1. Ansible Playbook to Install Tomcat Server

```
---
- name: Download Tomcat8 from tomcat.apache.org
  hosts: testserver
  vars:
    download_url:
https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.83/bin/apache-tomcat-8.
5.83.tar.gz
  tasks:
   - name: Download Open JDK
     become: yes
     apt:
      name: openjdk-8-jre-headless
      update_cache: yes
      state: present

   - name: validate if Java is availble
     shell:
      java -version

   - name: Create the group
     become: yes
     group:
      name: tomcat
      state: present

   - name: Create the user
     become: yes
     user:
        name: tomcat
        state: present

   - name: Create a Directory /opt/tomcat8
     become: yes
     file:
       path: /opt/tomcat8
       state: directory
       mode: 0755
       owner: tomcat
       group: tomcat

   - name: Download Tomcat using unarchive
     become: yes
     unarchive:
```

```
        src: "{{download_url}}"
        dest: /opt/tomcat8
        mode: 0755
        remote_src: yes
        group: tomcat
        owner: tomcat

    - name: Move files to the /opt/tomcat8 directory
      become: yes
      become_user: tomcat
      shell: "mv /opt/tomcat8/apache*/* /opt/tomcat8"

    - name: Creating a service file
      become: yes
      copy:
       content: |-
         [Unit]
         Description=Tomcat Service
         Requires=network.target
         After=network.target

         [Service]
         Type=forking
         User=tomcat
         Environment="CATALINA_PID=/opt/tomcat8/logs/tomcat.pid"
         Environment="CATALINA_BASE=/opt/tomcat8"
         Environment="CATALINA_HOME=/opt/tomcat8"
         Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server
-XX:+UseParallelGC"

         ExecStart=/opt/tomcat8/bin/startup.sh
         ExecStop=/opt/tomcat8/bin/shutdown.sh
         Restart=on-abnormal

         [Install]
         WantedBy=multi-user.target
       dest: /etc/systemd/system/tomcat.service

    - name: Reload the SystemD to re-read configurations
      become: yes
      systemd:
         daemon-reload: yes

    - name: Enable the tomcat service and start
      become: yes
```

```
     systemd:
        name: tomcat
        enabled: yes
        state: started

   - name: Connect to Tomcat server on port 8080 and check status 200
- Try 5 times
     tags: test
     uri:
       url: http://localhost:8080
     register: result
     until: "result.status == 200"
     retries: 5
     delay: 10
```

Where can this playbook be used?

This ansible-playbook is designed specifically for `ubuntu` or `debian` architecture Linux Systems.

As we are using `apt` package manager command to install java. You can replace it with `yum` if you are using this playbook for other linux distributions.

Let me know if you need any help or face any issues.

