

Name: Repani, Justin Jello J.	Date Performed: September 11, 2023
Course/Section: CPE31S6	Date Submitted: September 11, 2023
Instructor: Dr. Jonathan V. Taylor	Semester and SY: 1st Sem SY 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 not successful

```
jello@workstation:~/CPE232_Repani$ ansible all -m apt -a update_cache=true
192.168.56.103 | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation: Failed to lock director
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
n (13: Permission denied)"
}
192.168.56.102 | FAILED! => {
  "changed": false,
  "msg": "Failed to lock apt for exclusive operation: Failed to lock director
y /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - ope
n (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.

```
jello@workstation:~/CPE232_Repani$ ansible all -m apt -a update_cache=true --be
come --ask-become-pass
BECOME password:
192.168.56.103 | CHANGED => {
  "cache_update_time": 1694428581,
  "cache_updated": true,
  "changed": true
}
192.168.56.102 | CHANGED => {
  "cache_update_time": 1694428581,
  "cache_updated": true,
  "changed": true
}
```

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

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.

```
jello@workstation:~/CPE232_Repani$ ansible all -m apt -a name=vim-nox --become
--ask-become-pass
BECOME password:
192.168.56.103 | CHANGED => {
  "cache_update_time": 1694428581,
  "cache_updated": false,
  "changed": true,
  "stderr": "",
  "stderr_lines": [],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading s
tate information...\nThe following package was automatically installed and is n
o longer required:\n  libllvm7\nUse 'sudo apt autoremove' to remove it.\nThe fo
llowing additional packages will be installed:\n  fonts-lato javascript-common
libjs-jquery liblua5.2-0 libruby2.5 libtcl8.6\n  rake ruby ruby-did-you-mean ru
by-minitest ruby-net-telnet ruby-power-assert\n  ruby-test-unit ruby2.5 rubygem
s-integration vim-runtime\nSuggested packages:\n  apache2 | lighttpd | httpd tc
l8.6 ri ruby-dev bundler cscope vim-doc\nThe following NEW packages will be ins
talled:\n  fonts-lato javascript-common libjs-jquery liblua5.2-0 libruby2.5 lib
tcl8.6\n  rake ruby ruby-did-you-mean ruby-minitest ruby-net-telnet ruby-power-
assert\n  ruby-test-unit ruby2.5 rubygems-integration vim-nox vim-runtime\n0 up
graded, 17 newly installed, 0 to remove and 0 not upgraded.\nNeed to get 13.8 M
B of archives.\nAfter this operation, 64.5 MB of additional disk space will be
used.\nGet:1 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato a
) in auto mode",
  "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/view (
view) in auto mode",
  "update-alternatives: using /usr/bin/vim.nox to provide /usr/bin/ex (ex
) in auto mode",
  "Processing triggers for libc-bin (2.27-3ubuntu1.6) ...",
  "Processing triggers for man-db (2.8.3-2ubuntu0.1) ...",
  "Processing triggers for fontconfig (2.12.6-0ubuntu2) ..."
]
}
jello@workstation:~/CPE232_Repani$
```

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 is successfully run

