Red Hat Enterprise Linux Automation with Ansible

Guided Exercise: Managing Users and Authentication ☆

In this exercise, you will create multiple users on your managed hosts and populate the authorized SSH keys for them.

Outcomes

You should be able to:

Create a new user group.

Manage users with the `user` module.

Populate SSH authorized keys using the `authorized_key` module.

Modify both the `sudoers` and the `sshd_config` files using the `lineinfile` module.

On `workstation`, run the lab start script to confirm the environment is ready for the lab to begin. The script creates the working directory, called `system-users`, and populates it with an Ansible configuration file, a host inventory, and some lab files.

```
[student@workstation ~]$ lab system-users start
```

Procedure 9.2. Instructions

Your organization requires that all hosts have the same local users available. These users should belong to the `webadmin` user group, which has the ability to use the `sudo` command without specifying a password. Also, the users' SSH public keys should be distributed in the environment and the `root` user should not be allowed to log in using SSH directly.

You are tasked with writing a playbook to ensure that the users and user group are present on the remote host. The playbook must ensure the users can log in using the authorized SSH key, as well as use `sudo` without specifying a password, and that the `root` user can't log in directly using SSH.

As the `student` user on `workstation`, change to the `/home/student/system-users` working directory.

```
[student@workstation ~]$ cd ~/system-users
[student@workstation system-users]$
```

Take a look at the existing `vars/users_vars.yml` variable file.

```
[student@workstation system-users]$ cat vars/users_vars.yml
---
users:
  - username: user1
    groups: webadmin
  - username: user2
    groups: webadmin
  - username: user3
    groups: webadmin
  - username: user4
    groups: webadmin
  - username: user5
    groups: webadmin
```

It uses the `username` variable name to set the correct username, and the `groups` variable to define additional groups that the user should belong to.

Start writing the `users.yml` playbook. Define a single play in the playbook that targets the `webservers` host group. Add a `vars_files` clause that defines the location of the `vars/users_vars.yml` filename, which has been created for you, and contains all the user names that are required for this exercise. Add the `tasks` clause to to the playbook.

Use a text editor to create the `users.yml` playbook. The playbook should contain the following:

```
---
- name: Create multiple local users
  hosts: webservers
  vars_files:
    - vars/users_vars.yml
  tasks:
```

Add two tasks to the playbook.

Use the `group` module in the first task to create the `webadmin` user group on the remote host. This task creates the `webadmin` group.

Use the `user` module in the second task to create the users from the `vars/users_vars.yml` file.

Execute the `users.yml` playbook.

4.1. Add the first task to the playbook. The first task contains the following:

```
  - name: Add webadmin group
    group:
      name: webadmin
      state: present
```

4.2. Add a second task to the playbook that uses the `user` module to create the users. Add a `loop: "{{ users }}"` clause to the task to loop through the variable file for every username found in the `vars/users_vars.yml` file. As the `name:` for the users, use the `item.username` as the variable name. This allows the variable file to contain additional information that might be useful for creating the users, such as the groups that the users should belong to. The second task contains the following:

```
  - name: Create user accounts
    user:
      name: "{{ item.username }}"
      groups: "{{ item.groups }}"
    loop: "{{ users }}"
```

4.3. Execute the playbook:

```
[student@workstation system-users]$ ansible-playbook users.yml

PLAY [Create multiple local users] *********************************************

TASK [Gathering Facts] *********************************************************
ok: [servera.lab.example.com]

TASK [Add webadmin group] ******************************************************
changed: [servera.lab.example.com]

TASK [Create user accounts] ****************************************************
changed: [servera.lab.example.com] => (item={u'username': u'user1', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user2', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user3', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user4', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user5', u'groups': u'webadmin'})

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=3    changed=2    unreachable=0    failed=0
```

Add a third task that uses the `authorized_key` module to ensure the SSH public keys have been properly distributed on the remote host. In the `files` directory, each of the users has a unique SSH public key file. The module loops through the list of users, finds the appropriate key by using the `username` variable, and pushes the key to the remote host.

The third task contains the following:

```
- name: Add authorized keys
  authorized_key:
    user: "{{ item.username }}"
    key: "{{ lookup('file', 'files/'+ item.username + '.key.pub') }}"
  loop: "{{ users }}"
```

Add a fourth task to the play that uses the `copy` module to modify the `sudo` config file and allow the `webadmin` group members to use `sudo` without a password on the remote host.

The fourth task appears as follows:

```
- name: Modify sudo config to allow webadmin users sudo without a password
  copy:
    content: "%webadmin ALL=(ALL) NOPASSWD: ALL"
    dest: /etc/sudoers.d/webadmin
    mode: 0440
```

Add a fifth task to ensure that the `root` user is not permitted to log in using SSH directly. Use `notify: "Restart sshd"` to trigger a handler to restart SSH.

