than running
false' to this command task or set 'command_warnings=False' in ansible.cfg to ge
[WARNING]: Platform linux on host 172.31.32.36 is using the discovered Python in
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices
172.31.32.36 | CHANGED | rc=0 >>
[ansible@ip-172-31-32-36 ~]$ ls
fileX  fileY  fileZ  node2  zakfile
[ansible@ip-172-31-34-118 ~]$ ls
fileA  fileB  node1  zakfile
```

*Trying install httpd but Failed because need sudo privilege, add **sudo / b** for proper installation.*

```
[@-ec2-user]# su - ansible
[ansible@ip]$ ansible zeko -a "yum install httpd -y"
[ansible@ip]$ ansible zeko -ba "yum install httpd -y"
```

failed
Installation done all nodes

```
[ansible@ip]$ which httpd
```

check Installed files in all nodes

```
[ansible@ip]$ /usr/sbin/httpd
```

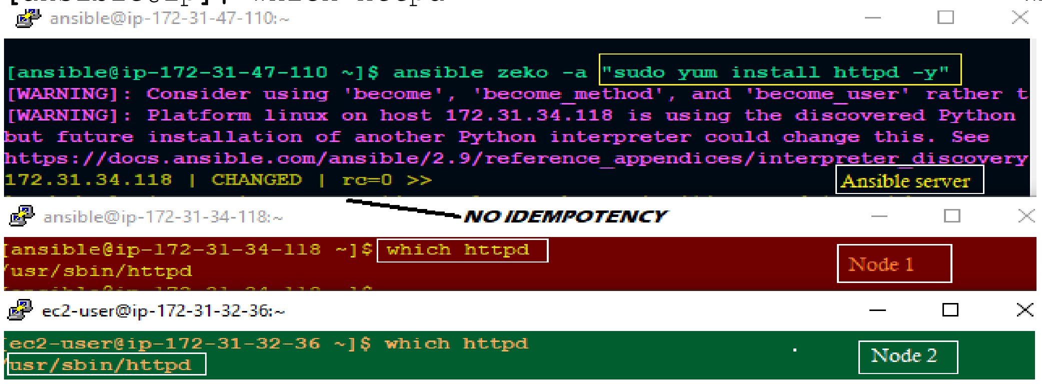
output in all nodes

```
[ansible@ip]$ ansible zeko -a "sudo yum remove httpd -y"
```

Uninstallation

```
[ansible@ip]$ which httpd
```

No output all deleted



```
ansible@ip-172-31-47-110:~
[ansible@ip-172-31-47-110 ~]$ ansible zeko -a "sudo yum install httpd -y"
[WARNING]: Consider using 'become', 'become_method', and 'become_user' rather t
[WARNING]: Platform linux on host 172.31.34.118 is using the discovered Python
but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery
172.31.34.118 | CHANGED | rc=0 >>
ansible@ip-172-31-34-118:~
[ansible@ip-172-31-34-118 ~]$ which httpd
/usr/sbin/httpd
ec2-user@ip-172-31-32-36:~
[ec2-user@ip-172-31-32-36 ~]$ which httpd
/usr/sbin/httpd
```

Ansible Module:

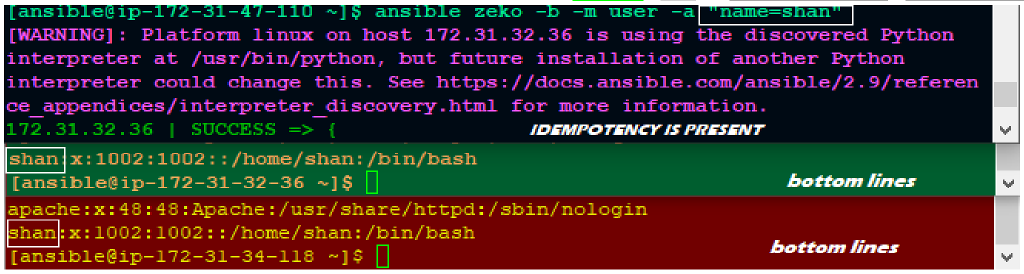
Ansible ships with a number of modules (called module library) that can be executed directory on remote host or through **playbook**. Your library of modules can reside on any machine and there are no servers, daemon or database required. (***Idempotency is present***).

Qn. Where ansible modules are stored? **Ans**: the default location for the inventory file is **/etc/ansible/hosts**

Important Module Commands:

Install >> present   update >> latest   uninstall >> remove

```
[ansible@ip]$ ansible zeko -b -m yum -a "pkg=httpd state=present"
[ansible@ip]$ ansible zeko -b -m service -a "name=httpd state=started"
[ansible@ip]$ ansible zeko -b -m user -a "name=shan"
[ansible@ip]$ ansible zeko -b -m copy -a "src=fileA dest=/tmp"
```



```
[ansible@ip-172-31-47-110 ~]$ ansible zeko -b -m user -a "name=shan"
[WARNING]: Platform linux on host 172.31.32.36 is using the discovered Python
interpreter at /usr/bin/python, but future installation of another Python
interpreter could change this.  See https://docs.ansible.com/ansible/2.9/referen
ce_appendices/interpreter_discovery.html for more information.
172.31.32.36 | SUCCESS => {
```

```
shan:x:1002:1002::/home/shan:/bin/bash
[ansible@ip-172-31-32-36 ~]$
```

```
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
shan:x:1002:1002::/home/shan:/bin/bash
[ansible@ip-172-31-34-118 ~]$
```

Update package of installed httpd:

```
[ansible@ip]$ ansible zeko -b -m yum -a "pkg=httpd state=latest"
[ansible@ip]$ which httpd
[ansible@ip]$ /usr/bin/httpd
```
check in all nodes

Delete installed package: both nodes:

```
[ansible@ip]$ ansible zeko -b -m yum -a "pkg=httpd state=absent"
[ansible@ip]$ sudo service httpd status
```
Inactive

To start a service execute this command:

```
[ansible@ip]$ ansible zeko -b -m service -a "name=httpd state=started"
[ansible@ip]$ sudo service httpd status
```
Active = Running

To create the user:

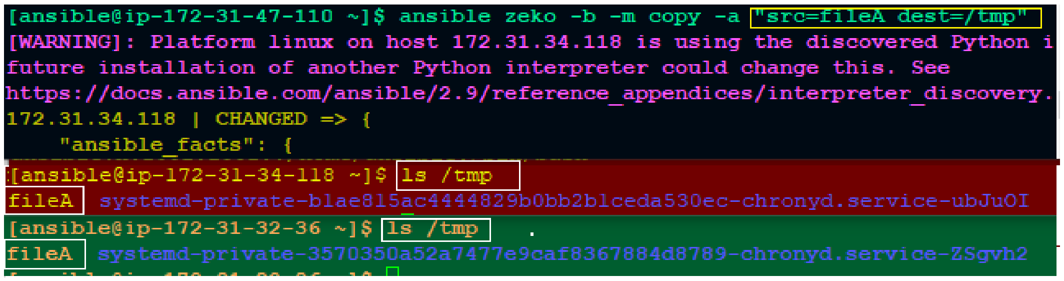
```
[ansible@ip]$ ansible zeko -b -m user -a "name=shan"
```

To check the status of user:

```
[ansible@ip]$ cat /etc/passwd
```
*Check in all nodes you will get user **shan** at the bottem lines*

To copy item from source to a Particular Node/destination: suppose only on last node [-1].

```
[ansible@ip]$ touch fileXYZ
[ansible@ip]$ ansible zeko[-1] -b -m copy -a "src=fileXYZ dest=/tmp"
```



```
[ansible@ip-172-31-47-110 ~]$ ansible zeko -b -m copy -a "src=fileA dest=/tmp"
[WARNING]: Platform linux on host 172.31.34.118 is using the discovered Python i
future installation of another Python interpreter could change this.  See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.
172.31.34.118 | CHANGED => {
    "ansible_facts": {
[ansible@ip-172-31-34-118 ~]$ ls /tmp
fileA  systemd-private-b1ae815ac4444829b0bb2b1ceda530ec-chronyd.service-ubJuOI
[ansible@ip-172-31-32-36 ~]$ ls /tmp
fileA  systemd-private-3570350a52a7477e9caf8367884d8789-chronyd.service-ZSgvh2
```

To copy item from source to destination: for all nodes:

```
[ansible@ip]$ ansible zeko -b -m copy -a "src=fileA dest=/tmp"
```

Ansible setup: It works like Ohai works in CHEF, avoids Noidempotency.

Useful commands: It will give all the details related to IP addresses of that particular node.

```
[ansible@ip]$ ansible zeko -m setup
[ansible@ip]$ ansible zeko -m setup -a "filter=*ipv4*"
```

Playbook:

Playbook is written in **YAML**. It is like a file where you write codes consist of **vars**, **task**, and **handlers** **templates** and **roles.**

Each playbook is composed of one or more **module** in a list, Module is a collection of configuration files

Playbook is divided into many sections:

Target section: It defines the host against which playbook's task has to be executed.

Variable section: It defines variables

Task section: It defines list of all modules that we need to run an order.

YAML:

For Ansible, nearly every YAML file starts with a list.

Each item in the List is a list of **key-volume** pair's commonly called a Dictionary.

A Dictionary is represented in a simple **Key: Volume** form

All YAML files have to begin "_ _ _" and end with "…" and extension for playbook is **.yml.**

All members of a list lines must begin with same **Indentation** level starting with **"--".**

Example YAML for Dictionary:

