Name: Shaniah Rose Hope M. Sumaoang	**Date Performed: September 7, 2023**
Course/Section: CPE 232-CPE31S5	**Date Submitted: September 12, 2023**
Instructor: Engr. Roman Richard	**Semester and SY: 1st Sem 2023-2024**

Activity 4: Running Elevated Ad hoc Commands

1. **Objectives:**

1.1 Use commands that makes changes to remote machines

1.2 Use playbook in automating ansible commands

2. **Discussion:**

Provide screenshots for each task.

Elevated Ad hoc commands

So far, we have not performed ansible commands that makes changes to the remote servers. We manage to gather facts and connect to the remote machines, but we still did not make changes on those machines. In this activity, we will learn to use commands that would install, update, and upgrade packages in the remote machines. We will also create a playbook that will be used for automations.

Playbooks record and execute **Ansible**'s configuration, deployment, and orchestration functions. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material. At a basic level, playbooks can be used to manage configurations of and deployments to remote machines. At a more advanced level, they can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way. You can check this documentation if you want to learn more about playbooks. Working with playbooks — Ansible Documentation

Task 1: Run elevated ad hoc commands

1. Locally, we use the command *sudo apt update* when we want to download package information from all configured resources. The sources often defined in /etc/apt/sources.list file and other files located in /etc/apt/sources.list.d/ directory. So, when you run update command, it downloads the package information from the Internet. It is useful to get info on an updated version of packages or their dependencies. We can only run an apt update command in a remote machine. Issue the following command:

ansible all -m apt -a update_cache=true

What is the result of the command? Is it successful? **The command is unsuccessful.**

```
srhmshan@localmachine:~$ ansible all -m apt -a update_cache=true
127.0.0.1 | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "msg": "Failed to lock apt for exclusive operation: Failed to
 lock directory /var/lib/apt/lists/: E:Could not open lock file /
var/lib/apt/lists/lock - open (13: Permission denied)"
}
```

Try editing the command and add something that would elevate the privilege. Issue the command *ansible all -m apt -a update_cache=true --become --ask-become-pass.* Enter the sudo password when prompted. You will notice now that the output of this command is a success. The *update_cache=true* is the same thing as running *sudo apt update*. The --become command elevate the privileges and the *--ask-become-pass* asks for the password. For now, even if we only have changed the packaged index, we were able to change something on the remote server.

You may notice after the second command was executed, the status is CHANGED compared to the first command, which is FAILED.

```
srhmshan@localmachine:~$ ansible all -m apt -a update_cache=true --become
become-pass
BECOME password:
127.0.0.1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694355124,
    "cache_updated": true,
    "changed": true
}
```

2. Let's try to install VIM, which is an almost compatible version of the UNIX editor Vi. To do this, we will just changed the module part in 1.1 instruction. Here is the command: *ansible all -m apt -a name=vim-nox --become --ask-become-pass.* The command would take some time after typing the password because the local machine instructed the remote servers to actually install the package.

```
srhmshan@localmachine:~$ ansible all -m apt -a name=vim-nox --become --ask-becom
e-pass
BECOME password:
127.0.0.1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694355124,
    "cache_updated": false,
    "changed": true,
    "stderr": "",
    "stderr_lines": [],
    "stdout": "Reading package lists...\nBuilding dependency tree...\nReading st
ate information...\nThe following additional packages will be installed:\n  font
s-lato javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n  ruby-n
et-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integration
 vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd ri ruby-dev bund
ler cscope vim-doc\nThe following NEW packages will be installed:\n  fonts-lato
javascript-common libjs-jquery liblua5.2-0 libruby3.0 rake ruby\n  ruby-net-teln
et ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0\n  rubygems-integration vim-no
        update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/vi (vi)
 in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/view (v
iew) in auto mode",
        "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/ex (ex)
 in auto mode",
        "Setting up ruby (1:3.0~exp1) ...",
        "Setting up ruby-rubygems (3.3.5-2) ...",
        "Processing triggers for man-db (2.10.2-1) ...",
        "Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...",
        "Processing triggers for libc-bin (2.35-0ubuntu3.1) ..."
    ]
}
```

