1. **/etc/ansible directory  
     
   [ansitest@basuritam22c ansible]$ ls -lrt**

**total 24**

**drwxr-xr-x. 2 root root 6 Dec 14 05:57 roles 🡨 default roles path**

**-rw-r--r--. 1 root root 1016 Dec 14 05:57 hosts 🡨 default file for hosts**

**-rw-r--r--. 1 root root 20277 Dec 14 05:57 ansible.cfg <- the default config file**

**[ansitest@basuritam22c ansible]$ pwd**

**/etc/ansible  
  
1.2) Configuration for password less communication**

1. **HOSTS FILE :**

**2.1) Grouping your hosts in the hosts file :**

**[local]**

**Localhost**

**Localhost.Localdomain**

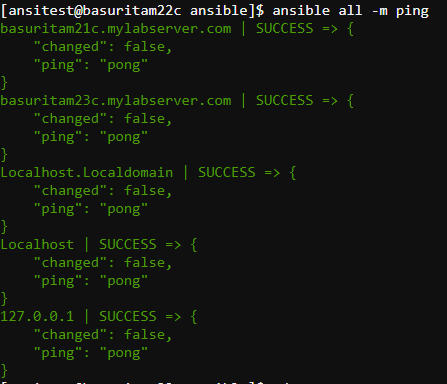
**127.0.0.1**

**[webhosts]**

**basuritam21c.mylabserver.com**

**[apphosts]**

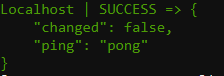
**basuritam23c.mylabserver.com**

**Now if I try to ping all hosts it will ping all entries.  
  
**

**So, it’s better to have one single entry for one server.**

**2.2) Overriding the HOSTS File  
  
remember we have 3 entries in [local] in my default hosts file at /etc/ansible/hosts.**

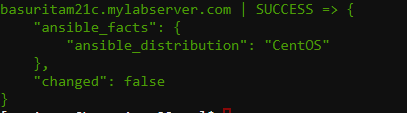
**2.2.1) via command line .  
  
pwd 🡪 /home/ansitest**$ ansible local -i /home/ansitest/exps/hosts -m ping

**-I <path\_to\_hosts\_file>  
  
2.2.2) From ansible.cfg**#inventory = /etc/ansible/hosts **old**  
  
inventory = /home/ansitest/exps/hosts  **new**   
 **and do 🡪** $ ansible local -m ping  
 **for both the cases O/P:**  
  


**3) Overriding the ansible.cfg file :  
  
Priority , where ansible looks for ansible.cfg  
  
A 🡪 export ANSIBLE\_CONFIG=/home/ansitest/exps/ansible.cfg   
 define ANSIBLE\_CONFIG var.  
  
B 🡪 Current Directory   
  
C 🡪 ~/.ansible.cfg ] remember .ansible.cfg at usre’s home dir  
  
D 🡪 /etc/ansible/ansible.cfg**

1. **PATH to your first PLAYBOOK.  
     
   3.1) Executing from command Line.  
   Some Examples.  
     
   3.1.1) List all your hosts from ansible command line  
   the python packages installed on hosts webhosts:**ansible webhosts -b -m shell -a 'yum list installed | grep python'

**-b, --become run operations with become (does not imply**

**password prompting)  
  
-m 🡪 Invoking module  
-a 🡪 passing arguments   
  
3.1.2) gather system facts of one host and save it to a file .**ansible webhosts -m setup --tree ~/webhosts\_facts  
  
**setup 🡪 module is used to gather\_facts**  
**Output will be json formatted data.**  
 **Filer some values .**ansible webhosts -m setup -a "filter=ansible\_distribution"  
  
****

**3.2) Command Line to playbook conversion :  
  
Install lynx package on webhosts   
  
command line 🡪** ansible webhosts -b -m yum -a "name=lynx state=installed update\_cache=true"

playbook1)