Run Using : `ansible-playbook install-tomcat.yaml`

---

2. How to send Slack Notifications using Ansible ?

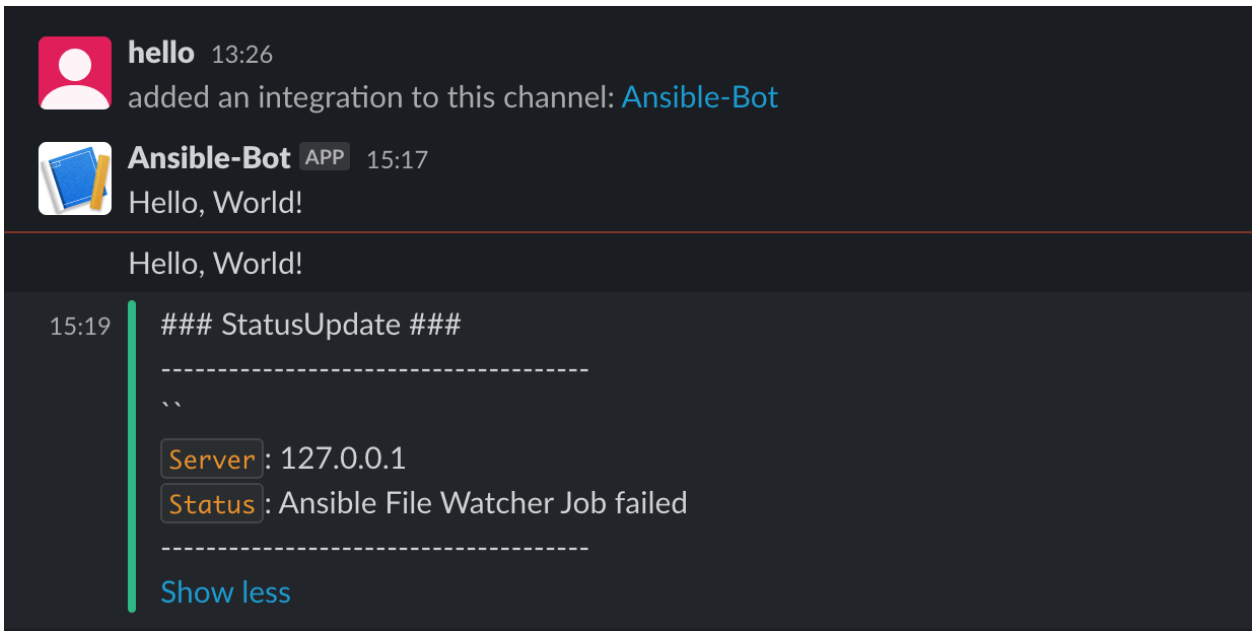
```
---
- hosts: localhost
  tasks:
  - name: Wait for the file to be available
    register: fileexists
    file:
      path: /tmp/myprocess.pid
      state: file
    until: fileexists is not failed
    retries: 5
    delay: 10
    ignore_errors: true

  - name: notify Slack that the job is failing
    tags: slack
    community.general.slack:
      token: T0266NX8KPF/B03FHCHAZJM/fk4OApn8KMGilhxxVtQ
      msg: |
          ### StatusUpdate ###
          - -----------------------------------
          ``
          `Server`: {{ansible_host}}
          `Status`: Ansible File Watcher Job failed
          - -----------------------------------
      channel: '#ansible-notifications'
      color: good
      username: 'Ansible on {{ inventory_hostname }}'
      link_names: 0
      parse: 'none'
    when: fileexists is failed
    ignore_errors: true

  - name: notify Slack that the job is Successful
    tags: slack
    community.general.slack:
      token: T0266NX8KPF/B03FHCHAZJM/fk4OApn8KMGilhxxVtQ
      msg: |
          ### StatusUpdate ###
          - -----------------------------------
          ``
```

```
          `Server`: {{ansible_host}}
          `Status`: Ansible File Watcher Job Successful.
          - ------------------------------------
      channel: '#ansible-notifications'
      color: good
      username: 'Ansible on {{ inventory_hostname }}'
      link_names: 0
      parse: 'none'
    when: fileexists is not failed
```

Output:



---

3. Ansible inventory_hostnames and ansible_hostnames

we are going to see two built-in variables of ansible mostly used in Ansible playbooks and they are `inventory_hostname` and `ansible_hostname` while both these variables are to give you the hostname of the machine. they differ in a way, where it comes from.

ansible_hostname	inventory_hostname
Ansible_hostname takes the hostname from the facts collected during the gather_facts this would mostly match to the `uname -n` or `hostname` command that you run on the remote machine	inventory_hostname takes the hostname from the inventory configuration or the hosts file. this may not match the hostname configuration of the remote system. this could just be a local identity mentioned on the control machine
If the `Gather_facts` is set to NO the `ansible_facts` variable would not be available to use in your playbook	inventory_hostname would always be available to use in your playbook.
Since the data taken from the configuration file. this may not match the `uname -n` or `hostname` command output of the remote machine	Since this is taken from the facts collected at runtime this would have the same hostname defined in system configuration like `/etc/hostname` and match the output of `uname -n` and `hostname`
As this is based on the Gather_facts step. **ansible_hostname not available in ad_hoc command**	Available for both playbook and ad hoc command