2.1 Verify that you have installed the package in the remote servers. Issue the command *which vim* and the command *apt search vim-nox* respectively. Was the command successful? **The command was successful.**

```
srhmshan@localmachine:~$ which vim
/usr/bin/vim
```

```
srhmshan@localmachine:~$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installed
]
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/jammy-updates,jammy-security,now 2:8.2.3995-1ubuntu2.11 amd64 [installe
d,automatic]
  Vi IMproved - enhanced vi editor - compact version
```

2.2 Check the logs in the servers using the following commands: *cd /var/log*. After this, issue the command *ls,* go to the folder *apt* and open history.log. Describe what you see in the history.log.

```
srhmshan@localmachine:~$ cd /var/log
srhmshan@localmachine:/var/log$ ls
alternatives.log  dist-upgrade  fontconfig.log   private
apport.log        dmesg         gdm3             speech-dispatcher
apt               dmesg.0       gpu-manager.log  syslog
auth.log          dmesg.1.gz    hp               ubuntu-advantage.log
boot.log          dmesg.2.gz    installer        unattended-upgrades
boot.log.1        dmesg.3.gz    journal          wtmp
bootstrap.log     dmesg.4.gz    kern.log
btmp              dpkg.log      lastlog
cups              faillog       openvpn
```

```
srhmshan@localmachine:/var/log/apt$ cat history.log

Start-Date: 2023-08-07  22:53:16
Commandline: apt-get --yes -oDebug::pkgDepCache::AutoInstall=yes --force-yes upg
rade
Upgrade: dpkg:amd64 (1.21.1ubuntu2, 1.21.1ubuntu2.2), libxtables12:amd64 (1.8.7-
1ubuntu5, 1.8.7-1ubuntu5.1), networkd-dispatcher:amd64 (2.1-2, 2.1-2ubuntu0.22.0
4.2), libpam-runtime:amd64 (1.4.0-11ubuntu2, 1.4.0-11ubuntu2.3), python3.10:amd6
4 (3.10.4-3, 3.10.12-1~22.04.2), python3-gi:amd64 (3.42.0-3build1, 3.42.1-0ubunt
u1), libext2fs2:amd64 (1.46.5-2ubuntu1, 1.46.5-2ubuntu1.1), libgssapi-krb5-2:amd
64 (1.19.2-2, 1.19.2-2ubuntu0.2), apt:amd64 (2.4.5, 2.4.9), systemd-timesyncd:am
Start-Date: 2023-09-10  22:14:53
Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Opt
ions::=--force-confold install vim-nox
Requested-By: srhmshan (1000)
Install: fonts-lato:amd64 (2.0-2.1, automatic), liblua5.2-0:amd64 (5.2.4-2, auto
matic), ruby-net-telnet:amd64 (0.1.1-2, automatic), rubygems-integration:amd64 (
1.18, automatic), libruby3.0:amd64 (3.0.2-7ubuntu2.4, automatic), rake:amd64 (13
.0.6-2, automatic), vim-nox:amd64 (2:8.2.3995-1ubuntu2.11), ruby:amd64 (1:3.0~ex
p1, automatic), vim-runtime:amd64 (2:8.2.3995-1ubuntu2.11, automatic), ruby3.0:a
md64 (3.0.2-7ubuntu2.4, automatic), libjs-jquery:amd64 (3.6.0+dfsg+~3.5.13-1, au
tomatic), ruby-rubygems:amd64 (3.3.5-2, automatic), javascript-common:amd64 (11+
nmu1, automatic), ruby-xmlrpc:amd64 (0.3.2-1ubuntu0.1, automatic), ruby-webrick:
amd64 (1.7.0-3, automatic)
End-Date: 2023-09-10  22:15:01
srhmshan@localmachine:/var/log/apt$
```

The history.log file is consisted of records package management activities like package installations, updates etc. At the tail image, it is indicated that the vim-nox was installed at 2023-09-10 at 22:14:53. The activities are shown in a chronological order.

3. This time, we will install a package called snapd. Snap is pre-installed in Ubuntu system. However, our goal is to create a command that checks for the latest installation package.