```
- - -  #Detail of customers                           ..............commented
Customers:
  name: Zeeshan
  Job: Engineer
  Skill: Development
  experience: 5 years                                                        :wq
```

Example-create a Target playbook:

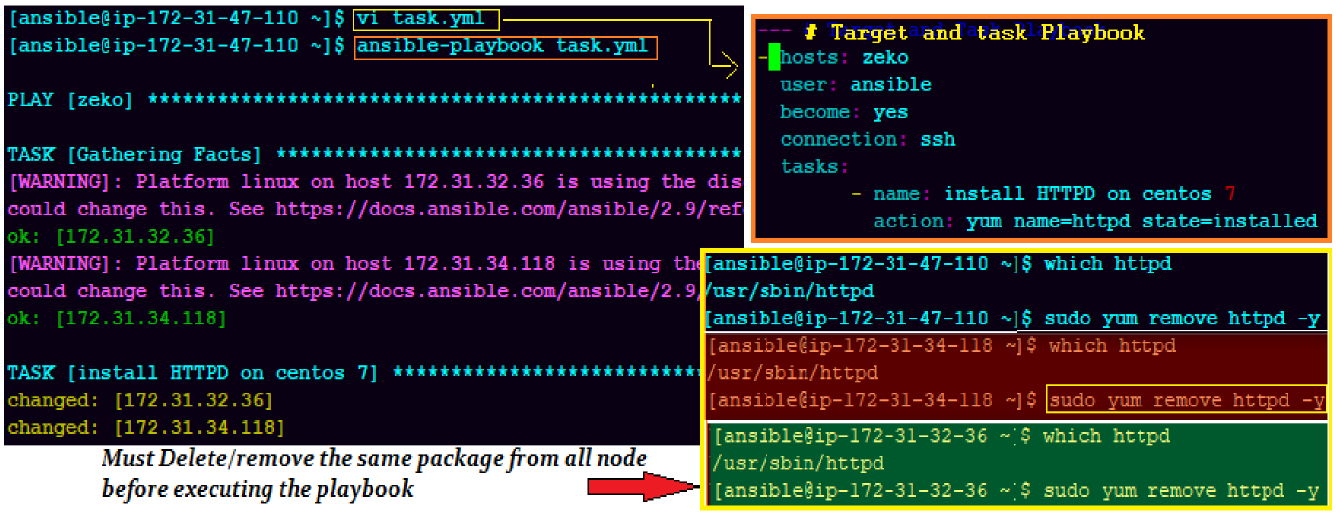
```
[ansible@Ip]$ Vi target.yml

- - -  #My Target Playbook
- host: zeko
  user: ansible
  become: yes
  connection: ssh
  gather-facts: yes                                                            :wq

[ansible@Ip]$ ansible-playbook target.yml                     ←To execute this playbook
```

Example-create a Task playbook:

```
[ansible@Ip]$ Vi task.yml

- - - #My Task Playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
```

```
tasks:
      - name: install HTTPD on centos 7
        action: yum name=httpd state=installed                              :wq
```

```
[ansible@Ip]$ ansible-playbook task.yml
```
←To execute this playbook



```
[ansible@ip-172-31-47-110 ~]$ vi task.yml
[ansible@ip-172-31-47-110 ~]$ ansible-playbook task.yml

PLAY [zeko] ***************************************************

TASK [Gathering Facts] ****************************************
[WARNING]: Platform linux on host 172.31.32.36 is using the dis
could change this. See https://docs.ansible.com/ansible/2.9/ref
ok: [172.31.32.36]
[WARNING]: Platform linux on host 172.31.34.118 is using the
could change this. See https://docs.ansible.com/ansible/2.9/
ok: [172.31.34.118]

TASK [install HTTPD on centos 7] ******************************
changed: [172.31.32.36]
changed: [172.31.34.118]
```

```
--- # Target and task Playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
        - name: install HTTPD on centos 7
          action: yum name=httpd state=installed
```

```
[ansible@ip-172-31-47-110 ~]$ which httpd
/usr/sbin/httpd
[ansible@ip-172-31-47-110 ~]$ sudo yum remove httpd -y
```

```
[ansible@ip-172-31-34-118 ~]$ which httpd
/usr/sbin/httpd
[ansible@ip-172-31-34-118 ~]$ sudo yum remove httpd -y
```

```
[ansible@ip-172-31-32-36 ~]$ which httpd
/usr/sbin/httpd
[ansible@ip-172-31-32-36 ~]$ sudo yum remove httpd -y
```

Must Delete/remove the same package from all node before executing the playbook

Variables:

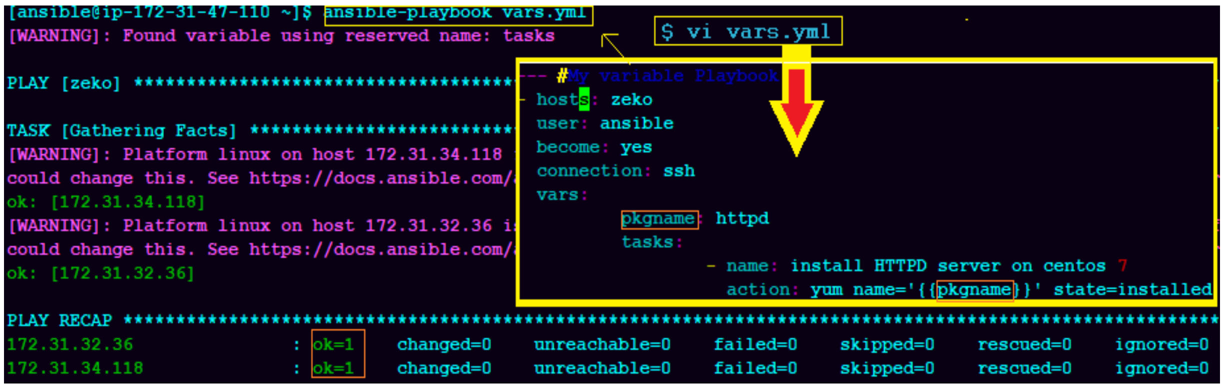
Ansible uses variables which enable more flexibility in playbook and roles they can be uses to loop through a set of given values, access various information like the hostname of a system and replace strings in templates with specific values.
Put the **variable section above the tasks** so that we define it first and use it later.

Example create a variable playbook:

```
[ansible@Ip]$ Vi vars.yml
```

```
- - - #My variable Playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  vars:
          pkgname: httpd
          tasks:
                  - name: install HTTPD server on centos 7
                    action: yum name='{{pkgname}}'state=installed          :wq
```

```
[ansible@Ip]$ ansible-playbook vars.yml
```
←execute this playbook



```
[ansible@ip-172-31-47-110 ~]$ ansible-playbook vars.yml
[WARNING]: Found variable using reserved name: tasks

PLAY [zeko] ****************************************

TASK [Gathering Facts] *****************************
[WARNING]: Platform linux on host 172.31.34.118
could change this. See https://docs.ansible.com/
ok: [172.31.34.118]
[WARNING]: Platform linux on host 172.31.32.36 i
could change this. See https://docs.ansible.com/
ok: [172.31.32.36]

PLAY RECAP *****************************************************************************************
172.31.32.36               : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.34.118              : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
$ vi vars.yml
--- #My variable Playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  vars:
          pkgname: httpd
          tasks:
                  - name: install HTTPD server on centos 7
                    action: yum name='{{pkgname}}' state=installed
```

Handlers:

Handler is same like task but it will run when called by another task. Handler only run when task contains a ***notify*** directive and also indicate that it changed something.

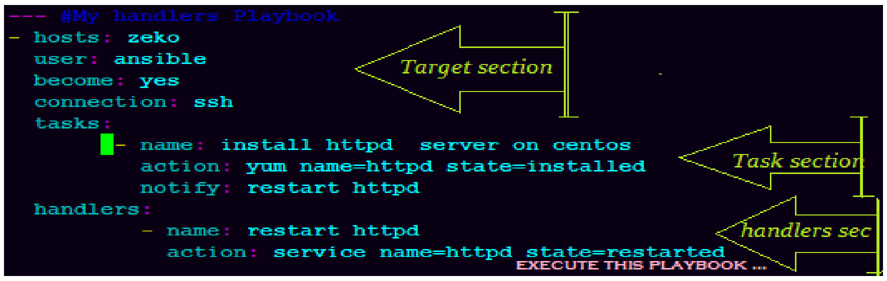
Example-create a Task playbook:

```
[ansible@Ip]$ Vi handlers.yml
```

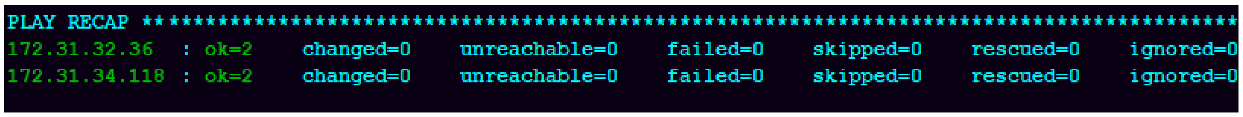
```
--- #My handlers Playbook

- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
         - name: install HTTPD server on centos 7
           action: yum name=httpd state=installed
           notify: restart httpd
  handlers:
           - name: restart httpd
             action: service name=httpd state=restarted
```

```
[ansible@Ip]$ ansible-playbook Handlers.yml
```

Before execution delete httpd from nodes



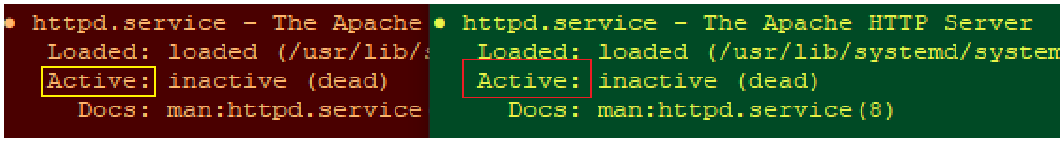
```
--- #My handlers Playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: install httpd  server on centos
            action: yum name=httpd state=installed
            notify: restart httpd
  handlers:
            - name: restart httpd
              action: service name=httpd state=restarted
```



```
PLAY RECAP **********************************************************************************************
172.31.32.36  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.34.118 : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Dry run: It is the process to check errors before execute the playbook, it checks that Playboook is formatted correctly or not?

```
[ansible@Ip]$ ansible-playbook Handlers.yml –check
[ansible@Ip]$ sudo service httpd status
```
check the status of all nodes active or not?



```
● httpd.service - The Apache
  Loaded: loaded (/usr/lib/s
  Active: inactive (dead)
    Docs: man:httpd.service
```

```
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system
  Active: inactive (dead)
    Docs: man:httpd.service(8)
```

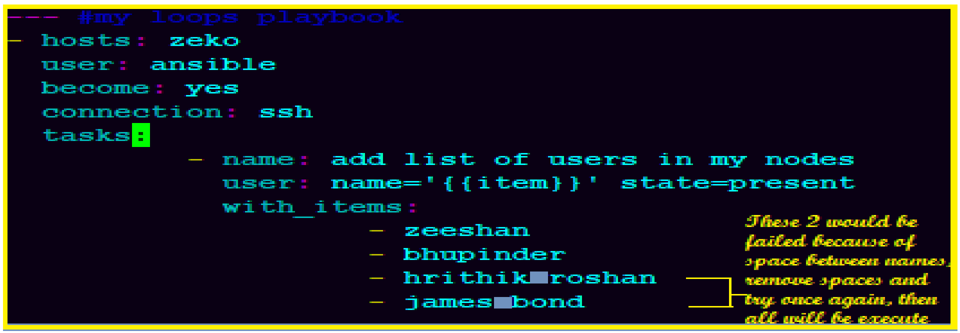
Loops: Sometimes you repeat a task multiple time in programming it is called loop.
Common ansible loop include changing ownership on server files and/or directories with the file module, creating multiple user with the user module and repeating a polling step untill certain result is reached.