A Simple playbook to demonstrate the inventory_hostname and ansible_hostname

```
---
 - name: "Playbook to test the inventory_hostname
and ansible_hostname"
   hosts: testserver
   tasks:

    - name: What is my inventory_hostname
      debug: var={{inventory_hostname}}

    - name: What is my ansible_hostname
      debug: var={{ansible_hostname}}
```

Ansible Get IP Address of Current Host or Target

Method 1 : Get IP Used by Ansible master to connect

n this method, we are going to use the IP address used by Ansible master to connect to the Remote Host.

Sometimes, we would use the public IP to connect to the remote host in such cases we would want to use this method.

This is the method I have been using for a while and It never failed me. The /etc/hosts update post tagged above has also implemented with this method.

So in this method. to get the remote IP address with ansible we are relying on the SSH connectivity between the Ansible Master and remote host(target)

If you know already, ansible Gather_facts collects all the information about the remote hosts and it provides a various lot of information about the remote host including the IP address being used for SSH connection.

The following playbook would show how to get the IP address of the remote target or host using the SSH Connection between the ansible master and the host.

```
---
- hosts: all
  gather_facts: yes
  tasks:
    - debug:
var=hostvars[inventory_hostname]['ansible_env'].SSH_CONNECTION.split(' ')[2]
```

How to Use Find in Ansible Examples

```yaml
---
- name: Ansible Find Example
  hosts: testserver
  tasks:
   - name : Find files older than 30 days
     find:
       paths: /var/log
       patterns: '*.log'
       age: 30d
       age_stamp: mtime
     register: output

   - debug: var=item.path
     with_items: "{{ output.files }}"
```

Find All directories excluding few, as a list

```yaml
---
- name: Ansible Find Example
  hosts: testserver
  vars:
    Files: []
  tasks:
   - name : Find files bigger than 100mb in size
     become: true
     find:
       paths: /var/log
       file_type: directory
       recurse: no
       excludes: 'nginx,mysql'
     register: output

   - name: Adding Files to the LIST
     no_log: true
     set_fact:
       Files: "{{ Files + [item.path]}}"
     with_items: "{{ output.files }}"
```

```
   - debug: var=Files
```

Find Files with a Regex Pattern in Ansible find

```
---
- name: Ansible Find Example
  hosts: testserver
  vars:
    Files: []
  tasks:
   - name : Find files bigger than 100mb in size
     become: true
     find:
       paths: /var/log
       file_type: file
       patterns:  '^[a-z]*_[0-9]{8}\.log$'
       size: 100m
       use_regex: yes
     register: output

   - name: Adding Files to the LIST
     no_log: true
     set_fact:
       Files: "{{ Files + [item.path]}}"
     with_items: "{{ output.files }}"

   - debug: var=Files
```

Ansible Find and Delete the files

```
---
- name: Ansible Find Example
  hosts: testserver
  tasks:
   - name : Find files older than 30 days
     become: yes
     find:
       paths: /var/log
       patterns: '*.log'
```

```
      age: 30d
      age_stamp: mtime
    register: output

  - name: Delete the files matching
    become: yes
    file:
      path: "{{item.path}}"
      state: absent
    with_items: "{{ output.files }}"
```

Ansible Command Module Examples

```
---
 - name: Check the remote host uptime
    hosts: testservers
    tasks:
      - name: Execute the Uptime command
over Command module
        register: uptimeoutput
        command: "uptime"

      - debug:
          var: uptimeoutput.stdout_lines
```

> Example2: Get the Hostname and Version of remote servers with UNAME

```
---
```

```
  - name: Check the remote host Hostname,
Version, Distribution with UNAME
    hosts: testservers
    tasks:
      - name: Execute the UNAME command
        register: unameout
        command: "uname -a"

      - debug:
          var: unameout.stdout_lines
```