   3.1 Issue the command: *ansible all -m apt -a name=snapd --become --ask-become-pass*

   Can you describe the result of this command? Is it a success? Did it change anything in the remote servers? **The operation was carried out successfully, and it did not affect any remote servers, as indicated by the "changed" status being false. This implies that no modifications were made to remote servers. The command's purpose was to either install or verify the presence of the "snapd" package, but it appears**

that this package was already installed or accessible, rendering any alterations unnecessary.

```
srhmshan@localmachine:~$ ansible all -m apt -a name=snapd --become --ask-become-
pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694355124,
    "cache_updated": false,
    "changed": false
}
```

3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapd state=latest" --become --ask-become-pass*

Describe the output of this command. Notice how we added the command *state=latest* and placed them in double quotations.

```
srhmshan@localmachine:~$ ansible all -m apt -a "name=snapd state=latest" --becom
e --ask-become-pass
BECOME password:
127.0.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1694358400,
    "cache_updated": false,
    "changed": false
}
```

It attempted to update the "snapd" package to its latest version, with the "--become" flag prompting for elevated privileges, and "--ask-become-pass" prompted for the password.

4. At this point, make sure to commit all changes to GitHub.

```
srhmshan@localmachine:~/CPE232_Sumaoang$ git add .
srhmshan@localmachine:~/CPE232_Sumaoang$ git commit -m "ansible"
[main c8c8b3b] ansible
 1 file changed, 1 insertion(+)
 create mode 100644 ansible.txt
srhmshan@localmachine:~/CPE232_Sumaoang$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:srhmshan/CPE232_Sumaoang.git
   8a4312f..c8c8b3b  main -> main
srhmshan@localmachine:~/CPE232_Sumaoang$ git log
commit c8c8b3b6b00b49fae0449a913eef8f252f43af99 (HEAD -> main, origin/main, orig
in/HEAD)
Author: Shaniah Sumaoang <shansumaoang@gmail.com>
Date:   Sun Sep 10 23:07:46 2023 +0800
```

```
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:srhmshan/CPE232_Sumaoang.git
   8a4312f..c8c8b3b  main -> main
srhmshan@localmachine:~/CPE232_Sumaoang$ git log
commit c8c8b3b6b00b49fae0449a913eef8f252f43af99 (HEAD -> main, origin/main, orig
in/HEAD)
Author: Shaniah Sumaoang <shansumaoang@gmail.com>
Date:   Sun Sep 10 23:07:46 2023 +0800

    ansible

commit 8a4312f02d98b6088ae0fc954bcd24a2df44c331
Author: Shaniah Sumaoang <shansumaoang@gmail.com>
Date:   Thu Aug 24 23:01:10 2023 +0800

    grrah

commit 1318b9c4f2f789d271b68020368fa230b55c01d6
Author: srhmshan <143104625+srhmshan@users.noreply.github.com>
Date:   Thu Aug 24 22:40:03 2023 +0800

    Initial commit
```

Task 2: Writing our First Playbook

1. With ad hoc commands, we can simplify the administration of remote servers. For example, we can install updates, packages, and applications, etc. However, the real strength of ansible comes from its playbooks. When we write a playbook, we can define the state that we want our servers to be in and the place or commands that ansible will carry out to bring to that state. You can use an editor to create a playbook. Before we proceed, make sure that you are in the directory of the repository that we use in the previous activities (*CPE232_yourname*). Issue the command *nano install_apache.yml*. This will create a playbook file called *install_apache.yml*. The .yml is the basic standard extension for playbook files.

   When the editor appears, type the following:

