Guided Practice: Create Custom Bash Modules

Outcome

In this Guided Practice, you will create a few custom Bash modules. While Ansible comes with several modules, you might have the need to create your own. Modules are segments of code that can be triggered from a playbook. Ansible allows you to write modules in any language. You should have already completed the Guided Practice to create a custom Python module. Another good option is Bash.

Resources Needed

- VCASTLE Pod configured for the class. For this Guided Practice, we use the CentOS 8 and Ubuntu 20.04 LTS machines.
- Your user needs to be able to elevate their privileges with sudo.

Level of Difficulty

Moderate

Deliverables

Deliverables are marked in red font or with a red picture border around the screenshot. Additionally, there are questions at the end. **Your username or studentid should be visible in all screenshots that you submit.**

General Considerations

Ansible should already be installed on your machine from a prior Guided Practice.

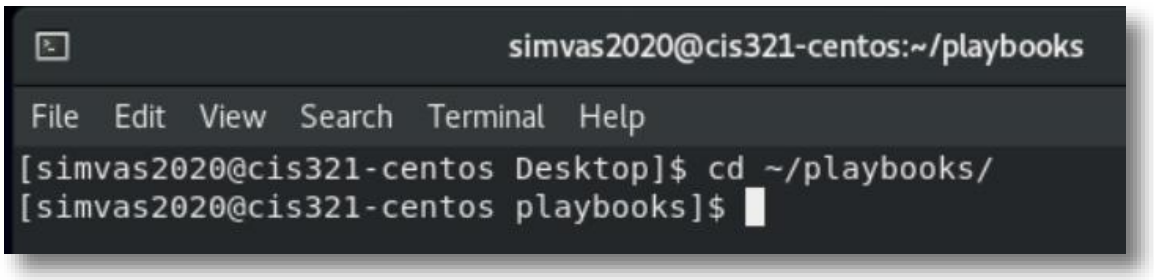
Create Your First BASH Module

Modules are building blocks of playbooks. We will use a text editor to write our very first custom BASH module.

1. Check that Python is installed (it should be, but let's check anyway).
   a. Log onto the CentOS computer using your own user.
   b. Open a terminal. Change to the playbooks directory (this should have been created in the previous guided practice). Type:

```
cd ~/playbooks
```

   Your screen should resemble this:



2. We will write a BASH module to create a Linux user and change their password. Create and edit the file **add_user.sh** using nano. The file should be under the library directory (this is a subdirectory of playbooks and should have also been created in the previous Guided Practice). Type:

```
nano ./library/add_user.sh
```

   a. Type the following in the new file. Then, save the file with Ctrl+S, and exit with Ctrl+X.

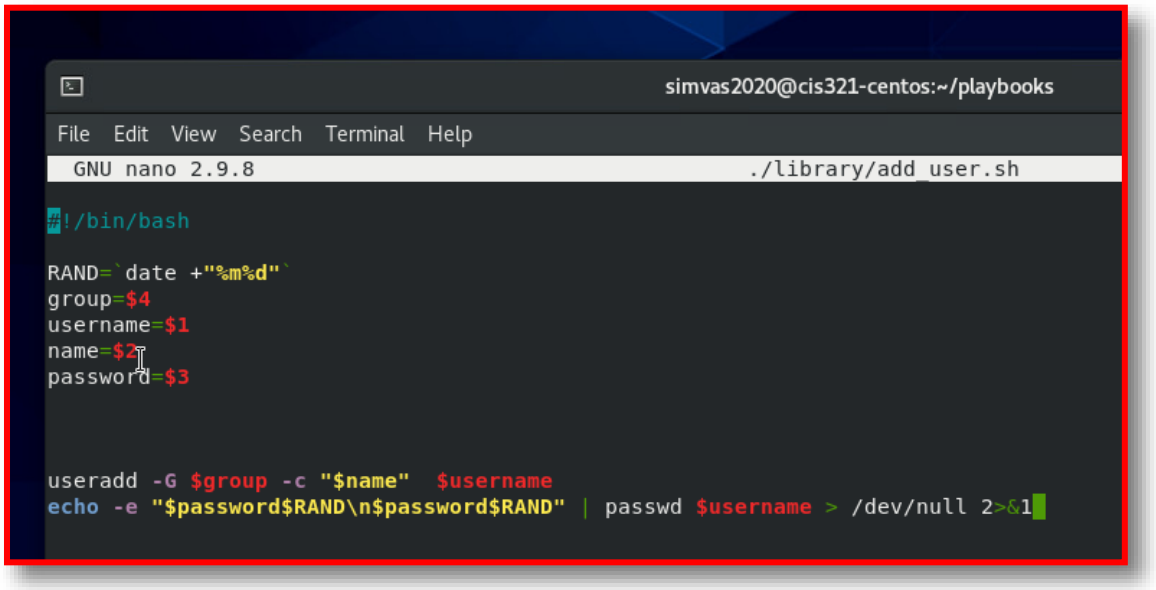
```
#!/bin/bash

RAND=`date +"%m%d"`
group=$4
username=$1
name=$2
password=$3



useradd -G $group -c "$name"  $username
```

```
echo -e “$password$RAND\n$password$RAND” | passwd $username >
/dev/null 2>&1
```