Example-create a Loops playbook:

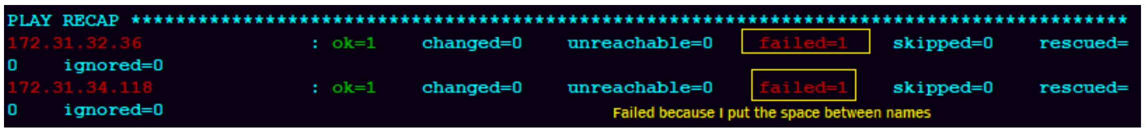
```
[ansible@Ip]$ Vi loops.yml
```

```
- - - #My Loops Playbook

- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: add list of users in my nodes
            user: name='{{item}}' state=present
            with_items:
                    - zeeshan
                    - bhupinder
                    - hrithik roshan
                    - james bond
```

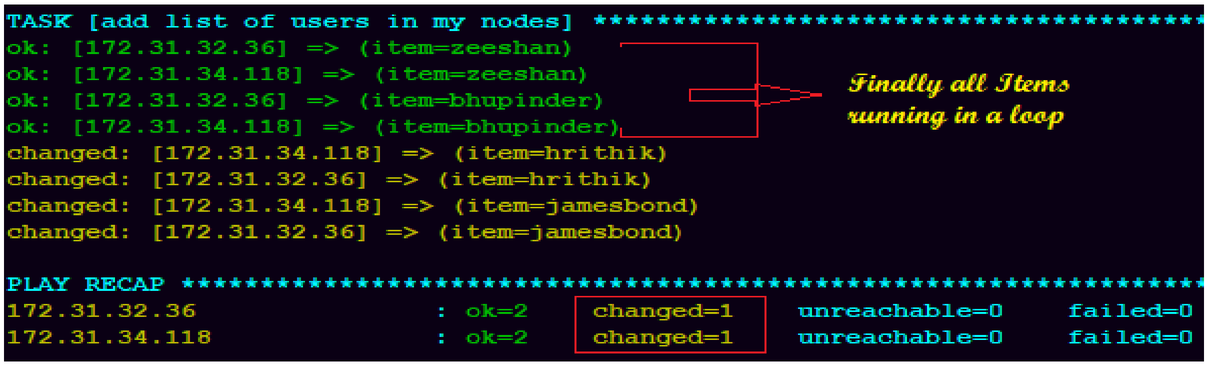
```
[ansible@Ip]$ ansible-playbook loops.yml
```



```
--- #my loops playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: add list of users in my nodes
            user: name='{{item}}' state=present
            with_items:
                    - zeeshan
                    - bhupinder
                    - hrithik roshan
                    - james bond
```
These 2 would be failed because of space between names, remove spaces and try once again, then all will be execute



```
PLAY RECAP *********************************************************************************
172.31.32.36               : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=
0     ignored=0
172.31.34.118              : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=
0     ignored=0
```



```
TASK [add list of users in my nodes] *****************************************
ok: [172.31.32.36] => (item=zeeshan)
ok: [172.31.34.118] => (item=zeeshan)
ok: [172.31.32.36] => (item=bhupinder)
ok: [172.31.34.118] => (item=bhupinder)
changed: [172.31.34.118] => (item=hrithik)
changed: [172.31.32.36] => (item=hrithik)
changed: [172.31.34.118] => (item=jamesbond)
changed: [172.31.32.36] => (item=jamesbond)

PLAY RECAP *******************************************************************
172.31.32.36               : ok=2    changed=1    unreachable=0    failed=0
172.31.34.118              : ok=2    changed=1    unreachable=0    failed=0
```

Condition: when we have different scenarios, then we put condition according to th scenario.
When statement: sometimes you want to skip a particular command on a particular node.

Example-conditional playbook:

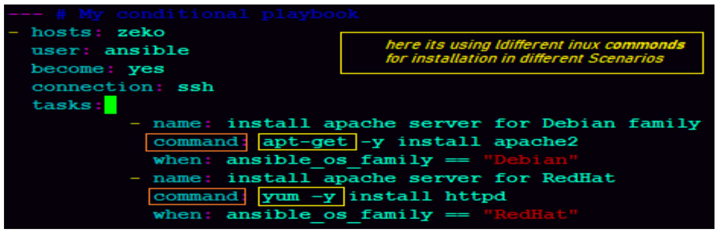
```
[ansible@Ip]$ Vi condition.yml
```

- - - #My Conditional Playbook   apt-get=debian yum=RedHat

```
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: install apache server for Debian family
            command: apt-get -y install apache2
            when: ansible_os_family == "Debian"
          - name: install apache for RedHat
            command: yum -y install httpd
            when: ansible_os_family == "RedHat"
```

```
[ansible@Ip]$ ansible-playbook condition.yml
```

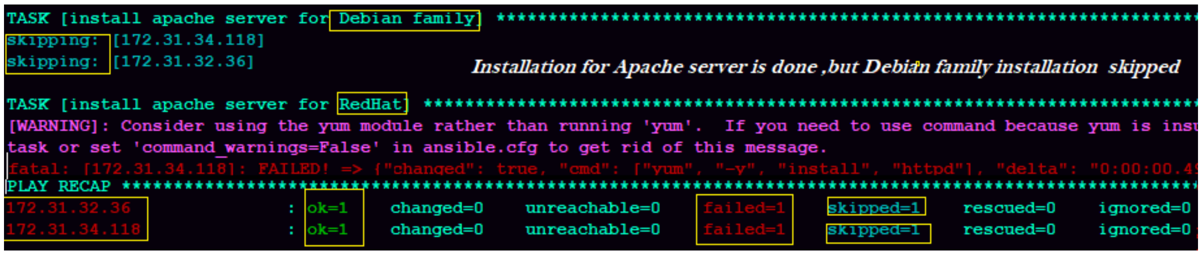
Before execution delete httpd from nodes



```
--- # My conditional playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: install apache server for Debian family
            command: apt-get -y install apache2
            when: ansible_os_family == "Debian"
          - name: install apache server for RedHat
            command: yum -y install httpd
            when: ansible_os_family == "RedHat"
```

here its using ldifferent inux commonds for installation in different Scenarios

Condition of skip is avoids failure



```
TASK [install apache server for Debian family] ***************************************************
skipping: [172.31.34.118]
skipping: [172.31.32.36]

TASK [install apache server for RedHat] **********************************************************
[WARNING]: Consider using the yum module rather than running 'yum'.  If you need to use command because yum is insu
task or set 'command_warnings=False' in ansible.cfg to get rid of this message.
fatal: [172.31.34.118]: FAILED! => {"changed": true, "cmd": ["yum", "-y", "install", "httpd"], "delta": "0:00:00.4
PLAY RECAP ***************************************************************************************
172.31.32.36               : ok=1    changed=0    unreachable=0    failed=1    skipped=1    rescued=0    ignored=0
172.31.34.118              : ok=1    changed=0    unreachable=0    failed=1    skipped=1    rescued=0    ignored=0
```

Installation for Apache server is done ,but Debian family installation skipped

Vault: Ansible allows keeping sensitive data such as passwords or keys in encrypted files, rather than a plain text in your playbooks.

Create a new encrypted Playbook

```
[ansible]$ ansible-vault create Zeevault.yml
```

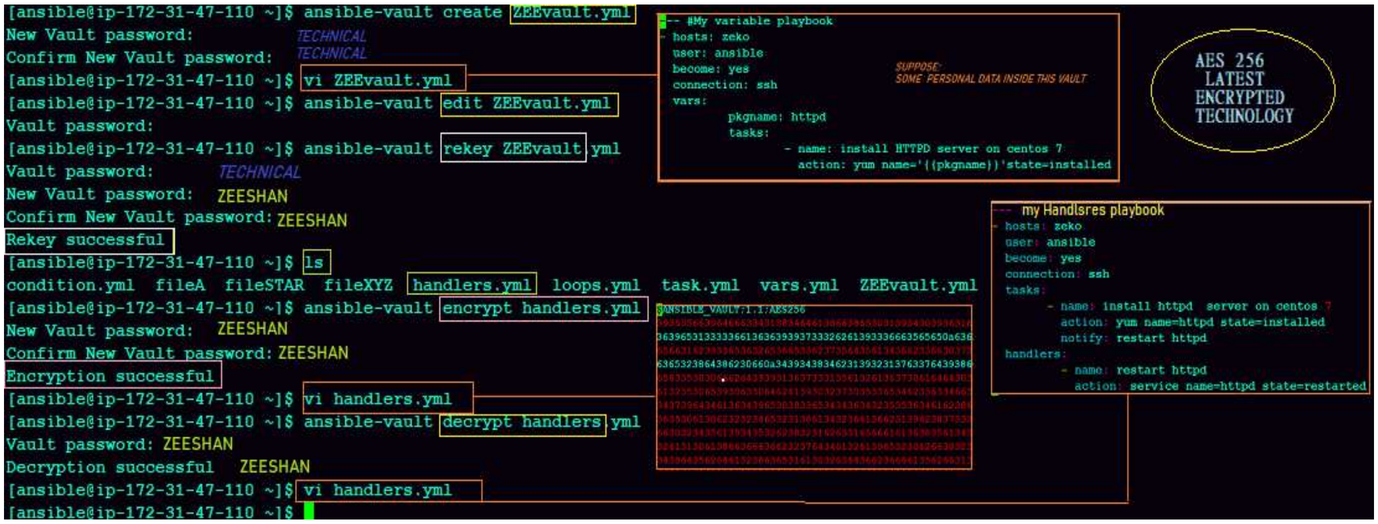
Edit the encrypted playbook

```
[ansible]$ ansible-vault edit Shanvault.yml
```