- **hosts: webhosts**

**tasks:**

**- name: Install lynx on webserver**

**yum: pkg=lynx state=installed update\_cache=true**

****

**GREEN 🡪 NO CHANGE  
YELLOW 🡪 CHANGE  
RED 🡪 FAILED**

**4) PLAYBOOK SECTION:  
4.1) TARGET SECTION :  
  
Edit the playbook and add the following to the target section we just created:**

**- Force use of SSH connections**

**- Always run the playbook as the user 'ansitest'**

**- Run this playbook as SUDO by default**

**- Do not use the setup module to gather facts from systems during execution  
  
  
target Section :**

**--- # Target Section**

**- hosts: apphosts**

**connection: ssh**

**user: ansitest**

**sudo: yes**

**gather\_facts: no**

**4.2) VARIABLE SECTION:  
  
3 types of variables  
  
a) Defined inside the playbook   
b) Defined outside of playbook in a separate file  
c) Assigned dynamically during playbook runtime   
  
4.2.1) Add 3 types of variables in your playbook .  
we have  
 ├── conf**

**├── copyright.yml**

**└── webdefaults.yml files containing var values**

**--- # Variable Section**

**- hosts: apphosts**

**connection: ssh**

**user: ansitest**

**sudo: yes**

**gather\_facts: no**

**vars:**

**playbook\_version: 0.1b**

**vars\_files:**

**- conf/copyright.yml**

**- conf/wbdefaults.yml**

**vars\_prompt:**

**- name: web\_domain**

**prompt: Web Domain-**

* **Conf   
  remember the ‘-‘  
    
  everything , referenced or sourced from outside the playbook should proceed with a  
  ‘-‘**

**4.2.2) Using variable:  
  
- enter author name as a variable and show it in the O/p.  
- prompt a package name from user and install in on the webhosts.  
  
private: yes is the default value  
  
no 🡪 displays the user input.**

**--- #Using variables**

**- hosts: webhosts**

**user: ansitest**

**sudo: yes**

**connection: ssh**

**gather\_facts: no**

**vars:**

**author: Ritam**

**vars\_prompt:**

**- name: pkg\_name**

**prompt: Enter Package name -**

**private: no**

**tasks:**

**- name: show author name**

**debug: msg="Written by -- {{ author }}"**

**- name: Install desired package**

**yum: name="{{ pkg\_name }}" state=present**



**4.3) TASK SECTION :**

**tasks:**

**- name: Install lynx on webserver**

**yum: pkg=lynx state=installed update\_cache=true**

**4.4) NOTIFY & HANDLERS:  
  
HANDLERS SECTION:**

Pr)   
**- Uses SSH**

**- Logs in to the remote system as 'ansitest' user**

**- Connects to one server or group from Step #2 above**

**- The playbook runs as 'sudo'**

**- Skip gathering remote facts**

**- Installs the 'Apache Web Server' using the appropriate package module**

**- Upon installation of the web server, notifies the appropriately titled handler to restart the service using the 'service' module.**

**--- # Handler Section**

**- hosts: webhosts**

**connection: ssh**

**user: ansitest**

**sudo: yes**

**gather\_facts: no**

**tasks:**

**- name: install httpd**

**yum: name=httpd state=installed**

**notify: Restart httpd**

**handlers:**

**- name: Restart httpd**

**action: service name=httpd state=restarted**



**Remember to give same name for notify and handlers section.**

**5) Loop:**

**5.1)  
- create a group test  
- Add 3 new devs , test1, test2, test3 user to group test apphosts using loop.  
[basic loop]**