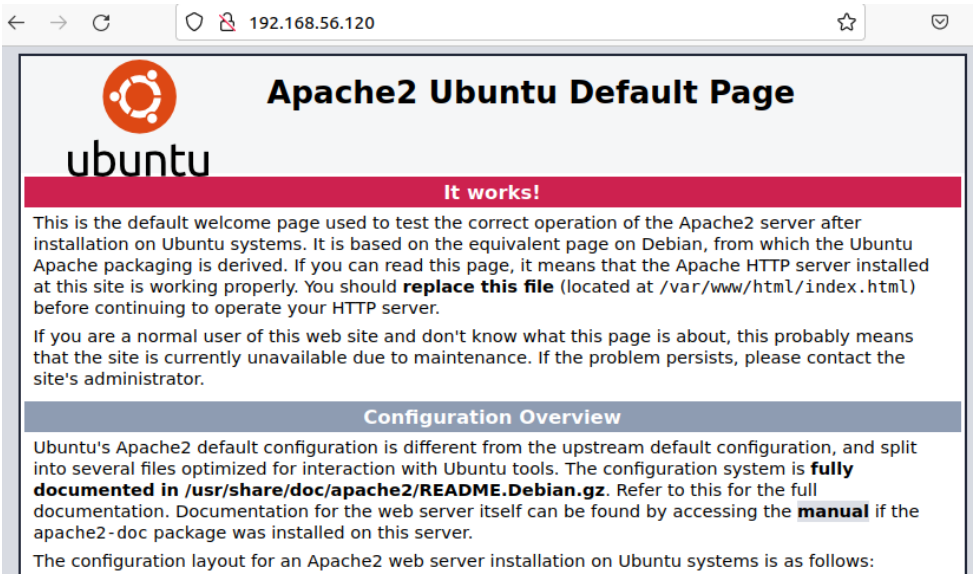
```
  GNU nano 4.8                 install_apache.yml
---
- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    apt:
      name: apache2
```

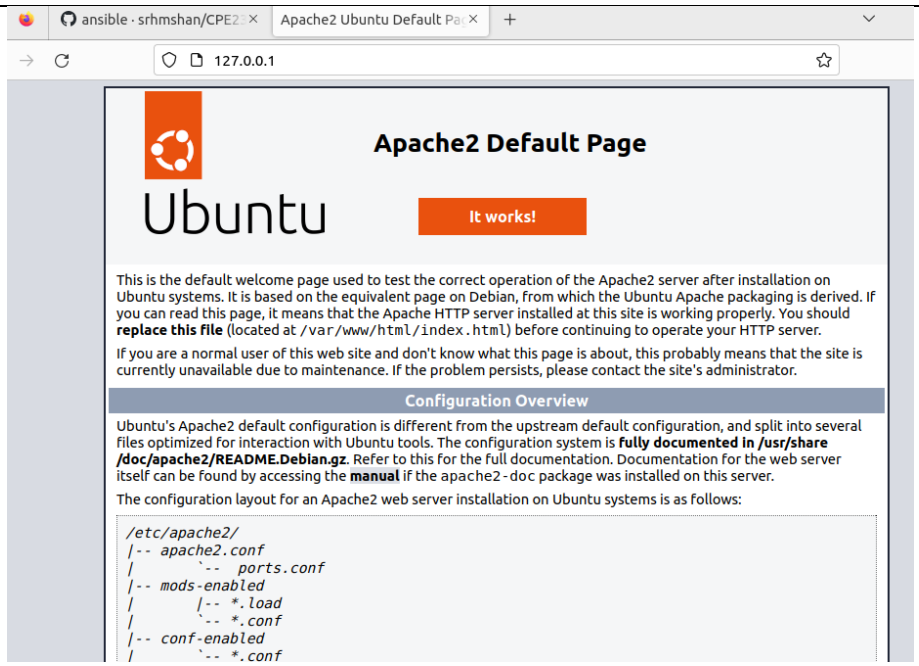
   Make sure to save the file. Take note also of the alignments of the texts.

2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml.* Describe the result of this command.
   The initial task, "Gathering Facts," successfully collected essential host information, while the next task, named "install apache2 package," performed the installation of the 'apache2' package using the 'apt' package manager.

```
srhmshan@localmachine:~/CPE232_Sumaoang$ ansible-playbook --ask-b
ecome-pass install_apache.yml
BECOME password:

PLAY [all] *********************************************************
***************

TASK [Gathering Facts] *********************************************
***************
ok: [127.0.0.1]

TASK [install apache2 package] *************************************
***************
changed: [127.0.0.1]

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

3. To verify that apache2 was installed automatically in the remote servers, go to the web browsers on each server and type its IP address. You should see something like this.





4. Try to edit the *install_apache.yml* and change the name of the package to any name that will not be recognized. What is the output? **There was no change. Since the specified package name could not be recognized or found in the package repository.**

```
  - name: install apache2 package karlo
```

```
srhmshan@localmachine:~/CPE232_Sumaoang$ ansible-playbook --ask-become-pass install_apache.yml
BECOME password:

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [127.0.0.1]

TASK [install apache2 package karlo] *******************************************
ok: [127.0.0.1]

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

5. This time, we are going to put additional task to our playbook. Edit the *install_apache.yml*. As you can see, we are now adding an additional command, which is the *update_cache*. This command updates existing

package-indexes on a supporting distro but not upgrading installed-packages (utilities) that were being installed.