To change the password of Playbook

```
[ansible]$ ansible-vault rekey Shanvault.yml
```

It will ask for set up a password for Encryption and ask for same password before Decryption.



```
[ansible@ip-172-31-47-110 ~]$ ansible-vault create ZEEvault.yml
New Vault password:            TECHNICAL
Confirm New Vault password:    TECHNICAL
[ansible@ip-172-31-47-110 ~]$ vi ZEEvault.yml
[ansible@ip-172-31-47-110 ~]$ ansible-vault edit ZEEvault.yml
Vault password:
[ansible@ip-172-31-47-110 ~]$ ansible-vault rekey ZEEvault.yml
Vault password:         TECHNICAL
New Vault password:   ZEESHAN
Confirm New Vault password: ZEESHAN
Rekey successful
[ansible@ip-172-31-47-110 ~]$ ls
condition.yml  fileA  fileSTAR  fileXYZ  handlers.yml  loops.yml  task.yml  vars.yml  ZEEvault.yml
[ansible@ip-172-31-47-110 ~]$ ansible-vault encrypt handlers.yml
New Vault password:   ZEESHAN
Confirm New Vault password: ZEESHAN
Encryption successful
[ansible@ip-172-31-47-110 ~]$ vi handlers.yml
[ansible@ip-172-31-47-110 ~]$ ansible-vault decrypt handlers.yml
Vault password: ZEESHAN
Decryption successful   ZEESHAN
[ansible@ip-172-31-47-110 ~]$ vi handlers.yml
[ansible@ip-172-31-47-110 ~]$
```

```
--- #My variable playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  vars:
          pkgname: httpd
          tasks:
                  - name: install HTTPD server on centos 7
                    action: yum name='{{pkgname}}'state=installed
```

```
$ANSIBLE_VAULT;1.1;AES256
[illegible]
363965313333366136363939373332626139333666356565650a636
[illegible]
636532386438623066 0a34393438346231393231376337643938 6
[illegible]
```

```
--- my Handlsres playbook
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: install httpd  server on centos 7
            action: yum name=httpd state=installed
            notify: restart httpd
  handlers:
           - name: restart httpd
             action: service name=httpd state=restarted
```

SUPPOSE:
SOME PERSONAL DATA INSIDE THIS VAULT
AES 256 LATEST ENCRYPTED TECHNOLOGY

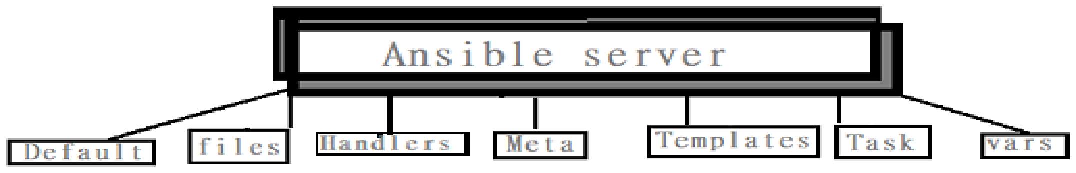
*To encrypt an **existing** Playbook*

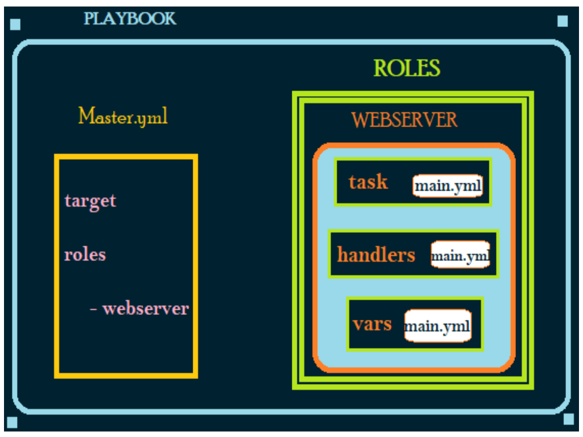
```
[ansible]$ ansible-vault encrypt var.yml
```

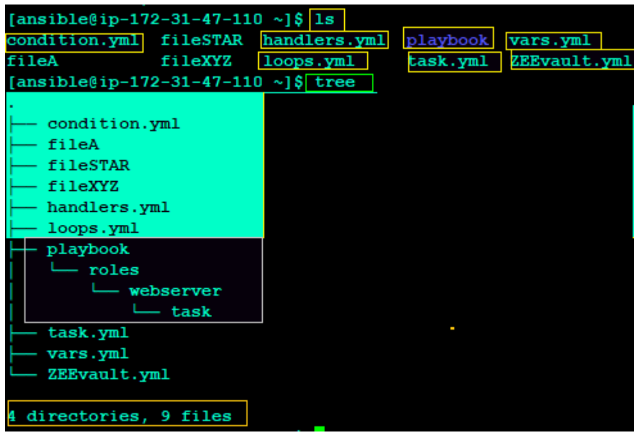
To decrypt and encrypted Playbook

```
[ansible]$ ansible-vault decrypt task.yml
```

Roles: We can use 2 technologies for reusing a set of task: includes and roles.
Roles are good for ***organizing task*** and ***encapsulating data*** needed to accomplish those tasks.
We can organize playbook into a directory structure called Roles.
Adding more and more functionality to the playbooks will make it different to maintain in a single file.







```
[ansible@ip-172-31-47-110 ~]$ ls
condition.yml  fileSTAR  handlers.yml  playbook  vars.yml
fileA          fileXYZ   loops.yml     task.yml  ZEEvault.yml
[ansible@ip-172-31-47-110 ~]$ tree
.
├── condition.yml
├── fileA
├── fileSTAR
├── fileXYZ
├── handlers.yml
├── loops.yml
├── playbook
│   └── roles
│       └── webserver
│           └── task
├── task.yml
├── vars.yml
└── ZEEvault.yml

4 directories, 9 files
```

```
[ansible]$ mkdir -p playbook/roles/webserver/tasks
[ansible]$ sudo yum install tree -y
```

Install package tree

```
[ansible]$ cd playbook/
[playbook]$ ls
 Roles
[playbook]$ tree

      .
        ---roles
     ---webserver
        ---task

[playbook]$ touch /roles/webserver/tasks/main.yml
```