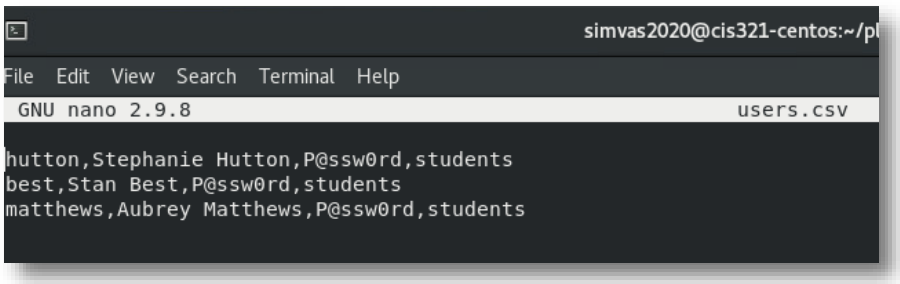


Take a screenshot for your lab report.

This script expects four arguments in the order: username, name, password and primary group name. Given these arguments, the script will create a user with the appropriate details and will set the password for the created user to the password string + month and day (for instance, if the password string is P@ssw0rd, and the user account is created on December 22, the password of the user will be set to P@ssw0rd1220).

b. You will create a CSV file, and this file will contain your user detail information. You do not have to enter the same exact information, but the file needs to be organized similarly. The file is comma-separated and the first field represents the username, the second the name, the third the password string, the fourth the primary group name. The file name is **users.csv** and is under the playbooks directory. If you name your file differently or save it elsewhere, please make the necessary adjustments in the Ansible playbook as well. Sample file below:

```
shutton,Stephanie Hutton,P@ssw0rd,students
sbest,Stan Best,P@ssw0rd,students
amatthews,Aubrey Matthews,P@ssw0rd,students
```



c. This activity assumes that the group(s) the users are added to already exist. Sample syntax for creating the students group is provided below. If you decide to add your new users to one or more different groups, please adjust accordingly.

```
sudo groupadd students
```