```
jello@workstation:~/CPE232_Repani$ which vim
jello@workstation:~/CPE232_Repani$ apt search vim-nox
Sorting... Done
Full Text Search... Done
vim-nox/bionic-updates,bionic-security 2:8.0.1453-1ubuntu1.13 amd64
  Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/bionic-updates,bionic-security,now 2:8.0.1453-1ubuntu1.13 amd64 [insta
lled]
  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*.

```
jello@workstation:~/CPE232_Repani$ cd /var/log
jello@workstation:/var/log$ ls
alternatives.log  btmp.1      installer    syslog.2.gz
alternatives.log.1 cups         journal      syslog.3.gz
apt              dist-upgrade kern.log     tallylog
auth.log         dpkg.log    kern.log.1  ubuntu-advantage.log
auth.log.1       dpkg.log.1  kern.log.2.gz ubuntu-advantage.log.1
auth.log.2.gz    faillog     kern.log.3.gz unattended-upgrades
auth.log.3.gz    fontconfig.log lastlog      wtmp
boot.log         gdm3        speech-dispatcher wtmp.1
bootstrap.log    gpu-manager.log syslog
btmp            hp          syslog.1

jello@workstation:/var/log$ cd apt
jello@workstation:/var/log/apt$ cat history.log

Start-Date: 2023-09-11 17:02:13
Commandline: apt install python-pip
Requested-By: jello (1000)
Install: libgcc-7-dev:amd64 (7.5.0-3ubuntu1~18.04, automatic), python-six:amd64 (1.11.0-2, automatic), libmpx2:amd64 (8.4.0-1ubuntu1~18.04, automatic), python 2.7-dev:amd64 (2.7.17-1~18.04ubuntu1.11, automatic), linux-libc-dev:amd64 (4.15.0-213.224, automatic), python-xdg:amd64 (0.25-4ubuntu1.1, automatic), libfakeroot:amd64 (1.22-2ubuntu1, automatic), libc6-dev:amd64 (2.27-3ubuntu1.6, automatic), libexpat1-dev:amd64 (2.2.5-3ubuntu0.9, automatic), python2.7-minimal:amd64 (2.7.17-1~18.04ubuntu1.11, automatic), libalgorithm-diff-perl:amd64 (1.19.03-1, automatic), python-gi:amd64 (3.26.1-2ubuntu1, automatic), libalgorithm-merge-perl:amd64 (0.08-3, automatic), python-dbus:amd64 (1.2.6-1, automatic), libitm1:amd64 (8.4.0-1ubuntu1~18.04, automatic), libpython-all-dev:amd64 (2.7.15~rc1-1

Start-Date: 2023-09-11 17:05:28
Commandline: apt install python3-pip
Requested-By: jello (1000)
Install: python3-dev:amd64 (3.6.7-1~18.04, automatic), python3-distutils:amd64 (3.6.9-1~18.04, automatic), libpython3.6-dev:amd64 (3.6.9-1~18.04ubuntu1.12, automatic), python3-pip:amd64 (9.0.1-2.3~ubuntu1.18.04.8), python3-wheel:amd64 (0.30.0-0.2ubuntu0.1, automatic), python3-lib2to3:amd64 (3.6.9-1~18.04, automatic), dh-python:amd64 (3.20180325ubuntu2, automatic), libpython3-dev:amd64 (3.6.7-1~18.04, automatic), python3.6-dev:amd64 (3.6.9-1~18.04ubuntu1.12, automatic), python3-setuptools:amd64 (39.0.1-2ubuntu0.1, automatic)
End-Date: 2023-09-11 17:05:33
jello@workstation:/var/log/apt$
```

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 command is a success. It installs a file or program with the name of `snapt` and the `ask become pass` asks for the password of the root user.

```
jello@workstation:~/CPE232_Repani$ ansible all -m apt -a name=snapt --ask-become-pass
BECOME password:
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694428581,
    "cache_updated": false,
    "changed": false
}
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694428581,
    "cache_updated": false,
    "changed": false
}
```

3.2 Now, try to issue this command: *ansible all -m apt -a "name=snapt 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.

- The command functioned the same way as the previous one

```
jello@workstation:~/CPE232_Repani$ ansible all -m apt -a "name=snapt state=latest" --become --ask-become-pass
BECOME password:
192.168.56.102 | SUCCESS => {
    "cache_update_time": 1694428581,
    "cache_updated": false,
    "changed": false
}
192.168.56.103 | SUCCESS => {
    "cache_update_time": 1694428581,
    "cache_updated": false,
    "changed": false
}
```

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

```
jello@workstation:~/CPE232_Repani$ git add *
jello@workstation:~/CPE232_Repani$ git commit -m "ansible basic comma
[main 409442f] ansible basic commands
 2 files changed, 4 insertions(+), 4 deletions(-)
jello@workstation:~/CPE232_Repani$ git push origin
Username for 'https://github.com': JelzLow
Password for 'https://JelzLow@github.com':
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 437 bytes | 437.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/JelzLow/CPE232_Repani.git
   d8809d1..409442f  main -> main
jello@workstation:~/CPE232_Repani$
```