create main.yml inside tasks

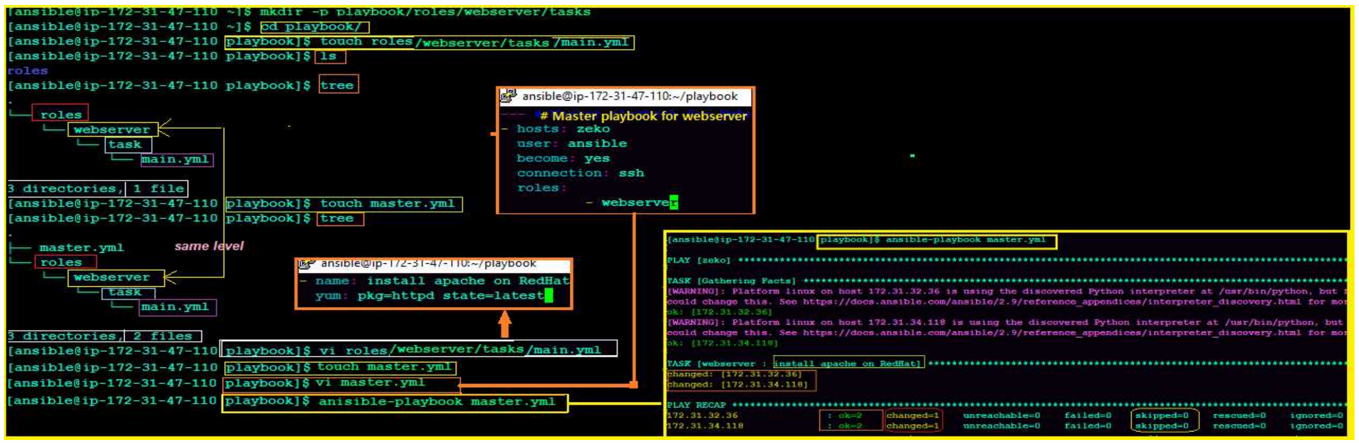
```
[playbook]$ vi roles/webserver/tasks/main.yml

- name: install apache on RedHat
  yum: pkg=httpd state=latest
```

```
:wq
```

```
[playbook]$ touch /master.yml
[playbook]$ vi master.yml

--- # Master playbook for Webserver
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
```

```
roles:
        - webserver                                        :wq
```

```
[playbook]$ ansible-playbook master.yml
```



```
[ansible@ip-172-31-47-110 ~]$ mkdir -p playbook/roles/webserver/tasks
[ansible@ip-172-31-47-110 ~]$ cd playbook/
[ansible@ip-172-31-47-110 playbook]$ touch roles/webserver/tasks/main.yml
[ansible@ip-172-31-47-110 playbook]$ ls
roles
[ansible@ip-172-31-47-110 playbook]$ tree
.
└── roles
    └── webserver
        └── task
            └── main.yml

3 directories, 1 file
[ansible@ip-172-31-47-110 playbook]$ touch master.yml
[ansible@ip-172-31-47-110 playbook]$ tree
.
├── master.yml
└── roles
    └── webserver
        └── task
            └── main.yml

3 directories, 2 files
[ansible@ip-172-31-47-110 playbook]$ vi roles/webserver/tasks/main.yml
[ansible@ip-172-31-47-110 playbook]$ touch master.yml
[ansible@ip-172-31-47-110 playbook]$ vi master.yml
[ansible@ip-172-31-47-110 playbook]$ anisible-playbook master.yml
```

```
ansible@ip-172-31-47-110:~/playbook
--- # Master playbook for webserver
- hosts: zeko
  user: ansible
  become: yes
  connection: ssh
  roles:
          - webserver
```

```
ansible@ip-172-31-47-110:~/playbook
- name: install apache on RedHat
  yum: pkg=httpd state=latest
```

```
[ansible@ip-172-31-47-110 playbook]$ ansible-playbook master.yml

PLAY [zeko] ***********************************************************************

TASK [Gathering Facts] ************************************************************
[WARNING]: Platform linux on host 172.31.32.36 is using the discovered Python interpreter at /usr/bin/python, but
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for mo
ok: [172.31.32.36]
[WARNING]: Platform linux on host 172.31.34.118 is using the discovered Python interpreter at /usr/bin/python, but
could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for mo
ok: [172.31.34.118]

TASK [webserver : install apache on RedHat] ***************************************
changed: [172.31.32.36]
changed: [172.31.34.118]

PLAY RECAP ************************************************************************
172.31.32.36               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.34.118              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

After execution of this master playbook pkg has installed in all nodes . . . . . The end

Jenkins

CI/CD *Pipelines…*

CI/CD: Continuous integration Continuous delivery (Deployment) is a type of methodology.

It is an automated process, Whenever developers write code, we integrate all that codes of all developers at that point of time and we build test and delivery/deploy to the client this process is called CI/CD.

[Continues integration = Continuous build + Continuous test]

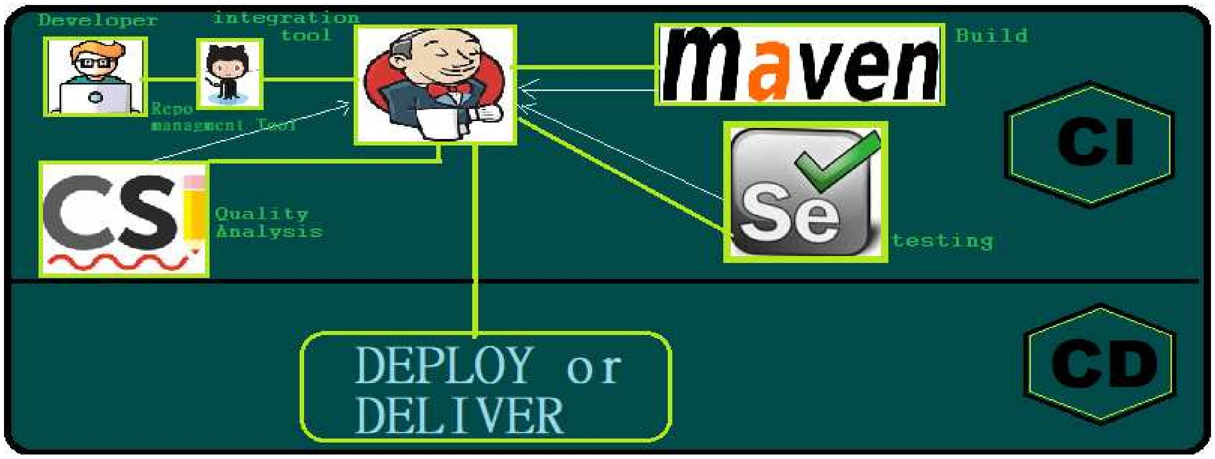
Jenkins : Integration tool

Jenkins is a open source project written in java, works on port 8080.
Jenkins automate the entire software development life cycle.
The project's name was Huston later named jenkins when oracle bought from sun microsystem.
It can run any on any major platform without any compatibility issue.
Because of CI continuous integration bugs will be reported fast and get rectified fast so the entire aoftware development happened fast.

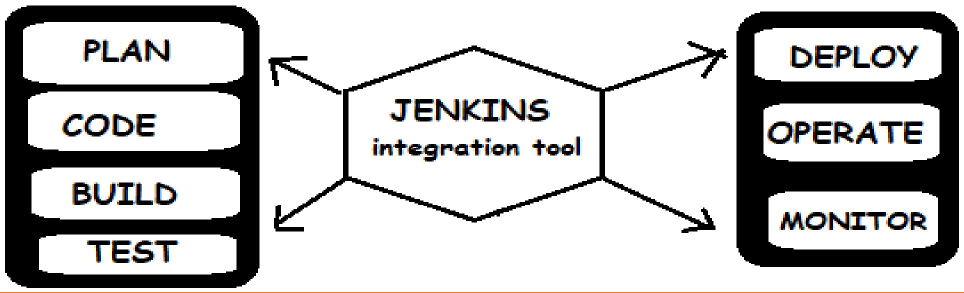
Jenkins Advantages:

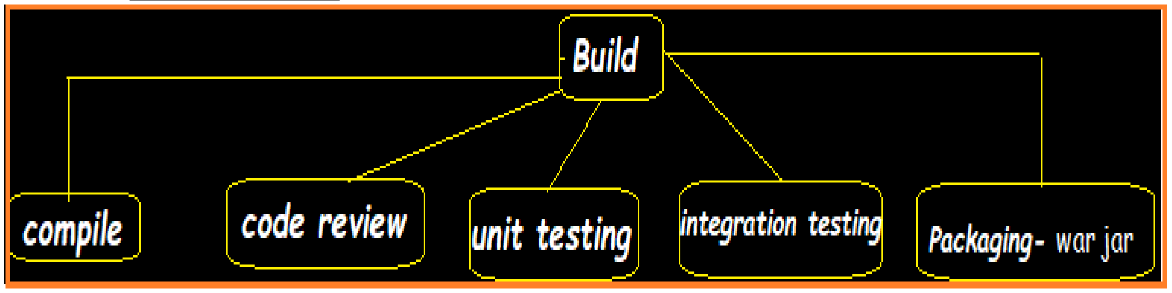
It has lots of plug-ins available.you can write you own plug-in,you can use community plug-in.
Its not just a tool it is a framework, you can do whatever you want all you need is just plug-ins.
We can attach slaves (nodes) to Jenkins master, it instruct other slaves (nodes) to do job. If slaves are not available Jenkins itself does the job. It can create labels, assign work to slave no.)
Jenkins also behaves as crave server replacement means, it can do scheduled tasks.

Workflow of Jenkins :



We can attach Git selenium and Artifactory plugins to Jenkins, once developer's puts code in git hub, Jenkins pull that code to maven for built.
When built is done Jenkins pulls that code and send to selenium for testing ,once testing is done ,then Jenkins will pull that code and sent to Artifactory for as per requirements and so on.
We can also deploy with Jenkins.





How to install Jenkins:

Create Ec2 instance with security ALL TRAFFIC.

Download Java `# yum install java* -y`

Grab commands from Jenkins website and paste on terminal

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

rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
```

Start jenkins `# systemctl start jenkins`
Enable jenkins `# systemctl enable jenkins`

Paste the public Ip of Jenkins server on web with: 8080 **port no.**
Then you will get a path, Copy that path and paste on Jenkins server's terminal with cat.

cat /var/lib/jenkins/secrets/initialAdminPassword

Then you will get a password on terminal like...>>**60642af835f94d3b8e208806036d198c**
Just paste this password on Jenkin's Administrator password Fill Aria.
After that you can install plugins and use easily.

kubernetes *Beginner*

DEFINITION*:* *Kubernetes is an open source container management tool, which automates container deployment, container scaling and load balancing.*

It schedules runs and manages isolated containers which are running on virtual/physical/cloud machines.

Online platforms: Kubernetes playground,
Play with K8s classroom
Play with Kubernetes k8s

K8s installation tools: kubeadm & minicube

FEATURES:

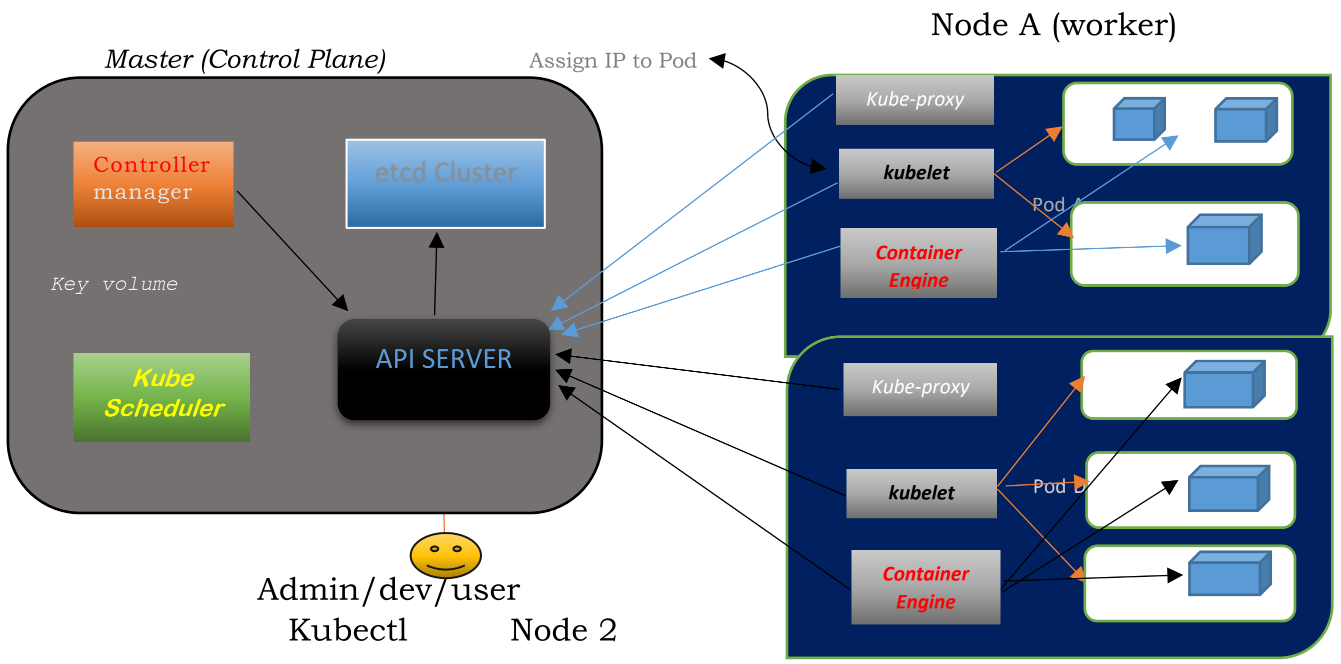
- Orchestration (clustering no of containers running on different network)
- Auto Scaling
- Auto-healing
- Load balancing
- Platform Independent (cloud/virtual/physical)
- Fault tolerance (node/pod failure)
- Roll Back (going back to previous version)
- Healthy monitoring & Containers
- Batch execution (one time sequential parallel).

