Red Hat Enterprise Linux Automation with Ansible

Lab: Managing Variables and Facts ☆

Performance Checklist

In this lab, you will write and run an Ansible Playbook that uses variables, secrets, and facts.

Outcomes

You should be able to define variables and use facts in a playbook, as well as use variables defined in an encrypted file.

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the `lab data-review start` command. The script creates the `/home/student/data-review` working directory and populates it with an Ansible configuration file and host inventory. The managed host `serverb.lab.example.com` is defined in this inventory as a member of the `webserver` host group. A developer has asked you to write an Ansible Playbook to automate the setup of a web server environment on `serverb.lab.example.com`, which controls user access to its website using basic authentication.

The `files` subdirectory contains:

A `httpd.conf` configuration file for the Apache web service for basic authentication

A `.htaccess` file, used to control access to the web server's document root directory

A `htpasswd` file containing credentials for permitted users

```
[student@workstation ~]$ lab data-review start
```

Procedure 3.4. Instructions

In the working directory, create the `playbook.yml` playbook and add the `webserver` host group as the managed host. Define the following play variables:

Table 3.5. Variables

Variable	Values
`firewall_pkg`	`firewalld`
`firewall_svc`	`firewalld`
`web_pkg`	`httpd`
`web_svc`	`httpd`
`ssl_pkg`	`mod_ssl`
`httpdconf_src`	`files/httpd.conf`
`httpdconf_dest`	`/etc/httpd/conf/httpd.conf`
`htaccess_src`	`files/.htaccess`
`secrets_dir`	`/etc/httpd/secrets`
`secrets_src`	`files/htpasswd`
`secrets_dest`	`"{{ secrets_dir }}/htpasswd"`
`web_root`	`/var/www/html`

1.1. Change to the `/home/student/data-review` working directory.

```
[student@workstation ~]$ cd ~/data-review
[student@workstation data-review]$
```

1.2. Create the `playbook.yml` playbook file and edit it in a text editor. The beginning of the file should appear as follows:

```
---
- name: install and configure webserver with basic auth
  hosts: webserver
  vars:
    firewall_pkg: firewalld
    firewall_svc: firewalld
    web_pkg: httpd
    web_svc: httpd
    ssl_pkg: mod_ssl
    httpdconf_src: files/httpd.conf
    httpdconf_dest: /etc/httpd/conf/httpd.conf
    htaccess_src: files/.htaccess
    secrets_dir: /etc/httpd/secrets
    secrets_src: files/htpasswd
    secrets_dest: "{{ secrets_dir }}/htpasswd"
    web_root: /var/www/html
```

HIDE SOLUTION

Add a `tasks` section to the play. Write a task that ensures the latest version of the necessary packages are installed. These packages are defined by the `firewall_pkg`, `web_pkg`, and `ssl_pkg` variables.

2.1. Define the beginning of the `tasks` section by adding the following line to the playbook:

```
  tasks:
```

2.2. Add the following lines to the playbook to define a task that uses the `yum` module to install the required packages.

```
    - name: latest version of necessary packages installed
      yum:
        name:
          - "{{ firewall_pkg }}"
          - "{{ web_pkg }}"
          - "{{ ssl_pkg }}"
        state: latest
```

HIDE SOLUTION

Add a second task to the playbook that ensures that the file specified by the `httpdconf_src` variable has been copied (with the `copy` module) to the location specified by the `httpdconf_dest` variable on the managed host. The file should be owned by the `root` user and the `root` group. Also set 0644 as the file permissions.

Add the following lines to the playbook to define a task that uses the `copy` module to copy the contents of the file defined by the `httpdconf_src` variable to the location specified by the `httpdconf_dest` variable.

```
  - name: configure web service
    copy:
      src: "{{ httpdconf_src }}"
      dest: "{{ httpdconf_dest }}"
      owner: root
      group: root
      mode: 0644
```

HIDE SOLUTION

Add a third task that uses the `file` module to create the directory specified by the `secrets_dir` variable on the managed host. This directory holds the password files used for the basic authentication of web services. The file should be owned by the `apache` user and the `apache` group. Set 0500 as the file permissions.

Add the following lines to the playbook to define a task that uses the `file` module to create the directory defined by the `secrets_dir` variable.

```
  - name: secrets directory exists
    file:
      path: "{{ secrets_dir }}"
      state: directory
      owner: apache
      group: apache
      mode: 0500
```

HIDE SOLUTION

Add a fourth task that uses the `copy` module to place a htpasswd file, used for basic authentication of web users. The source should be defined by the `secrets_src` variable. The destination should be defined by the `secrets_dest` variable. The file should be owned by the `apache` user and group. Set `0400` as the file permissions.

```
- name: htpasswd file exists
  copy:
    src: "{{ secrets_src }}"
    dest: "{{ secrets_dest }}"
    owner: apache
    group: apache
    mode: 0400
```

HIDE SOLUTION

Add a fifth task that uses the `copy` module to create a `.htaccess` file in the document root directory of the web server. Copy the file specified by the `htaccess_src` variable to `{{ web_root }}/.htaccess`. The file should be owned by the `apache` user and the `apache` group. Set 0400 as the file permissions.