**--- #Loop basic**

**- hosts: apphosts**

**user: ansitest**

**sudo: yes**

**connection: ssh**

**gather\_facts: no**

**tasks:**

**- name: create test group**

**group: name=test state=present**

**- name: add users**

**user: name="{{ item }}" groups=test**

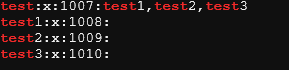
**loop:**

**- test1**

**- test2**

**- test3**





**5.2) Adding different users to different groups:  
[ double\_loop ]**

**--- #Loop double**

**- hosts: apphosts**

**user: ansitest**

**sudo: yes**

**connection: ssh**

**gather\_facts: no**

**tasks:**

**- name: add different users to different groups**

**user: name="{{ item.name }}" state=present groups="{{ item.groups }}"**

**loop:**

**- { name: 'dev1', groups: 'dev' }**

**- { name: 'test4', groups: 'test' }**

**5.3) Install 3 packages on apphosts .  
 create a file that has the package name   
   
cat ~/conf/pkg\_list.yml  
pkg\_list:**

**--- #var outside**

**- hosts: apphosts**

**user: ansitest**

**sudo: yes**

**connection: ssh**

**gather\_facts: no**

**vars\_files:**

**- ~/conf/pkg\_list.yml**

**tasks:**

**- name: install packages**

**yum: name="{{ pkg\_list }}" state=present**

**- lynx**

**- tree**

**- telnet**

**5.4) create master and admin user   
 add them to dev, test and wheel group   
[ complex loop ]**

**--- #loop complex**

**- hosts: apphosts**

**user: ansitest**

**sudo: yes**

**connection: ssh**

**gather\_facts: no**

**vars:**

**users:**

**- master**

**- admin**

**tasks:**

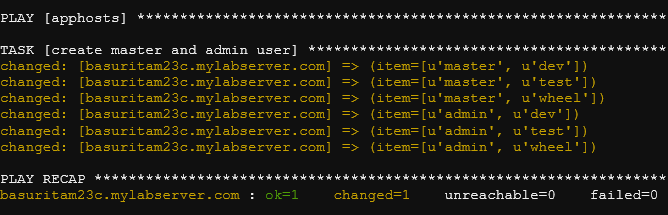
**- name: create master and admin user**

**user: name="{{ item[0] }}" state=present groups="{{ item[1] }}"**

**with\_nested:**

**- "{{ users }}"**

**- [ 'dev', 'test', 'wheel' ]**



**6) WHEN CONDITION:**

**- Installs the 'Apache Web Server' using the appropriate package module**

**- Each section will determine WHICH Apache package to install based on the value of the fact called 'ansible\_os\_family' and using the appropriate module for package management**

**--- # when condition**

**- hosts: apphosts**

**user: ansitest**

**sudo: yes**

**connection: ssh**

**tasks:**

**- name: Install Apache Appropriate to the Distribution Type (Debian/Ubuntu)**

**command: apt-get -y install apache2**

**when: ansible\_os\_family == "Debian"**

**- name: Install Apache Appropriate to the Distribution Type (RedHat/CentOS)**

**command: yum -y install httpd**

**when: ansible\_os\_family == "RedHat"**

**Another way to apply the condition**

**tasks:**

**- name: install apache on redhat**

**yum: name=httpd state=installed**

**when: ansible\_facts['os\_family'] == "RedHat"**

**- name: Install apache on debian**

**apt: name=apache2 state=present**

**when: ansible\_facts['os\_family'] == "Debian"**

**7) Using DEBUG:  
7.1) start time and end time of your playbook**

**tasks:**

**- name: start time**

**command: date**

**register: start\_time**

**- debug: var=start\_time**

**- name: install apache**

**yum: name=httpd state=installed**

**- name: end time**

**command: date**

**register: end\_time**

**- debug: var=end\_time**

**8) Asyncronous Polling  
  
To avoid blocking or timeout issues, you can use asynchronous mode to run all of your tasks at once and then poll until they are done.**

**To launch a task asynchronously, specify its maximum runtime and how frequently you would like to poll for status. The default poll value is 10 seconds if you do not specify a value for poll:**