Example3: Check the Disk Usage of Remote server

```
---
  - name: Check the disk usage of all the
file system in the remote servers
    hosts: testservers
    tasks:
      - name: Execute the df command
        register: dfout
        command: "df -h"

      - debug:
          var: dfout.stdout_lines
```

Example4: Restart Apache Server using Ansible Command Module

```
---
  - name: restart apache web server
    hosts: testservers
    tasks:
      - name: restartapache
        register: httpdresout
        become: yes
        command: "httpd -k restart"
        when: ansible_hostname ==
"mwiweb02"


      - debug:
          var: httpdresout.stdout_lines
```

Example5: Execute a command when a file exists or not exists

```
---
 - name: "Validate if a file is present or
not present using Ansible Command module"
   hosts: testservers
   tasks:
```

```
    - name: "Create a file if it does not exist"
      command: "touch /tmp/latestfile"
      args:
         creates: "/tmp/latestfile"
      register: createif

    - name: "Display the file to make sure its created"
      command: "ls -lrt /tmp/latestfile"
      register: displayif
      when: createif is changed

    - debug: var=displayif.stdout

    - name: "Remove the file if it exist"
      command: "rm -rf /tmp/latestfile"
      args:
         removes: "/tmp/latestfile"
      register: removeif
```

Ansible Unarchive Module Examples

Playbook without Unarchive

```
---
- name: Playbook to copy file and uncompress
  hosts: appservers
  vars:
    - userid : "weblogic"
    - oracle_home: "/opt/oracle"
    - jdk_instl_file: "server-jre-8u191-linux-x64.tar.gz"

  tasks:
  - name : Copy the Java JDK files
    become: yes
    become_user: "{{ userid }}"
    tags: app,cpbinaries
    copy:
      src: "{{ item }}"
      dest: "{{ oracle_home }}"
      mode: 0755
    with_items:
      - "{{ jdk_instl_file }}"

  - name: Install java
    become: yes
    become_user: "{{ userid }}"
    tags: javainstall
    shell: "tar xvfz {{ oracle_home }}/{{ jdk_instl_file }}"
    args:
      chdir: "{{ oracle_home }}"
    register: javainstall
```

Playbook with Unarchive :

```
---
- name: Playbook to copy file and uncompress
  hosts: appservers
  vars:
    - userid : "weblogic"
    - oracle_home: "/opt/oracle"
```

```
    - jdk_instl_file: "server-jre-8u191-linux-x64.tar.gz"

  tasks:
  - name : Copy and Install Java
    become: yes
    become_user: "{{ userid }}"
    tags: javainstall
    unarchive:
      src: "{{ item }}"
      dest: "{{ oracle_home }}"
      mode: 0755
    with_items:
      - "{{ jdk_instl_file }}"
```

Download a Zip file from remote URL and decompress using unarchive

```
---
- name: Playbook to download and install tomcat8
  hosts: appservers

  tasks:
  - name: install Java
    become: yes
    yum:
      name: java-1.8.0-openjdk-devel
      state: present

  - name: crate a directory
    become: yes
    file:
      path: "/opt/tomcat8"
      state: directory
      mode: 0755

  - name : Download and install tomcat
    become: yes
    tags: installtc
    unarchive:
```

```
      src:
"http://apachemirror.wuchna.com/tomcat/tomcat-8/v8.5.49/bin/apache-tomcat-8.5.49.tar.gz"
      dest: "/opt/tomcat8/"
      mode: 0755
      remote_src: yes
    register: "tcinstall"

  - name: Start the tomcat instance
    become: yes
    shell:
      "./startup.sh"
    args:
      chdir: "/opt/tomcat8/apache-tomcat-8.5.49/bin"
```

