Debugging Puppet Installation

Introduction

You're a member of the IT team at your company. One of your coworkers has recently set up a Puppet installation, but it doesn't seem to be doing its job. So, you're asked to debug the profile class, which is supposed to append a path to the environment variable `$PATH`. But it's somehow misbehaving and causing the `$PATH` variable to be broken. You'll need to locate the issue and fix it with the knowledge that you learned in this module.

What you'll do

- Check out what Puppet rules look like
- Run the Puppet agent locally
- Understand how file permissions are represented by numbers
- Learn how scripts under `/etc/profile.d/` perform startup tasks, including setting up your own environment variables
- Append a string to an environment variable

Puppet rules

The Puppet rules are present in the init file. The purpose of this script is to append /java/bin to the environment variable $PATH, so that scripts in that directory can be executed without writing the full path.

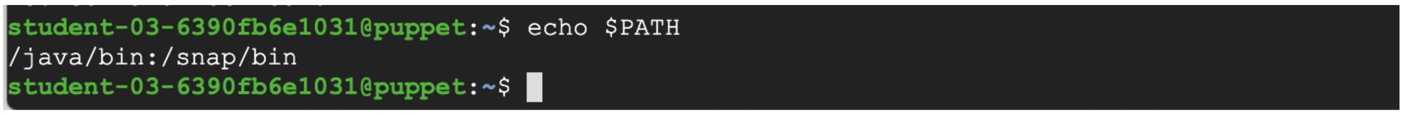
The purpose of this rule is to append `/java/bin` to the environment variable `$PATH` so that scripts in that directory can be executed without writing the full path.

Issue detection

Now, take a look at the environment variable `$PATH` by running the following command:

```
echo $PATH
```

Output:



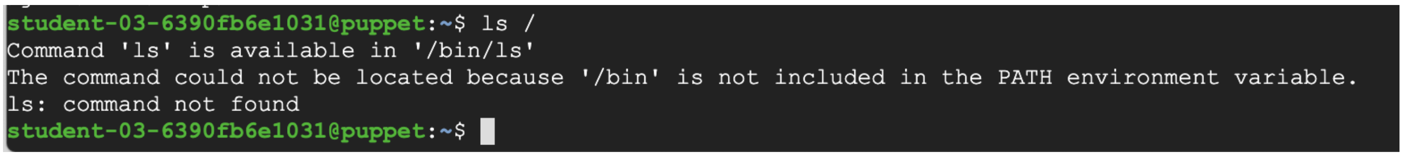
```
student-03-6390fb6e1031@puppet:~$ echo $PATH
/java/bin:/snap/bin
student-03-6390fb6e1031@puppet:~$ 
```

Oops, something is wrong; the main directories used for executing binaries in Linux (`/bin` and `/usr/bin`) are missing.

You can check this by executing the following command:

```
ls /
```

Output:



```
student-03-6390fb6e1031@puppet:~$ ls /
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH environment variable.
ls: command not found
student-03-6390fb6e1031@puppet:~$ 
```

Commands like `ls`, `cd`, `mkdir`, `rm`, and others are just small programs that usually live inside a directory on our systems called `/usr/bin`.

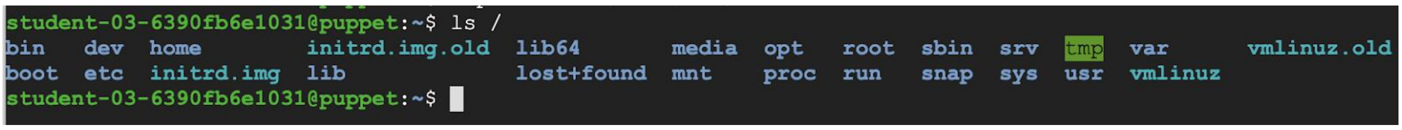
To get back to a working environment, run the following command:

```
export PATH=/bin:/usr/bin
```

By running this command, you manually add the directories that contain executing binaries (`/bin` and `/usr/bin`) to the environment variable `$PATH`. That way, the system will be able to find the commands when you try to run them. Now, run the list directory command, again:

```
ls /
```

Output:



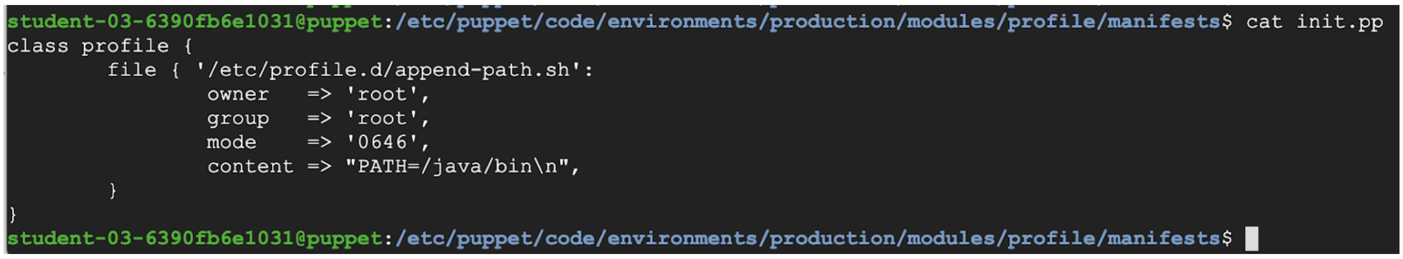
```
student-03-6390fb6e1031@puppet:~$ ls /
bin   dev  home        initrd.img.old  lib64       media  opt   root  sbin  srv  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lost+found  mnt    proc  run   snap  sys  usr  vmlinuz
student-03-6390fb6e1031@puppet:~$
```