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2
```

Save the changes to this file and exit.

6. Run the playbook and describe the output. Did the new command change anything on the remote servers? **Editing "install_apache.yml" led to changes on the remote server, including updating the package repository index and installing the Apache web server package, as indicated by the "changed" status in the output.**

```
srhmshan@localmachine:~/CPE232_Sumaoang$ ansible-playbook --ask-b
ecome-pass install_apache.yml
BECOME password:

PLAY [all] *********************************************************
***************

TASK [Gathering Facts] *********************************************
***************
ok: [127.0.0.1]

TASK [update repository index] *************************************
***************
changed: [127.0.0.1]

TASK [install apache2 package] *************************************
***************
ok: [127.0.0.1]

PLAY RECAP *********************************************************
***************
127.0.0.1                  : ok=3    changed=1    unreachable=0
  failed=0    skipped=0    rescued=0    ignored=0
```

7. Edit again the *install_apache.yml*. This time, we are going to add a PHP support for the apache package we installed earlier.

```
---
- hosts: all
  become: true
  tasks:

  - name: update repository index
    apt:
      update_cache: yes

  - name: install apache2 package
    apt:
      name: apache2

  - name: add PHP support for apache
    apt:
      name: libapache2-mod-php
```

Save the changes to this file and exit.

8. Run the playbook and describe the output. Did the new command change anything on the remote servers? **It altered the remote servers by updating the package repository index, installing the Apache web server, and adding PHP support to Apache.**

```
srhmshan@localmachine:~/CPE232_Sumaoang$ ansible-playbook --ask-b
ecome-pass install_apache.yml
BECOME password:

PLAY [all] *********************************************************
***************

TASK [Gathering Facts] *********************************************
***************
ok: [127.0.0.1]

TASK [update repository index] *************************************
***************
changed: [127.0.0.1]

TASK [install apache2 package] *************************************
***************
ok: [127.0.0.1]

TASK [add PHP support for apache] **********************************
***************
changed: [127.0.0.1]

PLAY RECAP *********************************************************
***************
127.0.0.1                  : ok=4    changed=2    unreachable=0
  failed=0    skipped=0    rescued=0    ignored=0
```

9. Finally, make sure that we are in sync with GitHub. Provide the link of your GitHub repository.

```
srhmshan@localmachine:~/CPE232_Sumaoang$ git add install_apache.y
ml
srhmshan@localmachine:~/CPE232_Sumaoang$ git commit -m "Apache"
[main 65713dd] Apache
 1 file changed, 16 insertions(+)
 create mode 100644 install_apache.yml
srhmshan@localmachine:~/CPE232_Sumaoang$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 462 bytes | 462.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:srhmshan/CPE232_Sumaoang.git
   c8c8b3b..65713dd  main -> main
```

https://github.com/srhmshan/CPE232_Sumaoang/blob/main/install_apache.yml

Reflections:

Answer the following:

1. What is the importance of using a playbook?

   Ansible Playbooks are vital for automating tasks on remote servers. They enable us to define the desired server state and execute tasks systematically. Playbooks enhance reproducibility, standardization, and efficiency, making them vital for system administrators and DevOps professionals.

2. Summarize what we have done on this activity.

   In this activity, we executed a series of tasks to explore Ansible's automation capabilities. We began by running elevated ad hoc commands, updating package indexes, and installing packages on remote servers. Then, we created a "install_apache.yml" playbook, a fundamental tool for automating server configuration. We tested the playbook, ensuring the successful installation of Apache on remote servers. Further enhancements were made to the playbook, including tasks like updating package indexes and adding PHP support to Apache. This activity underscored the crucial role of automation in maintaining consistent server configurations while minimizing manual intervention, providing hands-on experience with Ansible.