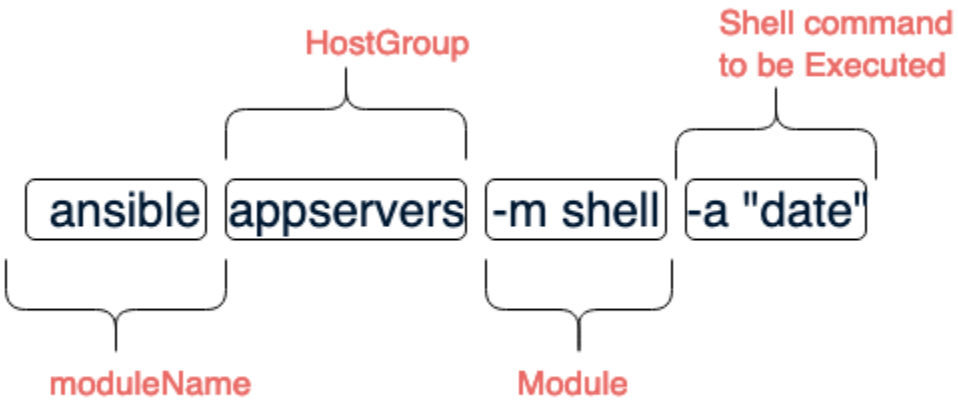
What is this remote_src in ansible unarchive module

By this time you would have understood what is this remote_src is but if you are still having a question. This is what it is. `remote_src` tells the unarchive module to look for the source file on the remote server unless otherwise unarchive would check for the file on the control machine and try to copy it.

Ansible unarchive is by default designed to copy the file mentioned in the `src` from the control machine (where Ansible is installed) to the remote server. Sometimes we do not want to copy the files or our files would somehow be already present on the remote target server. in that case, you can use this `remote_src` option.

Ansible Shell Module Examples

Here is the quick Syntax of Ansible Shell module in ADHOC manner.



Example 1: Execute a Single Command with Ansible Shell

```
---
 - name: Shell Examples
   hosts: testservers
   tasks:

   - name: Check Date with Shell command
     shell:
       "date"
     register: datecmd
     tags: datecmd

   - debug: msg="{{datecmd.stdout}}"
```

Example 2: Execute a Command with Pipe and Redirection

```
---
 - name: Shell Examples
   hosts: testservers
   tasks:

   - name: Dir list and write to file
     shell:
        " ls -lrt /apps|awk '{print $9}'|sed '/^$/d' > /tmp/dirlist.txt "
     register: lsout
     tags: lsout

   - name: Display the file
     shell: cat /tmp/dirlist.txt
     register: displaylist

   - debug: msg="{{displaylist.stdout_lines}}"
```

Example 3: Execute a Shell Script with Shell command

```
---
 - name: Shell Examples
   hosts: testservers
   tasks:

   - name: Start tomcat
     become: yes
     become_user: tomcat
     async: 10
     poll: 0
     shell:
        "./startup.sh"
     args:
       chdir: "/apps/tomcat/tomcat8/bin"
     register: datecmd
     tags: datecmd

   - name: Validate if tomcat is UP
     tags: tomvalidate
     wait_for:
       host: "localhost"
       port: 8080
```

```
delay: 10
timeout: 30
state: started
msg: "Tomcat server is not running"
```

How to use Ansible with Windows Host - Ansible Windows Example

ow to Setup Windows machine for Ansible to be able to connect or remote login just like SSH in Linux.

While there is a way to use SSH in windows which can be further leveraged by ansible for windows connectivity and automation.

There is a better way and more stable way to do it with Windows Remote Manager (WinRM)

So we are going to see how to use **WinRM** and connect to remote windows machine from Ansible control machine.

How to Setup WinRM in Windows Machine to Prepare for Ansible

The First step for us to be able to connect to the windows machine is to install this WinRM properly on our Windows machine.

Thanks to Ansible team. they have created a PowerShell script that does the required configuration on the windows machine for us.

Do not worry about downloading the Powershell script file. Just run the following powershell command in your PowerShell terminal

This downloads the script automatically and runs it in your terminal.

```
iex(iwr https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1).Content
```

If the installation is done right. you can see that your `WinRM` is UP and running and would be listening in port `5986`

here is a quick command to for you to check if winrm listens on the port 5986

```
netstat -anp|findstr 5986
```

Thats all.