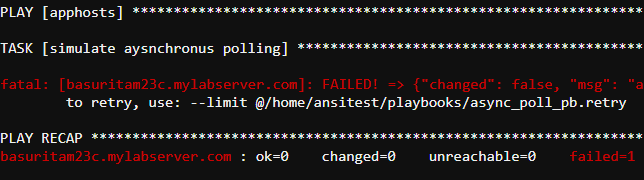
**tasks:**

**- name: simulate aysnchronus polling**

**command: sleep 15**

**async: 45**

**poll: 3**



**9) LOOKUP:  
  
9.1)Getting value of an env variable**

**- prompt user to type an env var , eg: HOSTNAME  
- use lookup to get the value of HOSTNAME from env variables**

**vars\_prompt:**

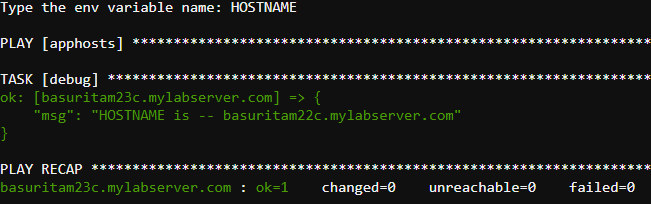
**- name: env\_var\_name**

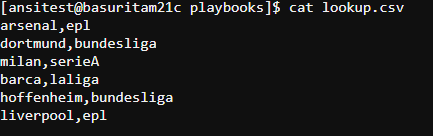
**prompt: Type the env variable name**

**private: no**

**tasks:**

**- debug: msg="{{ env\_var\_name }} is -- {{ lookup('env', "{{ env\_var\_name }}") }}"**



**9.2) use lookup to get values from a csv file**  
  


**tasks:**

**- debug: msg=" League for Arsenal -- {{ lookup('csvfile','arsenal file=lookup.csv delimiter=, col=1 default=NOMATCH') }}"**

**lookup('where\_to\_look','what\_to\_look file=file\_path delimiter=, col=1 default=NOMATCH')  
  
col=next\_column\_by\_default**

**10) RUN\_Once  
  
- check tree is present in all hosts  
- List the remote /var directory and pipe the output to the /home/test directory in a file called ‘var\_list.log’  
- use run\_once , that listing would be limited to first host only**

**--- # asynchronous polling**

**- hosts: all**

**connection: ssh**

**user: ansitest**

**sudo: yes**

**gather\_facts: no**

**tasks:**

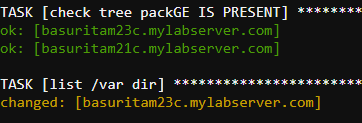
**- name: check tree packGE IS PRESENT**

**yum: name=tree state=present**

**- name: list /var dir**

**raw: ls -al /var > /home/ansitest/var\_list.log**

**run\_once: true**

**1st task executed on all hosts  
  
2nd task executed only on host 1 as run\_once is true**

**11) INCLUDE:  
Including tasks as plays from different files  
  
11.1) – write a task to uninstall httpd in a different file  
 - Include the file to create a playbook that uninstalls the package  
 - checks the package is uninstalled  
  
cat ./plays/package\_erase.yml  
  
 - name: Uninstall lynx package**

**yum: name=lynx state=absent**

**- name: Uninstall telnet package**

**yum: name=telnet state=absent  
  
12) Until:  
  
- In RedHat if you install httpd it does not automatically start.  
- write a playbook that executes until httpd is started**

**tasks:**

**- name: install httpd**

**yum: name=httpd state=installed**

**- name: check apache is up**

**shell: systemctl status httpd**

**register: apache\_stat**

**until: apache\_stat.stdout.find("active (running)") !=-1**

**retries: 3**

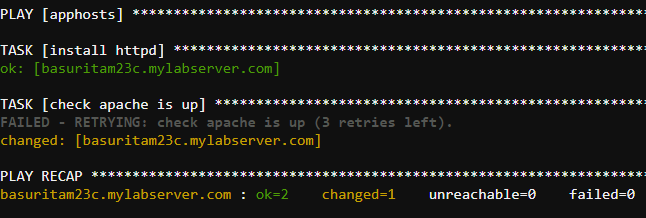
**delay: 2**

**tasks:**

**- include: plays/pkg\_erase.yml**

**- name: check telnet exists**

**raw: yum list installed | grep telnet > check\_pkg.log**

**it will retry 3 times with 2s delay   
  
**

**13) TAGS:  
  
13.1) Tagging your tasks:**

**tasks:**

**- include: plays/pkg\_erase.yml**

**tags:**

**pkg\_erase**

**- include: plays/pkg\_install.yml**

**tags:**

**pkg\_install**

**- name: verify the telnet package is installed**

**shell: yum list installed| grep telnet**

**register: pkg\_result**

**tags:**

**verification**

**- debug: var=pkg\_result**

**tags:**

**verification**

**Uses:**ansible-playbook tags\_pb.yml --tags "pkg\_install,verification"  
  
**a) it will execute 🡪 pkg\_install, then verification tasks  
b) see I have tagged verification twice, because the complete verification process comprises of both the tasks   
otherwise we will get**"pkg\_result": "VARIABLE IS NOT DEFINED!"  
  