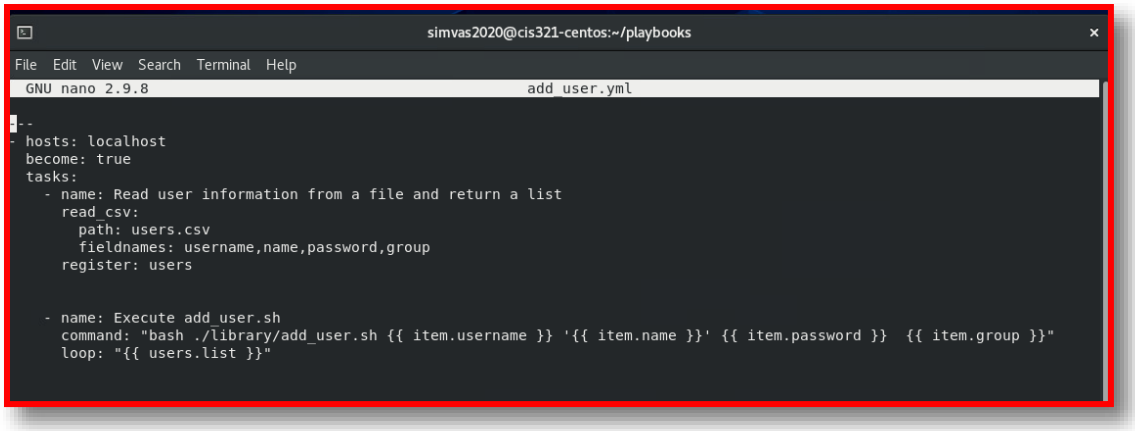
d. You will now create the playbook to help you test this very simple custom BASH module. In your CentOS terminal, type:

```
nano add_user.yml
```

e. Create the content below. Then save the file by typing Ctrl+S, and exit nano by typing Ctrl +X.

```
---
- hosts: localhost
  become: true
  tasks:
    - name: Read user information from a file and return a list
      read_csv:
        path: users.csv
        fieldnames: username,name,password,group
      register: users


    - name: Execute add_user.sh
      command: "bash ./library/add_user.sh {{ item.username }} '{{
item.name }}' {{ item.password }} {{ item.group }}"
      loop: "{{ users.list }}"
```

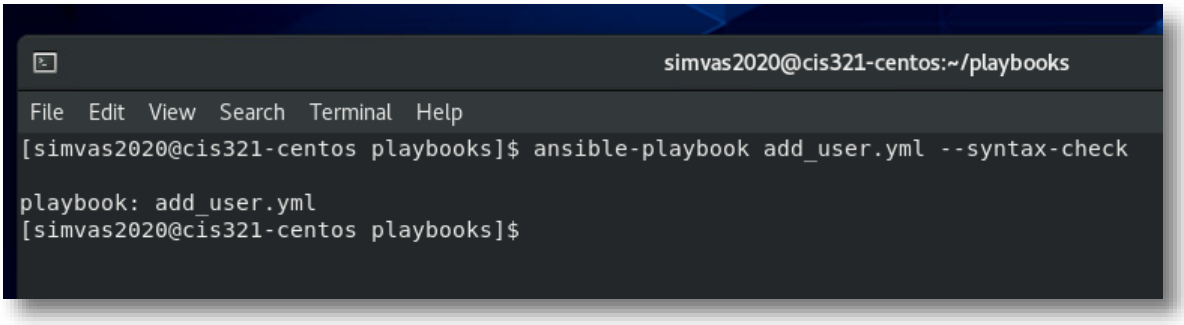


Take a screenshot for your lab report.