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.

```
jello@workstation: ~/CPE232_Repani
File Edit View Search Terminal Help
GNU nano 2.9.3      install_apache.yml      Modified
---
- hosts: all
  become: true
  tasks:
    - name: install apache2 package
      apt:
        name: apache2

File Name to Write: install_apache.yml
^G Get Help      M-D DOS Format  M-A Append      M-B Backup File
^C Cancel        M-M Mac Format  M-P Prepend     ^T To Files

jello@workstation:~/CPE232_Repani$ ls
ansible.cfg  install_apache.yml  inventory  README.md
```

2. Run the yml file using the command: *ansible-playbook --ask-become-pass install_apache.yml*. Describe the result of this command.
 - The command runs the playbook we have created which contains the command to download a package called apache2 on all the hosts


```

jello@workstation: ~/CPE232_Repani
File Edit View Search Terminal Help
jello@workstation:~/CPE232_Repani$ nano install_apache.yml
jello@workstation:~/CPE232_Repani$ ls
ansible.cfg  install_apache.yml  inventory  README.md
jello@workstation:~/CPE232_Repani$ ansible-playbook --ask-become-pass in
apache.yml
BECOME password:

PLAY [all] *****
*

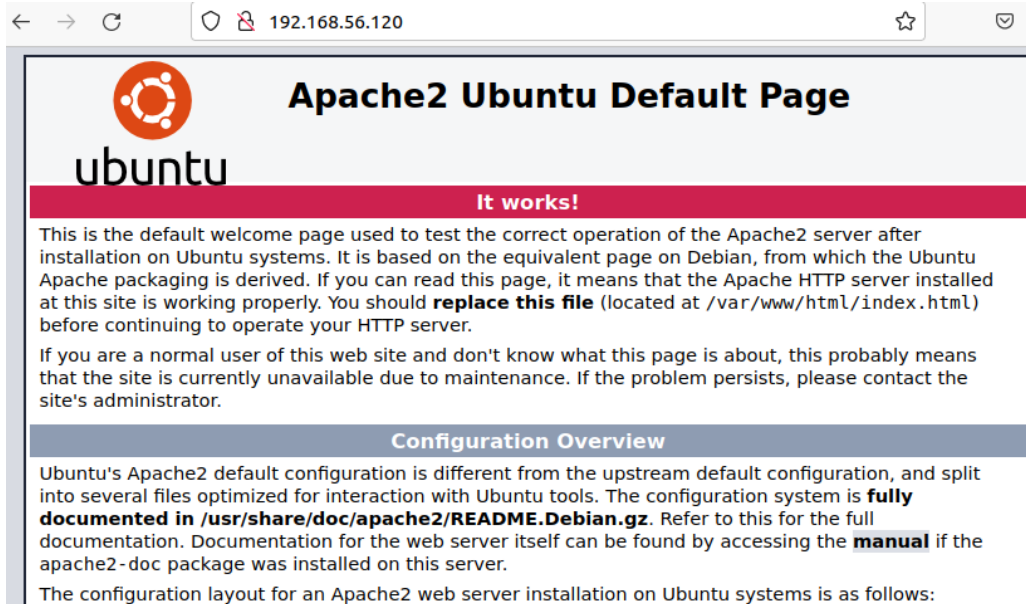
TASK [Gathering Facts] *****
*
ok: [192.168.56.103]
ok: [192.168.56.102]

TASK [install apache2 package] *****
*
changed: [192.168.56.102]
changed: [192.168.56.103]

PLAY RECAP *****
*
192.168.56.102      : ok=2    changed=1    unreachable=0    failu
skipped=0    rescued=0    ignored=0
192.168.56.103      : ok=2    changed=1    unreachable=0    failu
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?