One more thing is pending. If you are on a cloud,

Consider opening this port to ansible control machine. So the Ansible can connect this machine from control machine. (It is same like Opening up port 22 for linux to allow SSH)

If you are using Ansible with Python 2 use PIP to install this package

```
pip install pywinrm
```

If you are using Ansible with Python3 use PIP3 to install pywinrm

```
pip3 install pywinrm
```

once the `pywinrm` package is installed we are all good and we can go and do a quick health check with ping.

But I would recommend you can use `telnet` or `nc` command (whichever available) to make sure that the network connectivity is there to the remote machine

```
nc -w 3 -v <remote windows server ip/hostname> 5986
(or)
telnet <remote windows server ip/hostname>:5986
```

This would give you an additional confidence that your connection is OK.

Create or Update ansible hosts inventory file

Before you can connect to the remote machine with Ansible.

you need to let Ansible know about this machine, as usual you need to add this machine to any hostgroup. In my case the host group name is `win`

```
[win]

192.9.12.122
```

You can keep the IP or the hostname which ever is reachable from your ansible control machine

Additionally, since this is windows, we need to provide some more variables at the hostgroup level.

```
[win:vars]

ansible_connection=winrm

ansible_user=administrator

ansible_password=r$eBQNgc5U&A2at8kDwpWo.KzLT5NvHd

ansible_winrm_server_cert_validation=ignore
```

- `ansible_connection=winrm` to define the connection is not SSH should use winrm
- `ansible_user` what ever the username you have created in the windows machine
- `ansible_password` password for that user (the same one you use for RDP)
- `ansible_winrm_server_cert_validation` this is fine in DEV/TEST environment to tell ansible to ignore hostkey/server cert validation.

The complete inventory file is given below for your reference

```
[win]

192.9.12.122

[win:vars]
```

```
ansible_connection=winrm

ansible_user=administrator

ansible_password=r$eBQNgc5U&A2at8kDwpWo.KzLT5NvHd

ansible_winrm_server_cert_validation=ignore
```

I have saved this file in my custom directory where I would create my playbooks and named this as `ansible_hosts`

Ansible `ping` is to check the connection from control machine to remote linux machine.

Likewise, Ansible `win_ping` is to check the connectivity from Control machine to Windows.

It is like a `Hello world` of programming language we can say.

So we are going execute the following command

```
ansible win -m win_ping -i ansible_hosts
```

here the `win` is our host group name and with `-m` we are telling ansible to use `win_ping` module

Validate Other Ansible AD Hoc commands and Playbooks

Once the `win_ping` is green. you can execute some other modules and commands either as `ad_hoc` or as playbook to test it

here is a quick playbook you can use which executes a command on the remote server

```
---

- name: Windows Test Playbook

 hosts: win

 tasks:

   - name: Remote Execute the mqsc files

     win_shell: |

       hostname

       Get-Date

     register: scriptoutput

   - name: Script output

     debug: var=scriptoutput.stdout
```

The same playbook can be executed as two ansible ad hoc commands

```
ansible win -m win_shell -a "hostname" -i ansible_hosts
```

```
ansible win -m win_shell -a "Get-Date" -i ansible_hosts
```

```
-----------------------------------------------------
```

Ansible delegate_to Examples - Run task on specific host

Ansible Delegate_to module helps us to execute a specific task in our playbook to run in other host or machine.

This process of handing over the control of execution to another host (or) executing a task in another machine is called as `delegation` and the module `delegate_to` helps you to configure it properly and achieve the desired result.

While we have already written various playbooks in Devops Junction with delegate_to directive.

As ansible `delegate_to` is a directive, not an individual module, It integrates with other modules and it controls the task execution by deciding which host should run the task at runtime.

Refer to the following snapshot of the playbook, there are few tasks.

If you look at the Task3, you can see there is a `delegate_to` used to change the control of execution back to the ansible master from the remote host where the task was supposed to run.

this is a simple delegation where the control of execution transferred from remote host to the master.