ansible-playbook tags\_pb.yml --skip-tags "pkg\_erase"  
  
basically doing the same thing  
  
**if we want to execute a task always🡪  
  
 - name: verify the telnet package is installed**

**shell: yum list installed| grep telnet**

**register: pkg\_result**

**tags:**

**always**

**- debug: var=pkg\_result**

**tags:**

**always**

ansible-playbook tags\_pb.yml --tags "pkg\_install”  
  
**will do the same .**  
  
**14) Start\_at  
  
14.1)Starting playbook execution at any desired task.  
  
tasks defined as 🡪  
 tasks:**

**- name: Install Lynx**

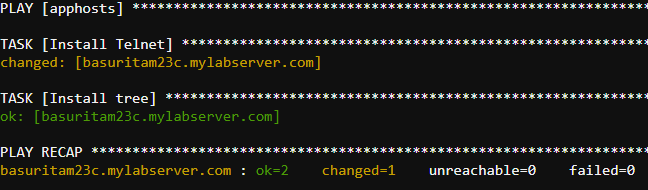
**yum: name=lynx state=present**

**- name: Install Telnet**

**yum: name=telnet state=present**

**- name: Install tree**

**yum: name=tree state=present**

**Example:** ansible-playbook start\_at\_pb.yml --start-at "Install Telnet"  
  
****

**It started from “Install Telnet” and skipped LYNX**

**14.2) Run the playbook, indicating that you should be prompted for each task run.**

ansible-playbook start\_at\_pb.yml –step  
  
**--step allows**Perform task: TASK: Install Lynx (N)o/(y)es/(c)ontinue:  
  
**y:** **execute task  
n: Do not execute  
c: continue executing all tasks from the present task**

**15.1) Delegate\_to**

**- hosts: apphosts**

**connection: ssh**

**user: ansitest**

**sudo: yes**

**gather\_facts: no**

**tasks:**

**- name: ping the app server**

**raw: ping -c 4 basuritam23c.mylabserver.com > /home/ansitest/ping.log**

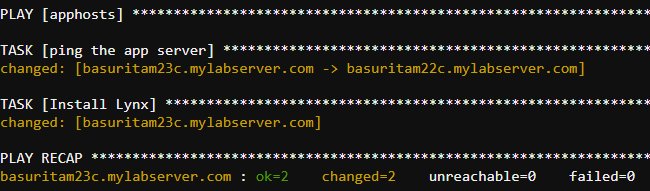
**delegate\_to: basuritam22c.mylabserver.com**

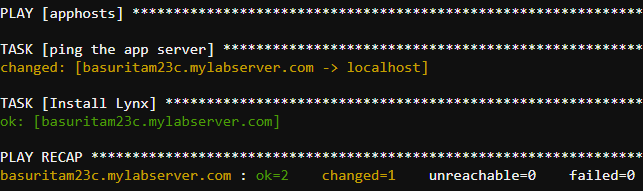
**- name: Install Lynx**

**yum: name=lynx state=present**

**We are executing from : basuritam21c  
apphosts: 23c  
webhosts: 22c  
  
now say , before executing any task on apphosts, we want to check if webhosts can ping apphosts.  
  
 raw: ping -c 4 basuritam23c.mylabserver.com > /home/ansitest/ping.log**

**delegate\_to: basuritam22c.mylabserver.com**

**so we delegetaed the pinging task to webhosts(22c)  
  
now the ping data will be stored in basuritam22c  
though we are executing the playbook on 23c  
  
**

**15.2) Local\_ACTION  
  
This is a special case of delegate\_to, where we want to delegate the task to localhost only.  
  
