Prevent Connection timeout:

https://www.computerworld.com/article/2701512/how-to-prevent-ssh-from-timing-out.html

Play books

Notes:
Adhoc commands are capable of working only on one module and one set of arguments.
When we want to perform complex configuration management activities,

adhoc commands will be difficult to manage.

In such scenarios, we use play books.

Play book is combination of plays.

Each play is designed to do some activity on the managed nodes.

These plays are created to work on single host or a group of hosts or all the hosts.

The main advantage of play books is reusability.
Play books are created using yaml files.

```
$ mkdir  playbooks
$ cd playbooks
$ vim playbook1.yml
INSERT   mode

---
- name: Install git and clone a
remote repository
  hosts: all
  tasks:
    - name: Install git
      apt:
       name: git
       state: present
       update_cache: yes
    - name: clone remote git
repository
      git:
        repo:
https://github.com/sunilkumark11/git
-9am-batch.git
        dest: /home/ubuntu/newgit

...
```

To check the syntax:

```
$ ansible-playbook  playbook1.yml
--syntax-check
```

(Do not use tab when creating yml file)

To run the playbook

```
$ ansible-playbook  playbook1.yml
-b
```

+++++++++++++++++++++++++++++

2nd example on playbook

Create user on all managed nodes and I want to copy passwd file.

```
$ vim playbook2.yml

---
- name: Create user and copy passwd
file
  hosts: all
  tasks:
          - name: User creation
            user:
             name: kiran
             password: sunilsunil
             uid: 6779
             home: /home/kiran
          - name: Copy password into
users home dir
            copy:
             src: /etc/passwd
             dest: /home/kiran
...
```

Save and quit
$

Check the syntax:

```
$ ansible-playbook  playbook2.yml
--syntax-check
```

To run

```
$ ansible-playbook  playbook2.yml
-b
```

TO check user is created in managed nodes:

```
$ ssh  172.31.2.173
$ vim /etc/passwd
```

To check if passwd file is copied to /home/kiran

```
$  cd /home/kiran
$ ls
$ exit
```

Ex 3: Playbook to configure tomcat8 (earlier example)

1st uninstall tomcat

```
$ ansible  all  -m  apt -a
'name=tomcat8 state=absent
purge=yes'  -b
```

```
$ vim playbook3.yml
```

```
---
- name: Configure  tomcat8
  hosts: all
  tasks:
   - name: Install tomcat8
     apt:
      name: tomcat8
      state: present
   - name: copy tomcat-users.xml
file
     copy:
      src:  /home/ubuntu/newfile1
      dest: /etc/tomcat8
   - name: change port of tomcat
from 8080 to 9090
     replace:
      regexp: 8080
      replace: 9090
      path: /etc/tomcat8/server.xml
   - name: restart tomcat8
     service:
      name: tomcat8
      state: restarted
   - name: check url response of
server 1
     uri:
      url: http://172.31.34.91:9090
   - name:   check url response of
```

```
server 2
     uri:
      url: http://172.31.33.68:9090
...
```

```
$ ansible-playbook  playbook3.yml
--syntax-check

$ ansible-playbook  playbook3.yml
-b
```

++++++++++++++++++++++++

Requirment:
Install apache2 in all managed nodes, Place our own content in default homepage

```
$ cd playbooks
$ vim playbook4.yml

---
```

```
- name: configuring apache2
  hosts: all
  tasks:
   - name: Install apache2
     apt:
      name: apache2
      state: present
```

Save and quit

```
$ ansible-playbook  playbook4.yml
-b
```

To check apache2 is installed

```
$ ssh 172.31.12.239
```

(Homepage of apache2 is present in /var/www/html)

```
$ cd  /var/www/html
$ ls
```

we get index.html (this html file

is default homepage of apache)
Editing the index.html page
This is possible using copy module.

```
$ exit
$ vim playbook4.yml
```

```
- name: configuring apache2
  hosts: all
  tasks:
   - name: Install apache2
     apt:
      name: apache2
      state: present
   - name: Edit index.html file
     copy:
      content: "Welcome to
Playbooks\n"
      dest: /var/www/html/index.html
```

save and quit

```
$ ansible-playbook  playbook4.yml
-b
```

++++++++++++++++++++++
How to open url in terminal?
by using elinks
Ex:

```
$  elinks http://google.com
```

We get error (elinks not found)

Let's install elinks

```
$ sudo apt-get install -y elinks
```

Now run the command

```
$  elinks http://google.com
```

Now we want to look at index.html
file in managed nodes

```
$ elinks http://15.207.111.187
```

After editing the index.html file, i
need to restart the service and
check the url response

```
$ vim playbook4.yml

---
- name: configuring apache2
  hosts: all
  tasks:
   - name: Install apache2
     apt:
      name: apache2
      state: present
   - name: Edit index.html file
     copy:
      content: "Welcome to
playbooks\n"
      dest: /var/www/html/index.html
   - name: Restart apache2
     service:
      name: apache2
      state: restarted
   - name: check url response of
server1
     uri:
      url: http://172.31.7.134
      status: 200
   - name: check url response of
server2
     uri:
      url: http://172.31.3.46
      status: 200
   - name: check url response of
server3
     uri:
      url: http://172.31.2.140
      status: 200
...
```

```
 ansible-playbook  playbook4.yml  -b
```

Notes:
Ex: Ansible playbook for configure apache2

++

Creating reusable playbooks using variables

3 Types of variables

1) Global scope variables   (highest priority) - we pass values from command prompt
2) Host scope variables
3) play scope variables   (least priority)

Ex of Global scope variables

```
$ vim playbook5.yml
```

```
---
- name: Install software packages
  hosts: all
  tasks:
   - name: Install/uninstall/update etc
     apt:
      name: tree
      state: present
      update_cache: yes
```

...