Add the following lines to the playbook to define a task which uses the `copy` module to create the `.htaccess` file using the file defined by the `htaccess_src` variable.

```
- name: .htaccess file installed in docroot
  copy:
    src: "{{ htaccess_src }}"
    dest: "{{ web_root }}/.htaccess"
    owner: apache
    group: apache
    mode: 0400
```

HIDE SOLUTION

Add a sixth task that uses the `copy` module to create the web content file `index.html` in the directory specified by the `web_root` variable. The file should contain the message "*HOSTNAME* (*IPADDRESS*) has been customized by Ansible.", where `HOSTNAME` is the fully-qualified host name of the managed host and `IPADDRESS` is its IPv4 IP address. Use the `content` option to the `copy` module to specify the content of the file, and Ansible facts to specify the host name and IP address.

Add the following lines to the playbook to define a task that uses the `copy` module to create the `index.html` file in the directory defined by the `web_root` variable. Populate the file with the content specified using the `ansible_facts['fqdn']` and `ansible_facts['default_ipv4']['address']` Ansible facts retrieved from the managed host.

```
- name: create index.html
  copy:
    content: "{{ ansible_facts['fqdn'] }} ({{ ansible_facts['default_ipv4']['address'] }}) has been customized by Ansible.\n"
    dest: "{{ web_root }}/index.html"
```

HIDE SOLUTION

Add a seventh task that uses the `service` module to enable and start the firewall service on the managed host.

Add the following lines to the playbook to define a task that uses the `service` module to enable and start the firewall service.

```
- name: firewall service enabled and started
  service:
    name: "{{ firewall_svc }}"
    state: started
    enabled: true
```

HIDE SOLUTION

Add an eighth task that uses the `firewalld` module to allow the `https` service needed for users to access web services on the managed host. This firewall change should be permanent and should take place immediately.

Add the following lines to the playbook to define a task that uses the `firewalld` module to open the HTTPS port for the web service.

```
  - name: open the port for the web server
    firewalld:
      service: https
      state: enabled
      immediate: true
      permanent: true
```

HIDE SOLUTION

Add a final task that uses the `service` module to enable and start the web service on the managed host for all configuration changes to take effect. The name of the web service is defined by the `web_svc` variable.

```
  - name: web service enabled and started
    service:
      name: "{{ web_svc }}"
      state: started
      enabled: true
```

HIDE SOLUTION

Define a second play targeted at `localhost` which will test authentication to the web server. It does not need privilege escalation. Define a variable named `web_user` with the value `guest`.

11.1. Add the following line to define the start of a second play. Note that there is no indentation.

```
- name: test web server with basic auth
```

11.2. Add the following line to indicate that the play applies to the `localhost` managed host.

```
  hosts: localhost
```

11.3. Add the following line to disable privilege escalation.

```
  become: no
```

11.4. Add the following lines to define a variables list and the `web_user` variable.

```
  vars:
    web_user: guest
```

HIDE SOLUTION

Add a directive to the play that adds additional variables from a variable file named `vars/secret.yml`. This file contains a variable named `web_pass` that specifies the password for the web user. You will create this file later in the lab.

Define the start of the task list.

12.1. Using the `vars_files` keyword, add the following lines to the playbook to instruct Ansible to use variables found in the `vars/secret.yml` variable file.

```
  vars_files:
    - vars/secret.yml
```

12.2. Add the following line to define the beginning of the `tasks` list.

```
  tasks:
```

HIDE SOLUTION

Add two tasks to the second play.

The first uses the `uri` module to request content from `https://serverb.lab.example.com` using basic authentication. Use the `web_user` and `web_pass` variables to authenticate to the web server. Note that the certificate presented by `serverb` will not be trusted, so you will need to avoid certificate validation. The task should verify a return HTTP status code of `200`. Configure the task to place the returned content in the task results variable. Register the task result in a variable.

The second task uses the `debug` module to print the content returned from the web server.