Comparisons these according to their features:

Features	Kubernetes	Docker-swarm
Installation and cluster configuration	Complicated and Time consuming	Fast and Easy

Supports	Work with almost all Containers like rocket Docker	Work with Docker ONLY
GUI	Available	Not Available
Data volume	Only shared with the containers with same POD	Can be shared with any other container
Update and Roll back	Process Scheduling to maintain service while updating	Progressive updates & service health Monitoring throughout update
Auto scaling	Support vertical and Horizontal Auto scaling	Doesn't support Auto Scaling
Logging & Monitoring	Inbuilt tool present for monitoring	Used 3rd party tools like splunk

Architecture of Kubernetes:



Node is going to run to 3 important piece of software process.

KUBELET:

- Agent running on the node
- Listen to k8s master (ex: pod creation request)
- Use port 10255
- Sends success/fail report to master.

CONTAINER ENGINE:

- Works with kubeles
- Pulling images
- Start /stop containers
- Exposing container on port specified in manifest.

KUBE-PROXY:-

- Assign IP to each pod
- It is required to assign IP address to Pods (dynamic)
- Kube-proxy runs on each node and this make sure
  that each pod will get its own unique IP address.

Working with Kubernetes:

- We create manifest (Jason .yml)
- Apply this cluster to master (to master) to bring into desired state.
- Pods runs on node which is controlled by master.

Role of master node:

- Kubernetes cluster contains running on bare metal/VM instance /Cloud instances/All mix.
- Kubernetes designates one or more of these as master and all others are workers.
- The master is now going to run set of k8s process. These process will insure smooth functioning of cluster these process are called Control plane.
- It can be multi master for high availability.
- Master runs control plane to run cluster smoothly.

Components of Control Plane master:

Kube API server: - (*for all communications*)

This interacts directly with user

(*If we applied **Jason** or **.YML** manifest to kube API server).*

The kubeAPI server is meant to scale automatically as per load.

Kube API server is front end of control-plane.

etcd:-

It stores metadata and status of cluster.

It is consistent and H-A store (*key volume store*)

Source of Touch for cluster state. (*Information about cluster's state*).

etcd features:-

Fully replicated,
Secure > Implements automatic TLS with optional client certificate

Fast > Benchmarked at 10,000 writes per second.

Kube-Scheduler:-

When user make request for the creation and management of PODS kube scheduler is going to take actions on these requests.

Handled POD creation and management.
It match/assign any node to create and run pods.
A scheduler watches for newly created pods that have no node assigned for every pod that the scheduler discovers, the scheduler becomes responsible for finding best node for that POD to run ON.
Scheduler gets the information for hardware configuration from configuration files and schedules the PODS on nodes accordingly.

Controller manager:-

It makes sure actual state of cluster matches to desired state.

Two possible choices for controller manager:

If k8s on cloud then it will be cloud controller manager
If k8s on non-cloud, then it will be kube-controller manager.

Components on Master that runs controller:

Node–controller: - For checking the cloud provider to determine if a node has been detected in the cloud after it stops responding.

Route Controller: - Responsible for setting up network routes on your cloud.

Service controller: - Responsible for load balancers on your cloud against service of type load balancer.

Volume controller: - for creating attaching and maintaining volumes and interacting with the cloud provider to orchestrate volume.

POD: -

Smallest unit in kubernetes (*usually contains 1 container*).

It is a group of one or more container that are deployed together on the dame host.

A cluster is a group of nodes which has at least 1 master and 2 worker nodes.

In K8s Pod is the control unit not the container.

Pod runs on node which is control by master.

K8s communicates with pods not container.

Without POD we cannot start containers.

Multi-Container Pod:-

Share access to memory space.

Connect to each other using local host <*container host*>

Share access to the same volume

Container within pod are deployed in An, All or Nothing manner.

Entire pod is hosted on the same node (*scheduler will decide about node*).

Pod limitations: -

No auto healing and scaling

Pod creases

Higher Level K8s Objects:-

Deployment: Versioning and Rollback

Replication set: Scaling and healing

Service: Static (*non-ephemeral)* IP networking

Volume: Non ephemeral storage

Set up of K8S master and worker node on AWS:

Minimum requirement for master is 4 GB RAM and 2CPU.

Create 3 instances (Ubuntu 16.04 t2 medium) 1 for master 2 for nodes.

Commands for master and nodes:

```
sudo su
```

```
apt-get update
```

```
apt-get install apt-transport-https
```

This https is needed for intra cluster communication
(Particularly from control plane to individual pods).

Now install Docker on all 3 instances:

```
apt install docker.io -y
docker –version
```

To check whether Docker is installed or not?

```
systemctl start docker
systemctl enable docker
```

Set up open GPG key this is required for intra cluster communication it will be added to source key on this node when K8s sends singed info's to our host, it is going to accept those information because this open GPG key is present in the source key.

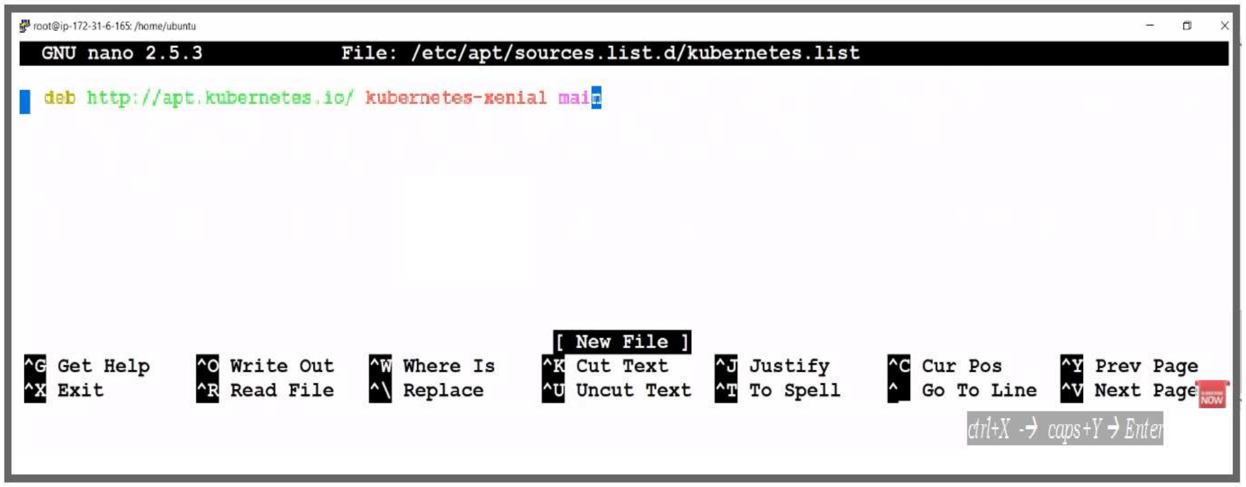
```
sudo curl -s https://packages.cloud.google.com/apt... | sudo apt-key add
```

Paste this on all three instances master node, node 1 and node 2.

Edit source list file (**apt-get-install nano**)

Create nano file, Go inside and paste this (Xenial) command in side all nodes

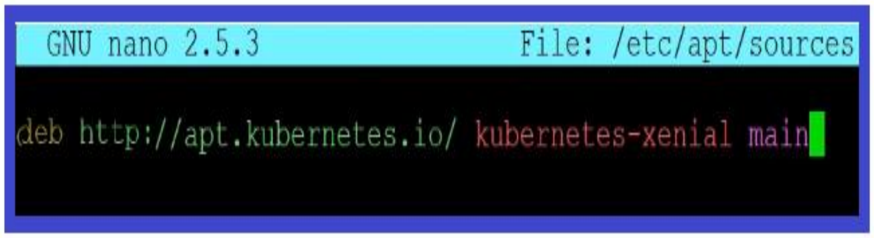
```
# nano /etc/apt/sources.list.d/kubernetes.list
# deb http://apt.kubernetes.io/ kubernetes-xenial main
```



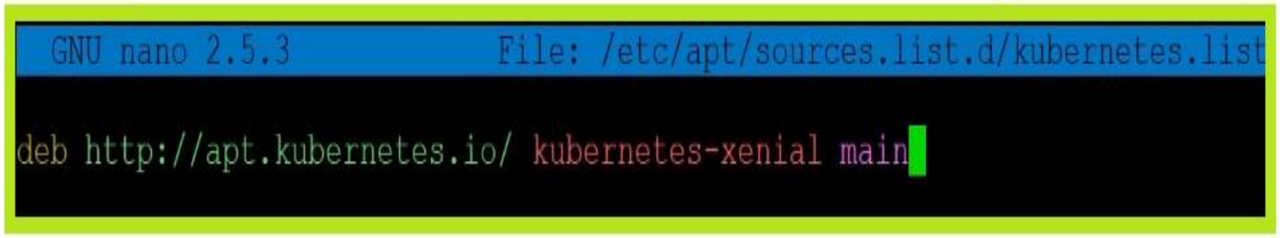
```
root@ip-172-31-6-165: /home/ubuntu
  GNU nano 2.5.3              File: /etc/apt/sources.list.d/kubernetes.list

 deb http://apt.kubernetes.io/ kubernetes-xenial main

                                    [ New File ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     ^Y Prev Page
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  ^V Next Page
```

ctrl+X → caps+Y → Enter

Exit from nano .. *ctrl+X -→ caps+Y → Enter*



```
GNU nano 2.5.3                File: /etc/apt/sources

deb http://apt.kubernetes.io/ kubernetes-xenial main
```



```
GNU nano 2.5.3          File: /etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main
```