If we run the above play book 10 times, what happens? tree package will install 10 times.
The above play book is not reusable.

we make small changes to the above code

```
$ vim playbook5.yml
```

```
---
- name: Install software packages
  hosts: all
  tasks:
   - name: Install/uninstall/update
etc
     apt:
      name: "{{a}}"
      state: "{{b}}"
      update_cache: "{{c}}"
```

...

To run the playbook by passing values to the variables

```
$ ansible-playbook  playbook5.yml
--extra-vars "a=git b=absent c=no"
-b
```

(The above command will uninstall git from all nodes)

Run the same playbook with diffrent values

```
$ ansible-playbook  playbook5.yml
--extra-vars "a=tree b=present c=no"
 -b
```

+++++++++++++++++++

Before going to host scope variables,
lets discuss play scope variables

Playscope variables are definined within the playbook and they can effect only in one single play.

Ex:

```
$ vim playbook7.yml

---
- name: Using play scope variable
  hosts: all
  vars:
   - a: tomcat8
   - b: present
   - c: no
  tasks:
   - name: Install tomcat8
     apt:
      name: "{{a}}"
      state: "{{b}}"
      update_cache: "{{c}}"
...

$ ansible-playbook  playbook7.yml
-b
( It will install tomcat8 )
```

We can run by using extra vars from command line

```
$ ansible-playbook  playbook7.yml
--extra-vars "a=tree b=present c=no"
 -b

$ ansible-playbook  playbook7.yml
--extra-vars "a=tree b=absent c=no"
-b
```

The above command will install tree because global scope variables have higher priority

Notes:
Playscope variables
These variables are definied at
level of individual plays and they
can effect only one play.

Ex:

```
---
- name: Using play scope variable
  hosts: all
  vars:
   - a: tomcat8
   - b: present
   - c: no
  tasks:
   - name: Install tomcat8
     apt:
      name: "{{a}}"
      state: "{{b}}"
      update_cache: "{{c}}"
...
```

Note: The above playbook works like a template, who's default behaviour is to install tomcat8
But, we can by pass that behaviour and make it work in some other software by passing the variables as extra vars

```
$ ansible-playbook  playbook7.yml
-b  --extra-vars "a=tree b=present
c=no"  -b
```

The above command will install tree because global scope variables have higher priority

Notes:
Playscope variables

These variables are definied at level of individual plays and they can effect only one play.

Ex:

```
---
- name: Using play scope variable
  hosts: all
  vars:
   - a: tomcat8
   - b: present
   - c: no
  tasks:
   - name: Install tomcat8
     apt:
      name: "{{a}}"
      state: "{{b}}"
      update_cache: "{{c}}"
...
```

Note: The above playbook works like a template, who's default behaviour is to install tomcat8

But, we can by pass that behaviour and make it work in some other software by passing the variables as extra vars

++++++++++++++++++++

++++++++++++++++++++++++++++++++++++
++
Today we will discuss about host scope variables
Lets create one more managed node.
So, we will have 1 controller 4 nodes.
In step 6 -- Add rule -- All Traffic -- Anywhere

Establish password less ssh connection

```
$ sudo passwd ubuntu
```

(lets give the password as ubuntu only)

```
$ sudo vim /etc/ssh/sshd_config

change
PasswordAuthentication yes
Save and QUIT

$ sudo service ssh restart
$ exit

++++++++++++++++
Now,  Connect to controller
Now , We need to generate ssh
connections
$ ssh-keygen

Now copy the key to managed nodes

$ ssh-copy-id ubuntu@172.31.44.229
( private Ip of server4 )

+++++++++++
Now, we need to add the information
of managed nodes in the inventory
file.
```

Location of inventory file
/etc/ansible

```
$ cd /etc/ansible
$ ls
$ sudo vim hosts
```
insert the private ip addresss of
4th server
save and quit

```
$ ansible all  -a  'ls  -la'    (
```
you will get the list of the files
in all managed nodes)

```
$ ansible all  -a  'free'
```
++++++++++++++++++
We can do grouping using
[groupname]

Ex:

To do grouping

```
$ sudo vim hosts
```

```
[webserver]
172.31.11.96
172.31.6.207
[appserver]
172.31.12.138
[dbserver]
172.31.31.161
```

+++++++++++++++++++

```
$ ansible appserver  -a 'free'    (
It runs on one machine
172.31.12.138)
```

```
$ ansible webserver  -a 'free'   (
It runs on two machines )
```

```
$ ansible all  -a 'free'
```

+++++++++++++++++++++++

We can perform grouping on groups