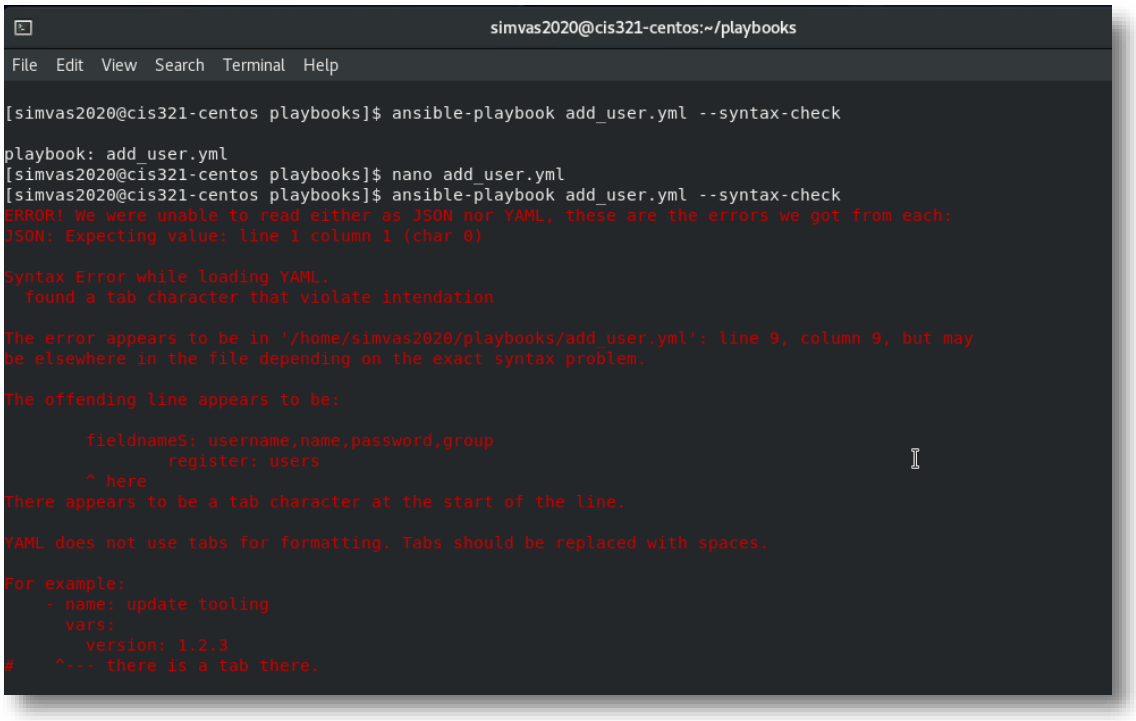
3. Part of the focus of this week is to introduce you to some options to validate your code and teach you how to debug it. Ansible offers you an option to check the syntax of a playbook for errors (syntax errors) before you run the playbook. That option is **- -syntax-check**. In the CentOs terminal, type:

```
ansible-playbook add_user.yml --syntax-check
```

If your syntax is correct, your output should resemble the one below, but there might be errors to correct. For instance, YAML does not use tabs to format the code. So, if you used tabs, your output might look like the second screenshot, and you will have to use the provided error messages to try to correct your code. You will have to repeat the process until your code is error free.

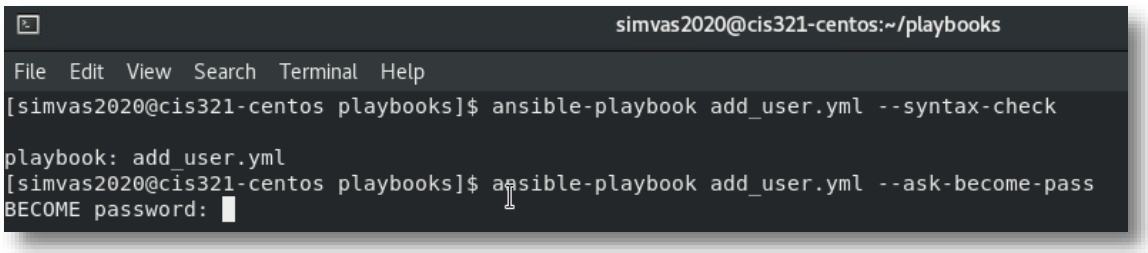


Sample errors can be seen in the image below:



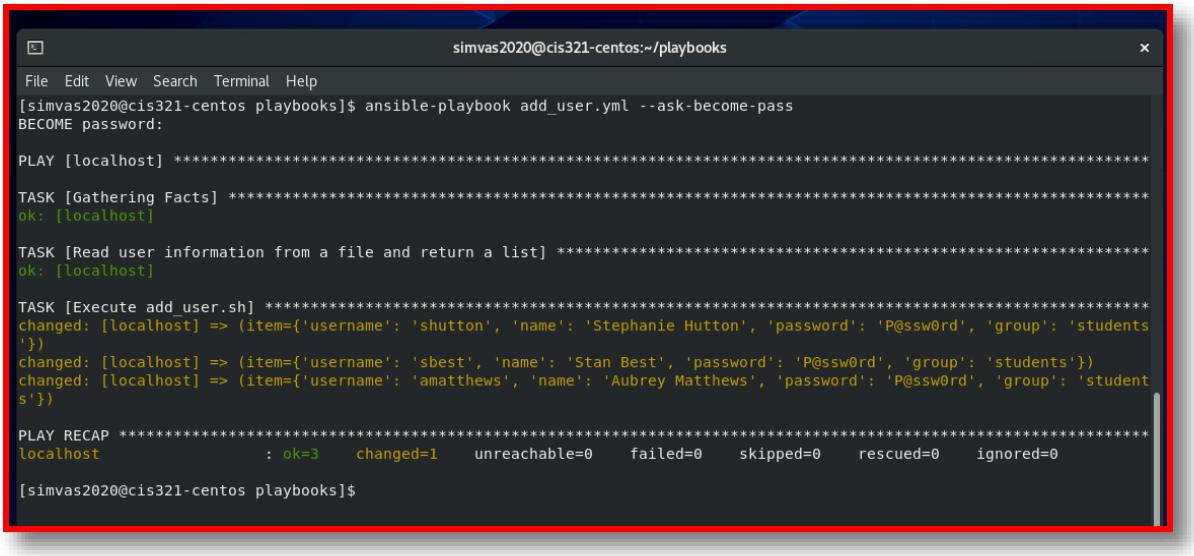
4. After correcting the syntax of your YAML code, you can try to run the playbook by typing:

```
ansible-playbook add_user.yml --ask-become-pass
```



However, you might encounter some issues with your BASH module syntax or your logic (for instance, running without the **--ask-become-pass** option), and you might need to correct those as well. As we saw in the previous Guided Practice, there can be quite a few sources or errors. You can try to correct the error where it occurs by looking at the error message Ansible provided you with, or you can use the –vvv option. Correct these errors as well, and run your playbook.

Your output should resemble the one below:

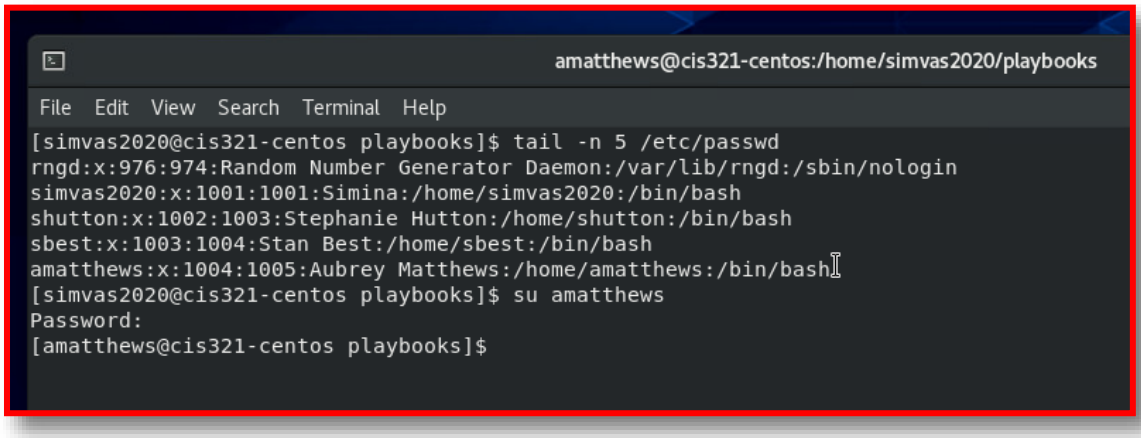


Take a screenshot for your lab report.

5. Verify the users were indeed created, and try to log in as one of the users you created by typing the following:

```
tail –n 5 /etc/passwd
su username

(Enter the password when prompted)
```



Take a screenshot for your lab report.

Guided Practice Questions

In your **Guided Practice Lab Report**, in addition to the screenshots, include answers for the following questions about this learning activity. Some may require research.

1. List at least two formatting/syntax guidelines for BASH.
2. What were some of the errors you received, and how did you fix them? After researching the syntax/formatting guidelines for question 1, can you explain why you received the errors and how you fixed them? **(An explanation is necessary – do not answer YES/NO.)**