Managing Users in Ansible

Andrew Mallett
Linux Author and Trainer

@theurbanpenguin www.theurbanpenguin.com

Overview

Working with Users

- Creating Users
  - Passing variables at the CLI
  - Managing user passwords
- Managing Sudo Privileges
- Managing SSH Key Authentication

Users

Managing users accounts on your remote system can be useful to ensure a local managed account exists on the remote node. This could even be a dedicated account for Ansible

```
- name: 'Manage User Account'
  hosts: all
  become: true
  gather_facts: false
  tasks:
    - name 'Manage User'
      user:
        name: 'ansible'
        state: 'present'
        shell: '/bin/bash'
```

Simple User Playbook

To create a user account, we do not need to do a lot. We add the explicit shell for the user as the default will differ across Linux distributions. The account is created but, we have not set a password. We will manage passwords later.

```
- name: 'Manage User Account'
  hosts: all
  become: true
  gather_facts: false
  tasks:
    - name 'Manage User'
      user:
        name: "{{ user_account | default('ansible') }}"
        state: 'present'
        shell: '/bin/bash'

$ ansible-playbook -e user_account=joe user.yaml
```

Adding Variables

To add flexibility to the Play book we can use variables. Using the Jinja default filter we can also provide a default value if a value is not supplied.

Demo

Creating Users

- Static
- Passing variables

```
- name: 'Manage User Account'
  hosts: all
  become: true
  gather_facts: false
  tasks:
    - name 'Manage User'
      user:
        name: "{{ user_account | default('ansible') }}"
        state: 'present'
        shell: '/bin/bash'
      when: user_create == 'yes'

$ ansible-playbook -e user_account=joe -e user_create=yes user.yaml
```

Creating More Flexibility

Adding another variable allows us to control execution using the when meta-parameter

```
- name: 'Manage User Account'
  hosts: all
  become: true
  gather_facts: false
  tasks:
    - name 'Delete User'
      user:
        name: "{{ user_account | default('ansible') }}"
        state: 'absent'
        remove: true
      when: user_create == 'no'

$ ansible-playbook -e user_account=joe -e user_create=no user.yaml
```

Deleting Users

We can add another task now, the complete play will now be able to both create and delete users. When deleting users, we can ensure their home directory is removed with remove: true

Demo

Managing Users

- Adding a delete task
- Using verbose options to see details

```
- name: 'Manage User Account'
  hosts: all
  become: true
  gather_facts: false
  tasks:
    - name 'Manage User'
      user:
        name: "{{ user_account | default('ansible') }}"
        state: 'present'
        shell: '/bin/bash'
        password: "{{ 'Password1' | password_hash('sha512') }}"
        update_password: on_create
      when: user_create == 'yes'
```

Password Authentication

If we want the user to authenticate via a password this must be set. The Password value must be passed encrypted and so we use the filter password_hash to create a SHA512 password hash. We can't read the password's hashed value, so adding the update_password argument ensures we only set the password on user creation.

Demo

User Passwords

- Adding the password argument
- Adding update_password argument

```
- name: 'Manage Local User Account'
  hosts: rhel
  become: true
  gather_facts: false
  tasks:
    - name 'Manage User'
      user:
        name: "{{ user_account }}"
        state: 'present'
        generate_ssh_key: true
        ssh_key_file: '.ssh/id_ecdsa'
        ssh_key_type: 'ecdsa'
  $ ansible-playbook -e user_account=$USER localuser.yaml
```

SSH Key Pairs

When creating or managing user's we can also generate SSH Key Pairs. We can use the key for authenticating to remote systems. If we manage our own user account, we can add a key pair for our account that can be used for authentication. This needs only to be configured on the controller

Demo

Generating SSH Key Pairs

- For our local user account
- Creating an ecdsa key pair

```
- name: 'Allow SSH Authentication to Remote Account'
  authorized_key:
     user: "{{ user_account }}"
     state: 'present'
     manage_dir: true
     key: "{{ lookup('file', '/home/vagrant/.ssh/id_ecdsa.pub') }}"
```

SSH Authentication

If we want to be able to authenticate from our vagrant account to the newly create remote user account via SSH keys, we need to copy our public key to the remote user account and their authorized_keys file. This replaces the need of ssh-copy-id to each remote host.

Demo

Manage SSH Authentication

- authorized_keys module

```
-  tasks
       - name: 'name of block'
         block:
           - name: task1
           - name: task2
         when: user_create == 'yes'
```

Using Blocks, we can Simplify the Code

We can group tasks within a block adding a single when meta-parameter for the block rather than each task

```
tasks:
  - name: 'Create User Account, SSH Auth and Sudoers Entry'
    block:
      - name: 'Create Ansible User'
        user:
          name: 'ansible'
          state: 'present'
          shell: '/bin/bash'
          password: "{{ 'Password1' | password_hash('sha512') }}"
          update_password: 'on_create'
      - name: 'Allow SSH Authentication'
        authorized_key:
          user: 'ansible'
          state: 'present'
          manage_dir: true
          key: "{{ lookup('file', '/home/vagrant/.ssh/id_ecdsa.pub') }}"
      - name: 'Copy Sudoers file'
        copy:
          dest: '/etc/sudoers.d/ansible'
          content: 'ansible ALL=(root) NOPASSWD: ALL'
    when: user_create == 'yes'
```

Demo

Manage User Deployment

- Using blocks
- Consolidating tasks
- Multiple Plays
- Adding Sudo Rights

Summary

Managing Users

- user
  - password
  - generate_ssh_key (ssh-keygen)
- authorized_key (ssh-copy-id)
- blocks
- -e (extra variables)
- when

Ancillary Ansible Playbooks