For getting update after closing the Nano editor.

```
# apt-get update
```

Install all package on All 3 nodes

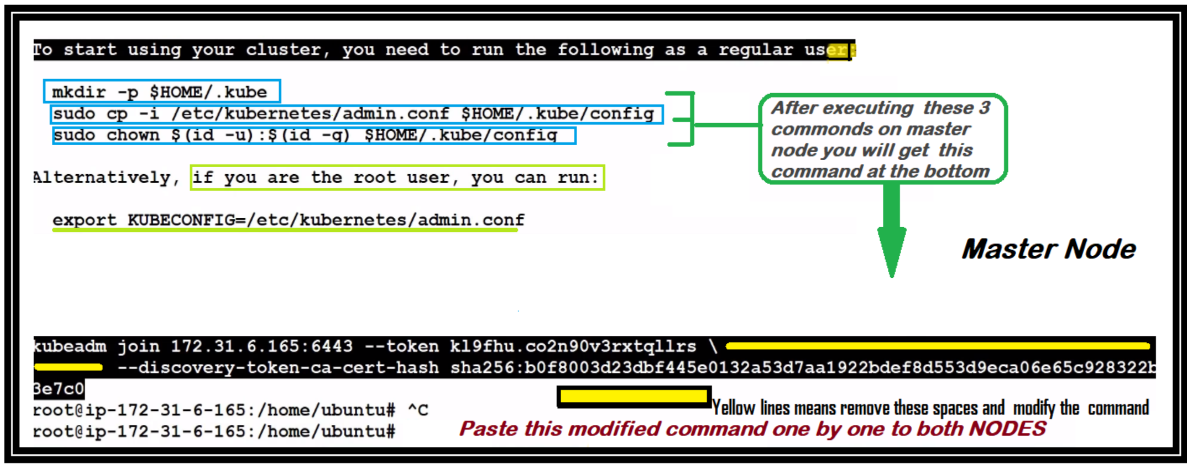
```
# apt-get install -y kubelet kubeadm kubectl kubernetes-cni
```

BOOTSTRAPPING IN THE MASTERNODE (in Master)

To initialize kubernetes cluster:

```
# kubeadm init
```

Then you will get one long command started from **"kubeadm join 172.31.6.265:6443** .... ***Copy the command and save on notepad***



```
To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

kubeadm join 172.31.6.165:6443 --token kl9fhu.co2n90v3rxtqllrs \
        --discovery-token-ca-cert-hash sha256:b0f8003d23dbf445e0132a53d7aa1922bdef8d553d9eca06e65c928322b
3e7c0
root@ip-172-31-6-165:/home/ubuntu# ^C
root@ip-172-31-6-165:/home/ubuntu#
```

After executing these 3 commonds on master node you will get this command at the bottom
Master Node
Yellow lines means remove these spaces and modify the command
Paste this modified command one by one to both NODES

Run this command in to nodes, then nodes will connect to the master

Create both **.Kube** and its parent directories (-p)

```
# Mkdir -p $HOME/.kube
```

Copy configuration to kube directory (un-configured file):

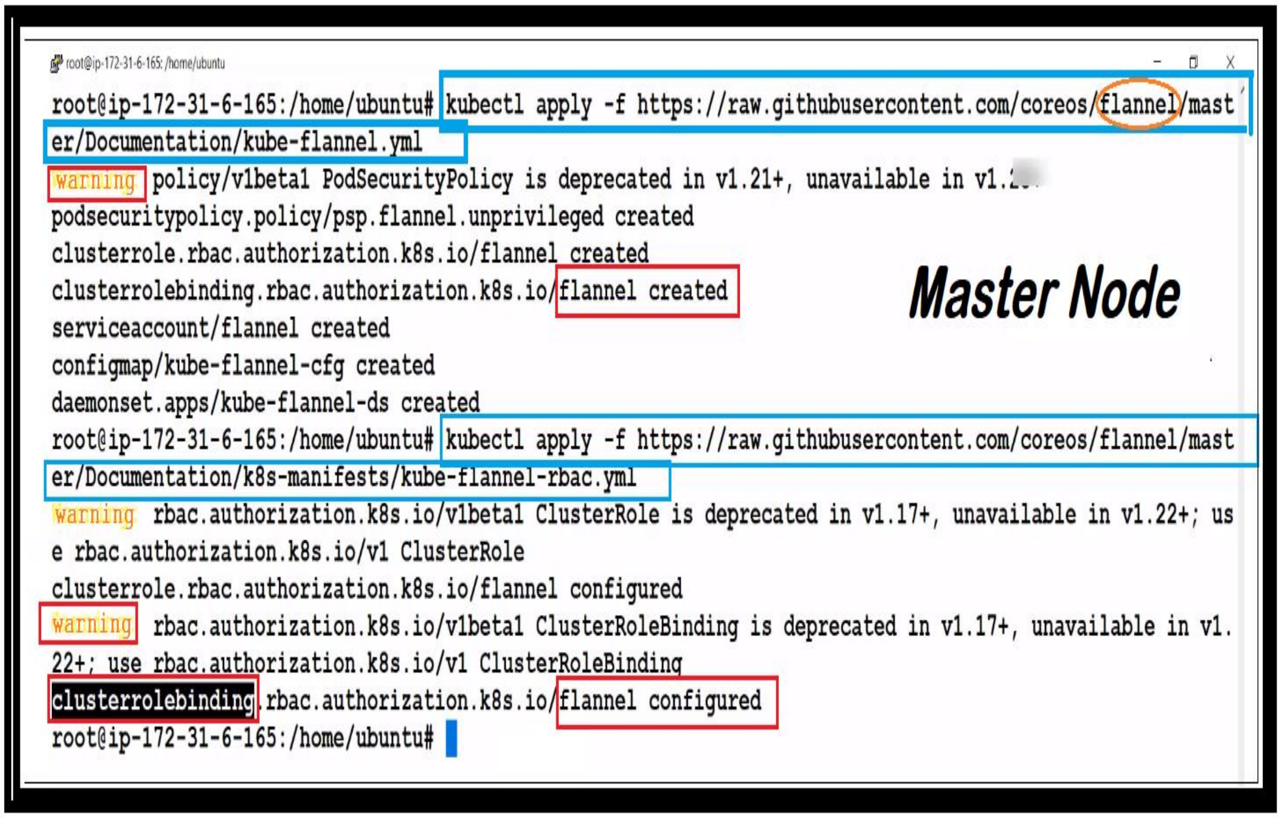
```
# mkdir -p $HOME/.kube
# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

Provide user permission to config file:

```
#chown $(id -u):$(id -g) $HOME/.kube/config
```

Deploy FLANNEL node network for its repository Path.
Flannel is going to place a binary in each node.
Cluster role binding /flannel creation/flannel Configured.

```
#kubectl apply -f https://raw.githubusercontent.com/cor...
#kubectl apply -f https://raw.githubusercontent.com/cor...
```

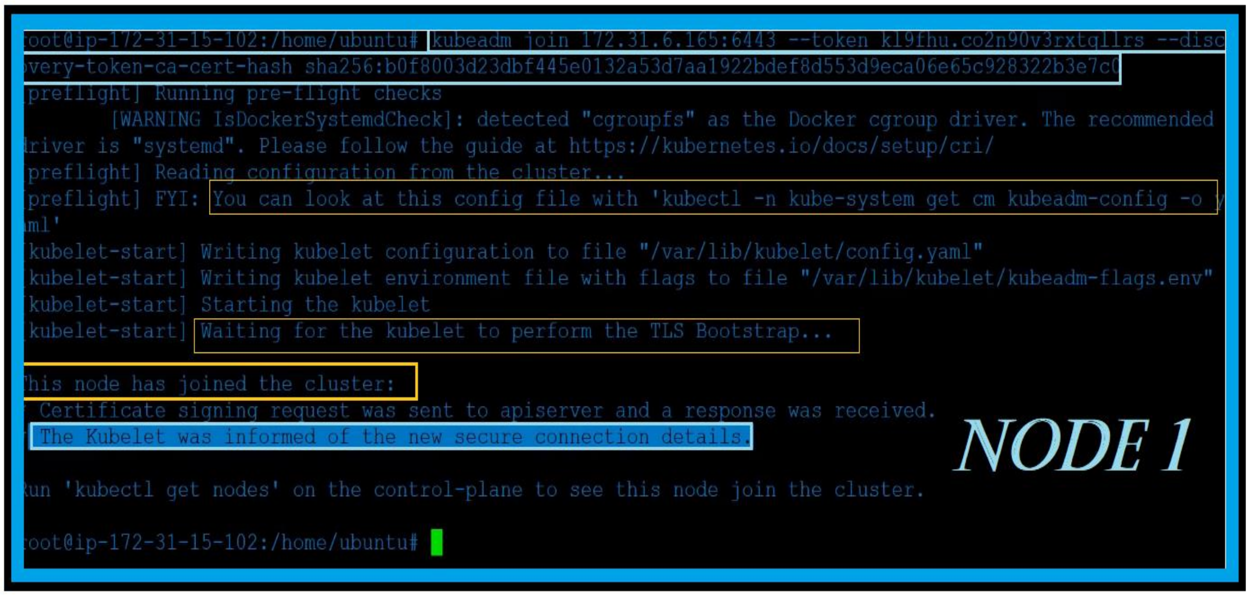