You can write playbooks to change the control between the remote servers as well. which is already covered in our aforementioned articles. We will take a look in detail shortly.

```yaml
---
- name: "Ansible Delegate_to examples"
  hosts: testserver
  tasks:

  - name: "Task1: start a quick webserver with NC on remote server"
    shell: "while true; do { echo -e 'HTTP/1.1 200 OK\r\n'; echo Hello from WebServer ;  } | nc -l 8081; done &"
    #Fire and Forget the job - let it run in background
    async: 1
    poll: 0

  - name: "Task2: Enabling the port in firewall"
    become: true
    firewalld:
      port: 8081/tcp
      permanent: true
      state: enabled
      immediate: yes

  - name: "Task3: Check the status of the remote server from localhost [Ansible master]"
    uri:
        url: http://{{inventory_hostname}}:8081
        method: GET
        timeout: 30
        status_code: 200
        return_content: yes
    delegate_to: localhost
    register: webresult

  - name: "Task4: Printing the website output from the Task3"
    debug: var=webresult
```

Hostgroup Name

Send Task

delegate

Ansible Master
Localhost

Test Server

delegated to localhost

DEVOPSJUNCTION.com

The Playbook to execute tasks in another host with delegate_to

```
---
- name: "Ansible Delegate_to examples"
  hosts: testserver
  tasks:

  - name: "Task1: start a quick webserver with NC on remote
server"
    shell: "while true; do { echo -e 'HTTP/1.1 200 OK\r\n'; echo
Hello from WebServer ;  } | nc -l 8081 > /tmp/serverout.log ;
done &"
    #Fire and Forget the job - let it run in background
    async: 1
    poll: 0

  - name: "Task2: Enabling the port in firewall"
    become: true
    firewalld:
      port: 8081/tcp
      permanent: true
      state: enabled
      immediate: yes

  - name: "Task3: Check the status of the remote server from
localhost [Ansible master]"
    uri:
        url: http://{{inventory_hostname}}:8081
        method: GET
        timeout: 30
        status_code: 200
        return_content: yes
    delegate_to: localhost
    register: webresult

  - name: "Task4: Printing the website output from the Task3"
    debug: var=webresult
```

How to use ansible with S3 - Ansible aws_s3

Ansible S3 List Objects in a Bucket

```
---
- name: AWS S3 Bucket List - Ansible
  hosts: localhost
  tasks:
  - name: List keys or Objects
    amazon.aws.aws_s3:
      profile: personal
      bucket: devopsjunction
      mode: list
    register: listresult

  - debug: msg={{listresult.s3_keys}}
```

Ansible S3 List Objects using the prefix

```
---
- name: AWS S3 Bucket List - Ansible
  hosts: localhost
```

```
  tasks:
  - name: List keys/Objects
    amazon.aws.aws_s3:
      profile: personal
      bucket: devopsjunction
      mode: list
      prefix: "2021/12"
      max_keys: 400
    register: listresult

  - debug: msg={{listresult.s3_keys}}
```

Ansible S3 Upload / PUT example

```
---
- name: AWS S3 Bucket Upload - Ansible
  hosts: localhost
  tasks:
  - name: Upload/PUT file to S3 bucket
    amazon.aws.aws_s3:
      profile: personal
      bucket: devopsjunction
      mode: put
```

```
      object: "2021/12/Screenshot
2021-12-26 at 8.31.33 PM.png"
      src:
"/Users/saravananthangaraj/Desktop/Screensh
ot 2021-12-26 at 8.31.33 PM.png"
    register: putresult

  - debug: msg="{{ putresult.msg }} and the
S3 Object URL is {{putresult.url}}"
    when: putresult.changed
```

Ansible Pre Tasks and Post Tasks Example

Ansible pretask is a conditional execution block that runs before running the play. It can be a task with some prerequisites check (or) validation

Some examples we can think where ansible pre-task would be helpful

- To install dependency packages before running the actual application `npm install` or `pip install -r requirements.txt`
- To validate if the environment meets sufficient criteria like memory, OS version etc before installing a software
- configuring SSH key prior so that you can log in
- Server provisioning steps like creating user, group etc before the installation or setup begin

Just like ansible pre_tasks but this is executed after the actual play or task is completed.

this is mostly useful for post validation or assertion to make sure things are in the right shape or the result is matching our expectations.