  
**

**tasks:**

**- name: ping the app server**

**local\_action: raw ping -c 4 basuritam23c.mylabserver.com > /home/ansitest/ping.log**

**- name: Install Lynx**

**yum: name=lynx state=present**

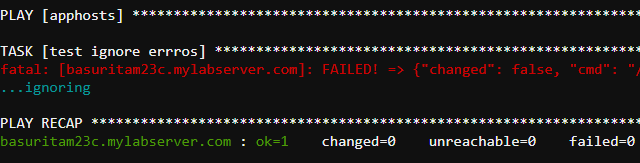
**16) Ignore\_errors:**

**tasks:**

**- name: test ignore errros**

**command: /bin/porn**

**ignore\_errors: yes**

****

**17)Creating jinja2 template;  
  
cat test.conf.j2**

**# Configuration for our custom widget**

**ConnectionType {{ connectionType }}**

**Username {{ userName }}**

**Password {{ userPassword }}**

**DistributionType {{ ansible\_os\_family }}**

**Playbook:**

**gather\_facts: yes**

**vars:**

**userName: test**

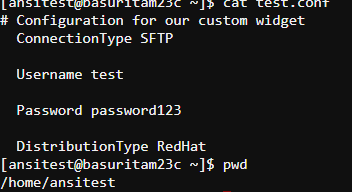
**userPassword: password123**

**connectionType: SFTP**

**tasks:**

**- name: Install the configuration file customized for the system**

**template: src=test.conf.j2 dest=/home/ansitest/test.conf owner=ansitest group=ansitest mode=750**

**Source: j2 template file at local  
dest: /home/ansitest/test.conf at target host  
we have also mentioned owner, group of the file and permissions  
  
execute at local:**$ ansible-playbook temp\_jinja2\_pb.yml  
  
**O/p:  
  
at target host server:**  
  
****

**18) SETUP Module: Lets gather some facts**

**--- # multiple facts**

**- hosts: apphosts**

**connection: ssh**

**user: ansitest**

**sudo: yes**

**gather\_facts: yes**

**tasks:**

**- name: gather some facts**

**setup: filter="{{ item }}"**

**loop:**

**- ansible\_architecture**

**- ansible\_distribution**

**- ansible\_domain**

**register: ans\_facts**

**- debug: var=ans\_facts**

**19) PAUSE MODULE:  
19.1)**

**tasks:**

**- name: Install httpd**

**yum: name=httpd state=present**

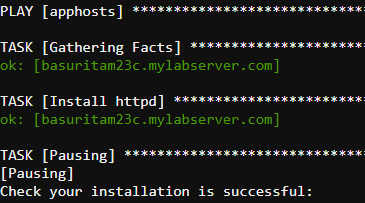
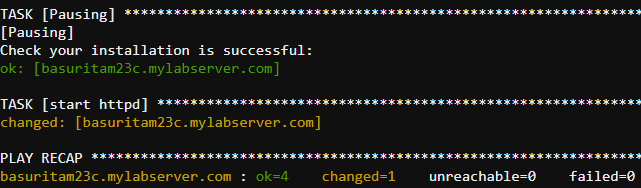
**- name: Pausing**

**pause:**

**prompt: Check your installation is successful**

**- name: start httpd**

**command: systemctl start httpd**

**Will pause here:  
  
  
it will wait for a Enter   
  
**

**19.2) If we wanted to pause for specific 5 minutes.  
  
- name: Pausing**

**pause:**

**minutes: 5**

**20) FILE module  
  
20.1) gather information about a file , say /etc/fstab**$ ansible webhosts -m file -a 'path=/etc/fstab'  
  
O/p:  
**basuritam22c.mylabserver.com | SUCCESS => {**

**"changed": false,**

**"gid": 0,**

**"group": "root",**

**"mode": "0644",**

**"owner": "root",**

**"path": "/etc/fstab",**

**"size": 342,**

**"state": "file",**

**"uid": 0**

**}**

**20.2) create a directory as /tmp/etc, owned by ansitest, group same  
with permission 700  
  
$ ansible webhosts -m file -a 'path=/tmp/etc state=directory owner=ansitest group=ansitest mode=700'**

**O/p:  
basuritam22c.mylabserver.com | CHANGED => {**

**"changed": true,**

**"gid": 1005,**

**"group": "ansitest",**

**"mode": "0700",**

**"owner": "ansitest",**

**"path": "/tmp/etc",**

**"size": 6,**

**"state": "directory",**

**"uid": 1004**

**}**

****

**20.3)copy the /etc/fstab file to /tmp/etc/fstab**$ ansible webhosts -m copy -a 'src=/etc/fstab dest=/tmp/etc/fstab'  
  
**20.4) Delete the /tmp/etc directory**$ ansible webhosts -m file -a 'path=/tmp/etc state=absent'

**basuritam22c.mylabserver.com | CHANGED => {**

**"changed": true,**

**"path": "/tmp/etc",**

**"state": "absent"**

**}**

**2**