```

jello@workstation:~/CPE232_Repani$ nano install_apache.yml
jello@workstation:~/CPE232_Repani$ mv install_apache.yml install_apacheEDIT.yml
jello@workstation:~/CPE232_Repani$ ansible-playbook --ask-become-pass install_a
pache.yml
ERROR! the playbook: install_apache.yml could not be found

```

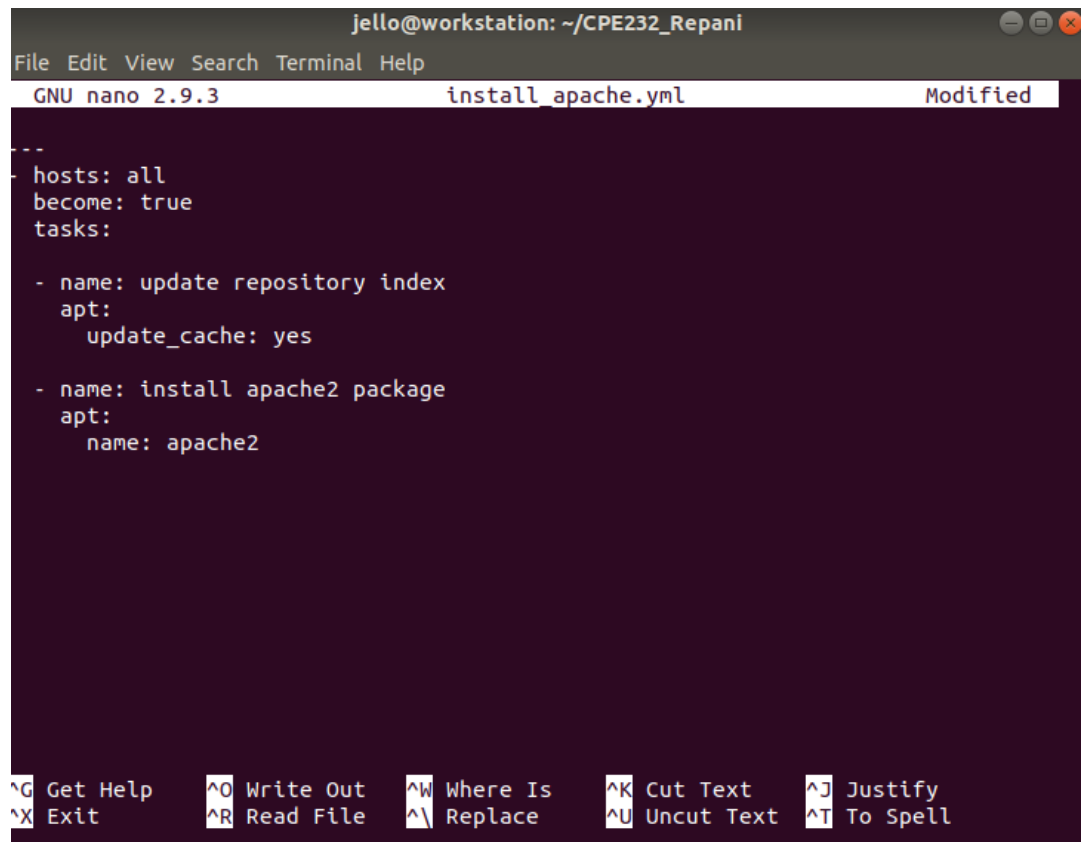
5. This time, we are going to put additional task to our playbook. Edit the `install_apache.yml`. As you can see, we are now adding an additional command, which is the `update_cache`. This command updates existing package-indexes on a supporting distro but not upgrading installed-packages (utilities) 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.