13.1. Add the following lines to create the task for verifying the web service from the control node. Be sure to indent the first line with four spaces.

```
  - name: connect to web server with basic auth
    uri:
      url: https://serverb.lab.example.com
      validate_certs: no
      force_basic_auth: yes
      user: "{{ web_user }}"
      password: "{{ web_pass }}"
      return_content: yes
      status_code: 200
    register: auth_test
```

13.2. Create the second task using the debug module. The content returned from the web server is added to the registered variable as the key `content`.

```
  - debug:
      var: auth_test.content
```

13.3. The completed playbook should appear as follows:

```
---
- name: install and configure webserver with basic auth
  hosts: webserver
  vars:
    firewall_pkg: firewalld
    firewall_svc: firewalld
    web_pkg: httpd
    web_svc: httpd
    ssl_pkg: mod_ssl
    httpdconf_src: files/httpd.conf
    httpdconf_dest: /etc/httpd/conf/httpd.conf
    htaccess_src: files/.htaccess
    secrets_dir: /etc/httpd/secrets
    secrets_src: files/htpasswd
    secrets_dest: "{{ secrets_dir }}/htpasswd"
    web_root: /var/www/html
  tasks:
    - name: latest version of necessary packages installed
      yum:
        name:
          - "{{ firewall_pkg }}"
          - "{{ web_pkg }}"
          - "{{ ssl_pkg }}"
        state: latest

    - name: configure web service
      copy:
        src: "{{ httpdconf_src }}"
        dest: "{{ httpdconf_dest }}"
        owner: root
        group: root
        mode: 0644

    - name: secrets directory exists
      file:
        path: "{{ secrets_dir }}"
        state: directory
        owner: apache
        group: apache
        mode: 0500

    - name: htpasswd file exists
      copy:
        src: "{{ secrets_src }}"
        dest: "{{ secrets_dest }}"
        owner: apache
        group: apache
        mode: 0400

    - name: .htaccess file installed in docroot
      copy:
        src: "{{ htaccess_src }}"
        dest: "{{ web_root }}/.htaccess"
        owner: apache
        group: apache
        mode: 0400

    - name: create index.html
      copy:
        content: "{{ ansible_facts['fqdn'] }} ({{ ansible_facts['default_ipv4']['address'] }}) has been customized by Ansible.\n"
        dest: "{{ web_root }}/index.html"

    - name: firewall service enable and started
      service:
        name: "{{ firewall_svc }}"
        state: started
        enabled: true

    - name: open the port for the web server
      firewalld:
        service: https
        state: enabled
        immediate: true
        permanent: true

    - name: web service enabled and started
      service:
        name: "{{ web_svc }}"
        state: started
        enabled: true

- name: test web server with basic auth
  hosts: localhost
  become: no
```

```
  vars:
    - web_user: guest
  vars_files:
    - vars/secret.yml
  tasks:
    - name: connect to web server with basic auth
      uri:
        url: https://serverb.lab.example.com
        validate_certs: no
        force_basic_auth: yes
        user: "{{ web_user }}"
        password: "{{ web_pass }}"
        return_content: yes
        status_code: 200
      register: auth_test

    - debug:
        var: auth_test.content
```

13.4. Save and close the `playbook.yml` file.

HIDE SOLUTION

Create a file encrypted with Ansible Vault, named `vars/secret.yml`. Use the password `redhat` to encrypt it. It should set the `web_pass` variable to `redhat`, which will be the web user's password.

14.1. Create a subdirectory named `vars` in the working directory.

```
[student@workstation data-review]$ mkdir vars
```

14.2. Create the encrypted variable file, `vars/secret.yml`, using Ansible Vault. Set the password for the encrypted file to `redhat`.

```
[student@workstation data-review]$ ansible-vault create vars/secret.yml
New Vault password: redhat
Confirm New Vault password: redhat
```

14.3. Add the following variable definition to the file.

```
web_pass: redhat
```

14.4. Save and close the file.

HIDE SOLUTION

Run the `playbook.yml` playbook. Verify that content is successfully returned from the web server, and that it matches what was configured in an earlier task.

15.1. Before running the playbook, verify that its syntax is correct by running `ansible-playbook --syntax-check`. Use the `--ask-vault-pass` to be prompted for the vault password. Enter `redhat` when prompted for the password. If it reports any errors, correct them before moving to the next step. You should see output similar to the following:

```
[student@workstation data-review]$ ansible-playbook --syntax-check \
> --ask-vault-pass playbook.yml
Vault password: redhat

playbook: playbook.yml
```

15.2. Using the `ansible-playbook` command, run the playbook with the `--ask-vault-pass` option. Enter `redhat` when prompted for the password.

```
[student@workstation data-review]$ ansible-playbook playbook.yml --ask-vault-pass
Vault password: redhat
PLAY [Install and configure webserver with basic auth] ********************

...output omitted...

TASK [connect to web server with basic auth] **********************************
ok: [localhost]

TASK [debug] ******************************************************************
ok: [localhost] => {
    "auth_test.content": "serverb.lab.example.com (172.25.250.11) has been customized by Ansible.\n"
}

PLAY RECAP ********************************************************************
localhost                  : ok=3    changed=0    unreachable=0    failed=0
serverb.lab.example.com    : ok=10   changed=8    unreachable=0    failed=0
```

HIDE SOLUTION

Evaluation

Run the `lab data-review grade` command on *workstation* to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab data-review grade
```

Finish

On `workstation`, run the `lab data-review finish` command to clean up this exercise.

```
[student@workstation ~]$ lab data-review finish
```

This concludes the lab.

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH03S06)

→ NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH03S08)