| **Name:** Torrecampo, Juan Piolo S. | **Date Performed:** Sep 13, 2022 |
| --- | --- |
| **Course/Section:** CPE 232 / CPE31S22 | **Date Submitted:** Sep 13, 2022 |
| **Instructor:** Dr. Jonathan Taylar | **Semester and SY:** 1st Sem, 2022 - 2023 |
| **Activity 5: Consolidating Playbook plays** | |
| 1. **Objectives:**    1. Use **when** command in playbook for different OS distributions    2. Apply refactoring techniques in cleaning up the playbook codes | |
| 1. **Discussion:**   We are going to look at a way that we can differentiate a playbook by a host in terms of which distribution the host is running. It's very common in most Linux shops to run multiple distributions, for example, Ubuntu shop or Debian shop and you need a different distribution for a one off-case or perhaps you want to run plays only on certain distributions.   It is a best practice in ansible when you are working in a collaborative environment to use the command git pull. git pull is a Git command used to update the local version of a repository from a remote. By default, git pull does two things. Updates the current local working branch (currently checked out branch) and updates the remote-tracking branches for all other branches. git pull essentially pulls down any changes that may have happened since the last time you worked on the repository.  **Requirement:** In this activity, you will need to create a CentOS VM. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the CentOS VM. Make sure to use the command *ssh-copy-id* to copy the public key to CentOS. Verify if you can successfully SSH to CentOS VM. | |
| **Task 1: Use when command for different distributions**   1. In the local machine, make sure you are in the local repository directory (*CPE232\_yourname*). Issue the command git pull. When prompted, enter the correct passphrase or password. Describe what happened when you issue this command. Did something happen? Why?  * **It pulls the git repository and gets the latest update. It does not change anything because we did not add anything into our previous repository. That is why the command displays it is “Already up to date”.**      1. Edit the inventory file and add the IP address of the Centos VM. Issue the command we used to execute the playbook (the one we used in the last activity): *ansible-playbook --ask-become-pass install\_apache.yml*. After executing this command, you may notice that it did not become successful in the Centos VM. You can see that the Centos VM has failed=1. Only the two remote servers have been changed. The reason is that Centos VM does not support "apt" as the package manager. The default package manager for Centos is "yum."   **Configuring Inventory**    **Executing “ansible-playbook –ask-become-pass install\_apache.yml”**     1. Edit the *install\_apache.yml* file and insert the lines shown below.     Make sure to save the file and exit.    Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.    If you have a mix of Debian and Ubuntu servers, you can change the configuration of your playbook like this.   * name: update repository index   apt:  update\_cache: yes  when: ansible\_distribution in [“Debian”, “Ubuntu]    *Note*: This will work also if you try. Notice the changes are highlighted.     1. Edit the *install\_apache.yml* file and insert the lines shown below.     Make sure to save and exit.    Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.     1. To verify the installations, go to CentOS VM and type its IP address on the browser. Was it successful? The answer is no. It’s because the httpd service or the Apache HTTP server in the CentOS is not yet active. Thus, you need to activate it first.     5.1 To activate, go to the CentOS VM terminal and enter the following:  *systemctl status httpd*  The result of this command tells you that the service is inactive.     * 1. Issue the following command to start the service:   *sudo systemctl start httpd*    (When prompted, enter the sudo password)  *sudo firewall-cmd --add-port=80/tcp*  (The result should be a success)     * 1. To verify the service is already running, go to CentOS VM and type its IP address on the browser. Was it successful? (Screenshot the browser) | |
| **Task 2:** **Refactoring playbook**  This time, we want to make sure that our playbook is efficient and that the codes are easier to read. This will also makes run ansible more quickly if it has to execute fewer tasks to do the same thing.   1. Edit the playbook *install\_apache.yml*. Currently, we have three tasks targeting our Ubuntu machines and 3 tasks targeting our CentOS machine. Right now, we try to consolidate some tasks that are typically the same. For example, we can consolidate two plays that install packages. We can do that by creating a list of installation packages as shown below:     Make sure to save the file and exit.    Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.     1. Edit the playbook *install\_apache.yml* again. In task 2.1, we consolidated the plays into one play. This time we can actually consolidated everything in just 2 plays. This can be done by removing the update repository play and putting the command *update\_cache: yes* below the command *state: latest*. See below for reference:     Make sure to save the file and exit.    Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.     1. Finally, we can consolidate these 2 plays in just 1 play. This can be done by declaring variables that will represent the packages that we want to install. Basically, the apache\_package and php\_package are variables. The names are arbitrary, which means we can choose different names. We also take out the line when: ansible\_distribution. Edit the playbook *install\_apache.yml* again and make sure to follow the below image. Make sure to save the file and exit.       Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result.     1. Unfortunately, task 2.3 was not successful. It’s because we need to change something in the inventory file so that the variables we declared will be in place. Edit the *inventory* file and follow the below configuration:     Make sure to save the *inventory* file and exit.    **Finally**, we still have one more thing to change in our *install\_apache.yml* file. In task 2.3, you may notice that the package is assign as apt, which will not run in CentOS. Replace the *apt* with *package*. Package is a module in ansible that is generic, which is going to use whatever package manager the underlying host or the target server uses. For Ubuntu it will automatically use *apt*, and for CentOS it will automatically use *dnf*. Make sure to save the file and exit. For more details about the ansible package, you may refer to this documentation: [ansible.builtin.package – Generic OS package manager — Ansible Documentation](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html)    Run *ansible-playbook --ask-become-pass install\_apache.yml* and describe the result. | |
| **Reflections:**  Answer the following:   * 1. Why do you think refactoring of playbook codes is important? * In doing refactoring, we are simplifying the massive yml configuration file into smaller and flexible code that can handle a bunch of commands that are suited for different operating systems without specifying it. As I observe in the task it performs the refactoring in a phase which shows on how we can simplify our yml ansible code into smaller version of it plus a feature of flexibility by defining a variable. Where the variable is stated in the yml file while the values are stated in the inventory file under ansible directory.   1. When do we use the “when” command in playbook? * The “when” command is a basic conditional command for ansible yml configuration files. This is used when we are integrating a condition to a process like the “when: ansible\_distributtion = osperating\_system”. If this command is inside of a block of process then it will first check whether the operating system matches the server before running the command. Therefore, this command is used in a wide variety of different yml configurations but mostly used in defining an operating system. | |
| **Honor Pledge** | |
| *“I affirm that I will not give or receive unauthorized help on this activity and that all will be my own.”* | |