```
jello@workstation: ~/CPE232_Repani
File Edit View Search Terminal Help
GNU nano 2.9.3 install_apache.yml Modified
---
- hosts: all
  become: true
  tasks:

    - name: update repository index
      apt:
        update_cache: yes

    - name: install apache2 package
      apt:
        name: apache2
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

6. Run the playbook and describe the output. Did the new command change anything on the remote servers?
 - Yes. The command downloaded the apache2 package while also updating the repository index cache

```
jello@workstation:~/CPE232_Repani$ ansible-playbook --ask-become-pass install_
apache.yml
BECOME password:

PLAY [all] *****
*

TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [update repository index] *****
*
changed: [192.168.56.103]
changed: [192.168.56.102]

TASK [install apache2 package] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]

PLAY RECAP *****
*
192.168.56.102      : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.103      : 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.

```
jello@workstation: ~/CPE232_Repani
File Edit View Search Terminal Help
GNU nano 2.9.3 install_apache.yml Modified
---
- 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

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

8. Run the playbook and describe the output. Did the new command change anything on the remote servers?
- Yes. The command functioned the same way as the previous two but this time i also added the PHP support for apache by installing the php mod

```

jello@workstation:~/CPE232_Repani$ ansible-playbook --ask-become-pass install_
apache.yml
BECOME password:

PLAY [all] *****
*

TASK [Gathering Facts] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]

TASK [update repository index] *****
*
changed: [192.168.56.103]
changed: [192.168.56.102]

TASK [install apache2 package] *****
*
ok: [192.168.56.102]
ok: [192.168.56.103]


TASK [add PHP support for apache] *****
*
changed: [192.168.56.103]
changed: [192.168.56.102]

PLAY RECAP *****
*
192.168.56.102      : ok=4    changed=2    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
192.168.56.103      : ok=4    changed=2    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
jello@workstation:~/CPE232_Repani$ █

```

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

```
jello@workstation:~/CPE232_Repani$ git add *
jello@workstation:~/CPE232_Repani$ git commit -m "ansible basic commands"
[main 7cbe1b6] ansible basic commands
1 file changed, 16 insertions(+)
create mode 100644 install_apache.yml
jello@workstation:~/CPE232_Repani$ git push origin
Username for 'https://github.com': JelzLow
Password for 'https://JelzLow@github.com':
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 501 bytes | 501.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/JelzLow/CPE232_Repani.git
409442f..7cbe1b6 main -> main
jello@workstation:~/CPE232_Repani$
```


CPE232_Repani
Public
Pin
Unwatch 1

main
1 branch
0 tags
Go to file
Add file
Code

	JelzLow ansible basic commands	7cbe1b6 1 minute ago	5 commits
README.md	My first commit	3 weeks ago	
ansible.cfg	ansible basic commands	18 minutes ago	
install_apache.yml	ansible basic commands	1 minute ago	
inventory	ansible basic commands	18 minutes ago	

[JelzLow/CPE232_Repani: SysAdS6 \(github.com\)](https://github.com/JelzLow/CPE232_Repani)
https://github.com/JelzLow/CPE232_Repani

Reflections:

Answer the following:

- What is the importance of using a playbook?
 - The importance of using a playbook is that it enables us to efficiently manage multiple hosts without having to manually access each host and perform the actions such as installing packages, updating software, and all the other commands you could normally perform on the command line. Having playbook simply needs us to create the series of commands in a note file which can be run at any time.
- Summarize what we have done on this activity.
 - In this hands-on activity I have learned all about the basic ansible commands. Firstly it is about installing python and ansible in order to create the playbooks.

The next part of the activity is cloning a github repository in which we used the previously learned commands for syncing files from the local machine to the git database. The next part is using the ansible commands in order to create a configuration file, inventory file, and the playbook with various commands to run.

Conclusion:

In this hands-on activity 4, the topic is all about the basic ansible commands, its purpose, syntax, and function. The lesson and activity covered how to install the required dependencies in order to make ansible run and as well as performing the different basic ansible commands such as creating the configuration file, inventory file, creating a playbook, running a playbook, and as well as the previously learned skills with Git such as cloning a git repository, adding, committing, and pushing files onto the repository in order to sync the local machine and the database.

Honor Pledge

"I affirm that I have not given or received any unauthorized help on this assignment, and that this work is my own."