```
root@ip-172-31-6-165:/home/ubuntu# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/mast
er/Documentation/kube-flannel.yml
warning policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.2
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
root@ip-172-31-6-165:/home/ubuntu# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/mast
er/Documentation/k8s-manifests/kube-flannel-rbac.yml
warning rbac.authorization.k8s.io/v1beta1 ClusterRole is deprecated in v1.17+, unavailable in v1.22+; us
e rbac.authorization.k8s.io/v1 ClusterRole
clusterrole.rbac.authorization.k8s.io/flannel configured
warning rbac.authorization.k8s.io/v1beta1 ClusterRoleBinding is deprecated in v1.17+, unavailable in v1.
22+; use rbac.authorization.k8s.io/v1 ClusterRoleBinding
clusterrolebinding.rbac.authorization.k8s.io/flannel configured
root@ip-172-31-6-165:/home/ubuntu#
```

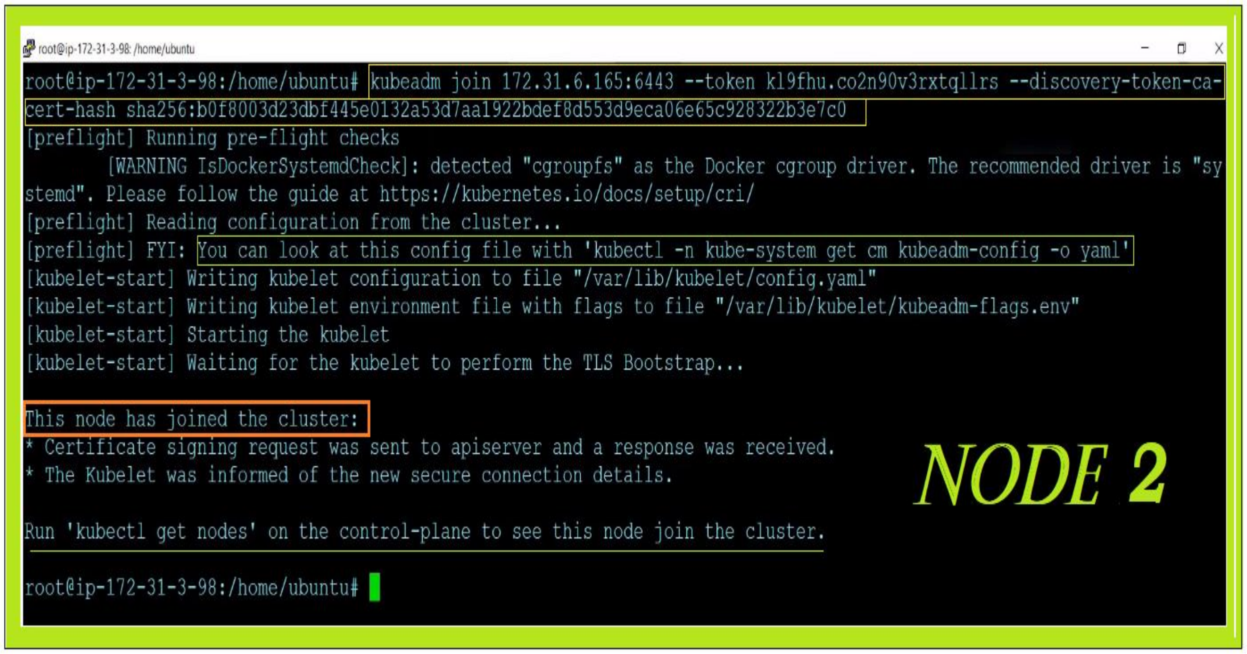
Configuration worker node

Paste long command (provided by master) in both the nodes

e.g- *kubeadm join 172.31.6.165:6443 --token kl9fhu.co2n90v3rxtqllrs --discovery-token-ca-cert-hash sha256:b0f8003d23dbf445e0132a53d7aa1922bdef8d553d9eca06e65c928322b3e7c0*



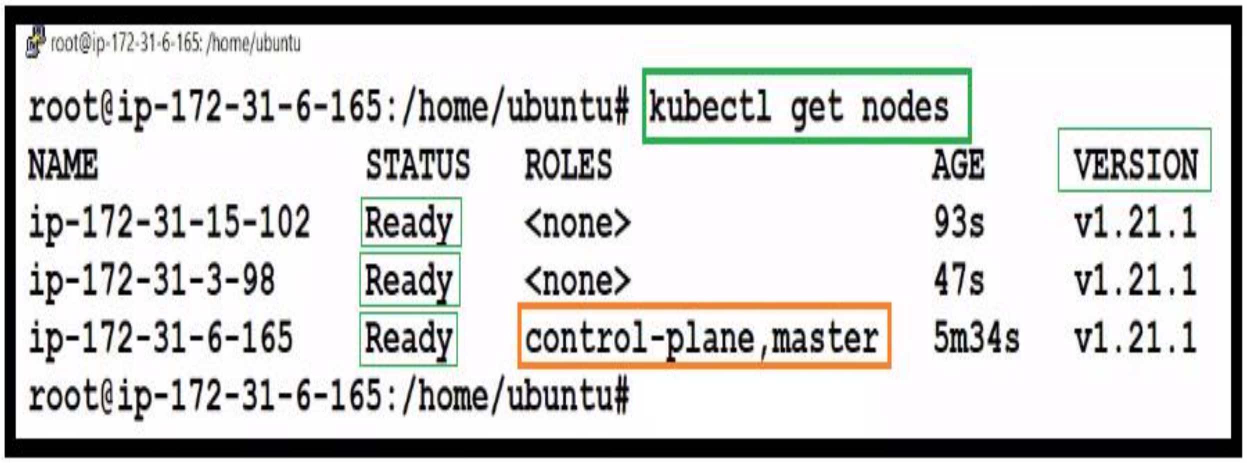
```
oot@ip-172-31-15-102:/home/ubuntu# kubeadm join 172.31.6.165:6443 --token kl9fhu.co2n90v3rxtqllrs --disc
very-token-ca-cert-hash sha256:b0f8003d23dbf445e0132a53d7aa1922bdef8d553d9eca06e65c928322b3e7c0
preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended
river is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
preflight] Reading configuration from the cluster...
preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o y
ml'
kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
kubelet-start] Starting the kubelet
kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

his node has joined the cluster:
 Certificate signing request was sent to apiserver and a response was received.
 The Kubelet was informed of the new secure connection details.

un 'kubectl get nodes' on the control-plane to see this node join the cluster.

oot@ip-172-31-15-102:/home/ubuntu#
```



```
root@ip-172-31-3-98:/home/ubuntu# kubeadm join 172.31.6.165:6443 --token kl9fhu.co2n90v3rxtqllrs --discovery-token-ca-
cert-hash sha256:b0f8003d23dbf445e0132a53d7aa1922bdef8d553d9eca06e65c928322b3e7c0
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "sy
stemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@ip-172-31-3-98:/home/ubuntu#
```

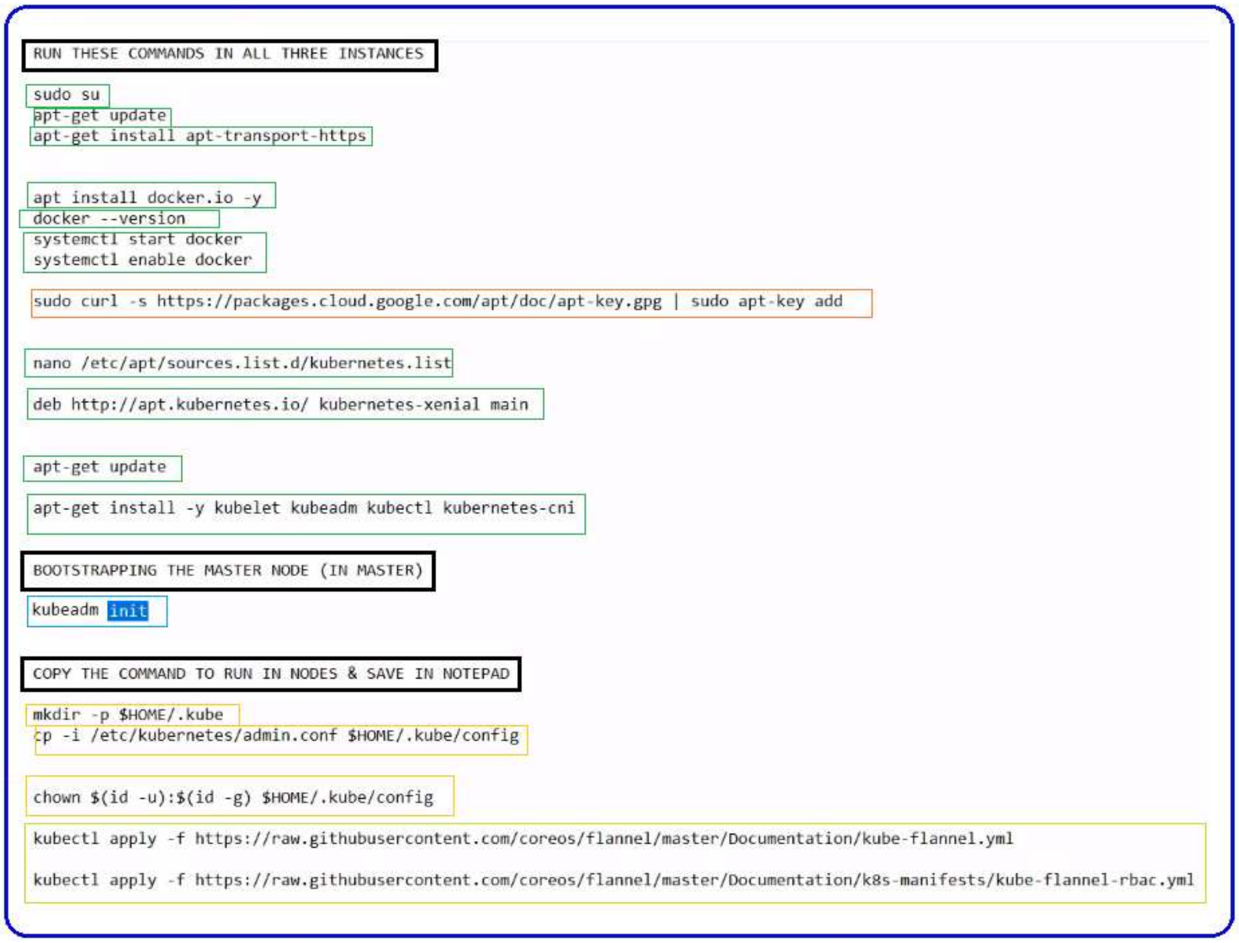
To check the status of nodes, Go to master and run this command

```
Kubectl get node
```



```
root@ip-172-31-6-165: /home/ubuntu
root@ip-172-31-6-165:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES                  AGE     VERSION
ip-172-31-15-102   Ready    <none>                 93s     v1.21.1
ip-172-31-3-98     Ready    <none>                 47s     v1.21.1
ip-172-31-6-165    Ready    control-plane,master   5m34s   v1.21.1
root@ip-172-31-6-165:/home/ubuntu#
```

All commands in one frame given by BR sir ……



```
RUN THESE COMMANDS IN ALL THREE INSTANCES

sudo su
apt-get update
apt-get install apt-transport-https

apt install docker.io -y
docker --version
systemctl start docker
systemctl enable docker

sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add

nano /etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

apt-get update

apt-get install -y kubelet kubeadm kubectl kubernetes-cni

BOOTSTRAPPING THE MASTER NODE (IN MASTER)

kubeadm init

COPY THE COMMAND TO RUN IN NODES & SAVE IN NOTEPAD

mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml
```

Dua me yad Rakhiyega........

Bhupinder Rajput l (Technical Guftgu)
भूपिंदर राजपूत l بھوپندر راجپوت

Founder at Technical Guftgu l
Youtuber | EdupreneurTalks about #devops #awscloud, #awstraining, #microsoftazure, #technicalguftgu
North Delhi, , India +917819848195
LinKdin :linkedin.com/in/bhupinder-rajput
Email: technicalguftgu99@gmail.com
Youtube: https://www.youtube.com/c/TechnicalGuftgu/playlists
Twitter: https://twitter.com/BhupinderRajp11?s=09