The fifth task appears as follows:

```
- name: Disable root login via SSH
  lineinfile:
    dest: /etc/ssh/sshd_config
    regexp: "^PermitRootLogin"
    line: "PermitRootLogin no"
  notify: Restart sshd
```

In the first line after the location of the variable file, add a new handler definition. Give it a name of `Restart sshd`.

8.1. The handler should be defined as follows:

```
...output omitted...
    - vars/users_vars.yml
  handlers:
  - name: Restart sshd
    service:
      name: sshd
      state: restarted
```

The entire playbook now looks as follows:

```
---
- name: Create multiple local users
  hosts: webservers
  vars_files:
    - vars/users_vars.yml
  handlers:
  - name: Restart sshd
    service:
      name: sshd
      state: restarted

  tasks:

  - name: Add webadmin group
    group:
      name: webadmin
      state: present

  - name: Create user accounts
    user:
      name: "{{ item.username }}"
      groups: "{{ item.groups }}"
    loop: "{{ users }}"

  - name: Add authorized keys
    authorized_key:
      user: "{{ item.username }}"
      key: "{{ lookup('file', 'files/'+ item.username + '.key.pub') }}"
    loop: "{{ users }}"

  - name: Modify sudo config to allow webadmin users sudo without a password
    copy:
      content: "%webadmin ALL=(ALL) NOPASSWD: ALL"
      dest: /etc/sudoers.d/webadmin
      mode: 0440

  - name: Disable root login via SSH
    lineinfile:
      dest: /etc/ssh/sshd_config
      regexp: "^PermitRootLogin"
      line: "PermitRootLogin no"
    notify: "Restart sshd"
```

8.2. Execute the playbook.

```
[student@workstation system-users]$ ansible-playbook users.yml

PLAY [Create multiple local users] *********************************************

TASK [Gathering Facts] *********************************************************
ok: [servera.lab.example.com]

TASK [Add webadmin group] ******************************************************
ok: [servera.lab.example.com]

TASK [Create user accounts] ****************************************************
ok: [servera.lab.example.com] => (item={u'username': u'user1', u'groups': u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user2', u'groups': u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user3', u'groups': u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user4', u'groups': u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user5', u'groups': u'webadmin'})

TASK [Add authorized keys] *****************************************************
changed: [servera.lab.example.com] => (item={u'username': u'user1', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user2', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user3', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user4', u'groups': u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user5', u'groups': u'webadmin'})

TASK [Modify sudo config to allow webadmin users sudo without a password] ***
changed: [servera.lab.example.com]

TASK [Disable root login via SSH] **********************************************
changed: [servera.lab.example.com]

RUNNING HANDLER [Restart sshd] *************************************************
changed: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=7    changed=4    unreachable=0    failed=0
```

As the `user1` user, log in to `servera` server using SSH. Once logged in, use `sudo su -` command to switch identity to the `root` user.

9.1. Use SSH as the `user1` user and log in to `servera` server.

```
[student@workstation system-users]$ ssh user1@servera
Activate the web console with: systemctl enable --now cockpit.socket

[user1@servera ~]$
```

9.2. Switch identity to the `root` user.

```
[user1@servera ~]$ sudo -i
root@servera ~]#
```

9.3. Log out from `servera`.

```
[root@servera ~]$ exit
logout
[user1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation system-users]$
```

Try to log in to `servera` as the `root` user directly. This step should fail because the SSH daemon configuration has been modified not to permit direct `root` user logins.

10.1. From `workstation` use SSH as `root` to log in to `servera` server.

```
[student@workstation system-users]$ ssh root@servera
root@servera's password: redhat
Permission denied, please try again.
root@servera's password:
```

This confirms that the SSH configuration denied direct access to the system for the `root` user.

Finish

On `workstation`, run the `lab system-users finish` script to clean up the resources created in this exercise.

```
[student@workstation ~]$ lab system-users finish
```

This concludes the guided exercise.

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S03) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S05)

Privacy Policy (http://s.bl-1.com/h/cZrgWbQn?url=https://www.redhat.com/en/about/privacy-policy?extIdCarryOver=true&sc_cid=701f2000001D8QoAAK)

Terms of Use (https://www.redhat.com/en/about/terms-use)

Release Notes (https://learn.redhat.com/t5/Red-Hat-Learning-Subscription/Red-Hat-Learning-Subscription-Release-Notes/ba-p/22952)

Red Hat Training Policies (http://s.bl-1.com/h/cZrb2DXG?url=https://www.redhat.com/en/about/red-hat-training-policies)

All policies and guidelines (https://www.redhat.com/en/about/all-policies-guidelines)

Cookie Preferences