Some examples we can think of are listed below

- To validate if things were installed properly and the playbook executed fine.
- To send emails or Slack notifications after successful completion of a playbook

- Running some other task or for accessing some API or external service upon playbook completion like webhook
- Starting the application after the installation and configuration is completed

We can keep this list going with more and more use cases for ansible post_tasks

Simply put, ansible post_tasks is to conditionally execute a block or play upon the successful completion of the playbook. it can be used for post validation, post-execution automation.

Let us see how to use ansible `pre_tasks` and `post_tasks` in the ansible-playbook practically.

In this playbook, we are going to perform the following tasks

- Install necessary commands and tools using `apt` module - `pre_tasks`
- Validate the nodejs installation is successful and print version using debug and assert

- Create a directory to download the nodejs application
- Download the nodejs codebase from GIT repo using Tokenized URL
- Tokens are saved and retrieved from the Secrets file named `gitsecrets.yml`
- Do `npm install`once the code is downloaded
- Start the nodejs application
- Validate if the port is open and Node js is accepting requests
- Send Slack notification using ansible slack module - `post_tasks`

```
---
  - name: Install and Launch the Simple NodeJS Application
    hosts: testserver
    vars_files:
        - gitsecrets.yml
    vars:
        - destdir: /apps/sampleapp

    pre_tasks:
```

```
      - name : install dependencies before
starting
        become: yes
        register: aptinstall
        apt:
           name:
            - nodejs
            - npm
            - git
           state: latest
           update_cache: yes

      - name : validate the nodejs
installation
        debug: msg="Installation of node is
Successfull"
        when: aptinstall is changed

    tasks:
       - name: Version of Node and NPM
         shell:
            "npm -v && nodejs -v"
         register: versioninfo

       - name: Validate if the installation
is intact
         assert:
```

```
            that: "versioninfo is changed"


        - name: Create Dest Directory if not
exists
          become: yes
          file:
           path: "{{ destdir }}"
           state: directory
           owner: vagrant
           group: vagrant
           mode: 0755

        - name: Download the NodeJS code
from the GitRepo
          become: yes
          git:
             repo:
'https://{{gittoken}}@github.com/AKSarav/Sa
mpleNodeApp.git'
             dest: "{{ destdir }}"

        - name: Change the ownership of the
directory
          become: yes
          file:
            path: "{{destdir}}"
```

```
            owner: "vagrant"
          register: chgrpout

        - name: Install Dependencies with
NPM install command
          shell:
             "npm install"
          args:
             chdir: "{{ destdir }}"
          register: npminstlout

        - name: Debug npm install command
          debug:
msg='{{npminstlout.stdout_lines}}'


        - name: Start the App
          async: 10
          poll: 0
          shell:
             "(node index.js > nodesrv.log
2>&1 &)"
          args:
            chdir: "{{ destdir }}"
          register: appstart

        - name: Validating the port is open
```

```
          tags: nodevalidate
          wait_for:
            host: "localhost"
            port: 5000
            delay: 10
            timeout: 30
            state: started
            msg: "NodeJS server is not
running"

    post_tasks:
        - name: notify Slack that the
servers have been updated
          tags: slack
          community.general.slack:
            token:
T026******PF/B02U*****N/WOa7r**********Ao0j
nWn
            msg: |
                ### StatusUpdate ###
                -
-----------------------------------
                ``
                `Server`: {{ansible_host}}
                `Status`: NodeJS Sample
Application installed successfully
```

```
                  -
--------------------------------------
            channel: '#ansible'
            color: good
            username: 'Ansible on {{
inventory_hostname }}'
            link_names: 0
            parse: 'none'
          delegate_to: localhost
```