Alright, now that we have a working PATH, let's look at the rule responsible for this breakage. It's located in the **profile** module of Puppet's **production** environment. To look at it, go to the `manifests/` directory that contains the Puppet rules by using the following command:

```
cd /etc/puppet/code/environments/production/modules/profile/manifests
```

Use `cat` to check out the contents of the `init.pp` file:

```
cat init.pp
```



```
student-03-6390fb6e1031@puppet:/etc/puppet/code/environments/production/modules/profile/manifests$ cat init.pp
class profile {
        file { '/etc/profile.d/append-path.sh':
                owner   => 'root',
                group   => 'root',
                mode    => '0646',
                content => "PATH=/java/bin\n",
        }
}
student-03-6390fb6e1031@puppet:/etc/puppet/code/environments/production/modules/profile/manifests$
```

This rule is creating a script under `/etc/profile.d/`. Scripts in this path will perform startup tasks, including setting up a user's own environment variables. The files under `/etc/profile.d/` should only be editable by root.

We see that the file resource is part of the **profile** class. The general rules for creating a Puppet class definition used here are:

- The class definition starts with the "class" keyword, followed by the name of the class. In this case, "profile".
- The contents of the class are defined between curly braces and generally contain at least one resource declaration. In this case, a "file" resource.

There could be other resources in this class, but for now it has only one file resource. Let's look at what it's doing. The file defined by this resource is '`/etc/profile.d/append-path.sh`', and the Puppet rule is using some of the available "file" attributes:

- It sets both the owner and group of the file to "root".

- It then sets the "mode" of the file to "0646". This number represents the permissions the file will have.

- You might remember that every file and directory on a Linux system is assigned permissions for three groups of people: the owner, the group and the others.. And for each group, the permissions refer to the possibility of reading, writing and executing the file.

- It's common to use numbers to represent the permissions: 4 for read, 2 for write and 1 for execute. The sum of the permissions given to each of the groups is then a part of the final number. For example, a permission of 6 means read and write, a permission of 5 means read and execute, and a permission of 7 means read, write and execute.

- In this example, we are using 4 numbers. The first one represents any special permissions that the file has (no special permissions). The second one is the permissions for the owner, (read and write), and then come the permissions for the group (read), and finally the permissions for the others (read and write)

- Finally, it sets the actual contents of the file. Here, the content is being set to`"PATH=/java/bin\n"`.

Fixing the problem

Before fixing the issue, let's learn a bit more about the PATH variable. This is an environment variable that contains an ordered list of paths that Linux will search for executables when running a command. Using these paths implies that we don't have to specify the absolute path for each command we want to run.

The PATH variable typically contains a few different paths, which are separated by colons. The goal of the Puppet rule that we saw was to add one specific directory to the list of paths, but unfortunately it's currently completely overwriting the contents. We need to change the Puppet rule to **append** the directory without overwriting the other paths in the variable.

To do this, we'll first include the current contents of the PATH variable followed by a colon, and then append the new contents we want to add.

Open the file init.pp using the nano editor:

```
sudo nano init.pp
```

Now, the content attribute should be changed. The line `content => "PATH=/java/bin\n"` should be changed to `content => "PATH=\$PATH:/java/bin\n"` since we want to append `/java/bin` to the environment variable `$PATH`, and not completely replace the current content in `$PATH`.

The extra backslash before the $ is necessary because Puppet also uses $ to indicate variables. But in this case, we want the dollar sign in the contents of the file.

On top of this, files in the `/etc/profile.d` directory should only be editable by the root user. In order to do this we'll need to change the mode of the file to avoid giving others permission to write the file. In other words, the mode should be 0644 not 0646.

Once you're done fixing the mode and content attributes, remember to save the file and close it.

After that, you can trigger a manual run of the Puppet agent by running the following command:

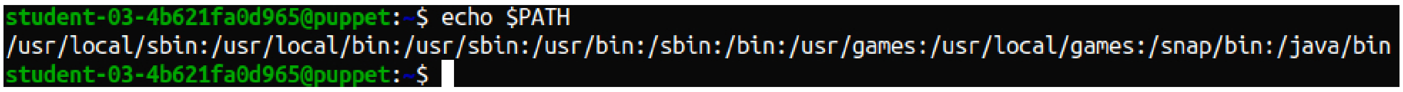
```
sudo puppet agent -v --test
```

Go ahead and initiate a **second SSH connection** following the same instructions from the `Accessing the linux virtual machine` section. Then, verify that the profile is fixed by checking the contents of the PATH variable.

To check the PATH variable has the right contents, use the command below in the second SSH connection:

```
echo $PATH
```

Output:



```
student-03-4b621fa0d965@puppet:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/java/bin
student-03-4b621fa0d965@puppet:~$
```

You've successfully fixed the misbehaving $PATH variable.