Ansible Playbook to deploy user with SSH Key

May 19, 2021

We will create a new user with sudo privilege and deploy it with their SSH Public Key to all the targeted hosts through ansible-playbook. After that, he/she doesn't need to type an SSH password while login server.

I hope you have already installed ansible on your control host and you able to connect all your target hosts through ansible admin user. When a new System Administrator joins the company, manually creating his/her user account on multiple hosts is a tedious job. So to overcome this issue I have created a playbook which will create his/her user account, add the user to the admin privileged group and also copy their SSH public key to the remote servers.

Generate SSH Key

First generate SSH key for new user.

```
[newuser@srv-01 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/newuser/.ssh/id_rsa):
Created directory '/home/newuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/newuser/.ssh/id_rsa.
Your public key has been saved in /home/newuser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+Z/N055jsnrr2s/k7zg3IWA6SaURTmETeWFqC803vhM
The key's randomart image is:
+---[RSA 2048]----+
|       =+o.      |
|      +++.       |
|     .o=oo       |
|      oo+o.o     |
|       .SEo .    |
|       o +o  . . |
|      . .oo   .o.|
|       .  ...**=+|
|           oB=X@B|
+----[SHA256]-----+
```

Inventory File

This is an inventory file for our playbook. Here I have defined 2 groups **CentOS** and **Ubuntu** for our all hosts. To give admin privileges to a normal user, in Ubuntu users should be in **sudo** group and in CentOS users should be in **wheel** group, so I've defined variable **super_group** for this. I have also defined common variables for both groups.
To run the playbook we are using an ansible admin user "**ansible_admin**".

```
[CentOS]
srv-02
srv-03

[Ubuntu]
srv-04

[all:vars]
ansible_ssh_user = ansible_admin
ansible_ssh_pass = Pass123$

[CentOS:vars]
super_group = wheel

[Ubuntu:vars]
super_group = sudo
```

Check Connection

Now check whether remote hosts are reachable or not. If it's ok then you should get **SUCCESS** in output.

```
[root@srv-01]# ansible all -m ping -i inventory.ini
```

```
srv-02 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
srv-03 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
srv-04 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

All servers are reachable through ansible admin user.

Ansible Playbook

This is our playbook file.

```yaml
---
- name: Create New User
  hosts: all
  become: true
  gather_facts: false
  vars:
# Define your username and password here that you want to create on target hosts.
    username: newuser
    userpass: user_password
  tasks:
    - name: Create User
      ansible.builtin.user:
        name: "{{ username }}"
        state: present
        shell: /bin/bash
        password: "{{ userpass | password_hash('sha512') }}"
        update_password: on_create
        groups: "{{ super_group }}"
        append: yes

    - name: Deploy SSH Public Key
      ansible.posix.authorized_key:
        user: "{{ username }}"
        state: present
        key: "{{ lookup('file', '/home/{{ username }}/.ssh/id_rsa.pub') }}"

```

om/bidhanahdib/b1cb7a388fdf314897313fd436dd6b1a/raw/46c3c9f064472ec829a923cd085e82164699f487/create_user_playbook.yaml)
create_user_playbook.yaml (https://gist.github.com/bidhanahdib/b1cb7a388fdf314897313fd436dd6b1a#file-create_user_playbook-yaml) hosted with ❤ by GitHub (https://github.com)

YAML is very sensitive. We should be more cautions with the spaces while writing ansible playbook. Tabs are not allowed here. In this Playbook it contains a single play. The play has 2 tasks: Create user and Deploy SSH Public Key. We have used 2 modules for this `user` and `authorized`.

> *Replace your new user with their password in variable **username** and **userpass** of playbook.*

Explained here:
—: starting of playbook.
- name: Name of a Ansible playbook.
hosts: lists of hosts or host group against which we want to run the task.
become: Instructs the remote host to execute the playbook as admin.
gather_facts: We are not using any gather facts in this playbook so disabling it.
vars: definevariables- In our case, username and userpass.
tasks: list of tasks to be executed. Evvery task should have a name, which will show you in the output while

running the playbook and also its easy for us to identify it.
name: Name of the task.
ansible.builtin.user: Ansible user module. It manage user accounts and user attributes.
name: Name of the user to create, remove or modify. In our case we have defined variable.
state: Whether the account should exist or not.
password: Masked value of the password. Passing user password to get sha512 password hash.
update_password: It will only set the password for newly created users.
groups: A list of groups that the user will be added to.
append: This is used with the groups key and ensures that the group list is appended to.

- name: Name of 2nd task.
ansible.posix.authorized_key: Ansible authorized_key module. It adds or removes SSH authorized keys for particular user accounts.
user: The username on the remote host whose authorized_keys file will be modified.
state: Whether the given key should or should not be in the file.
key: User SSH Public Key.

> *NOTE: Variable names are defined inside double curly braces.*

Now run an ansible playbook. But first verify the playbook for any syntax errors.

```
[root@srv-01]# ansible-playbook create_user.yaml -i inventory.ini --syntax-check

playbook: create_user.yaml
```

If everything is ok and no error shown then it will display playbook name only.

Below command don’t make any changes; instead, try to predict some of the changes that may occur.

```
[root@srv-01]# ansible-playbook create_user.yaml -i inventory.ini --check
```

If you also want to further check which hosts will be affected by this playbook then execute below command.

```
[root@srv-01]# ansible-playbook create_user.yaml -i inventory.ini --list-hosts

playbook: create_user.yaml

  play #1 (all): Create New User        TAGS: []
    pattern: [u'all']
    hosts (3):
      srv-02
      srv-03
      srv-04
```

3 hosts will be affected from playbook.

Ansible Collections

You probably noticed that all Ansible module names that we used is composed of three parts, separated by a dot.
- ansible.builtin.user
- ansible.posix.authorized_key
This triplet is called a ***fully-qualified collection name (FQCN).*** The last part is the bare module name. The middle part is the collection name that the module lives in. The first part is a namespace that the collection belongs to. You can also define only last part of module name in your playbook.
The ***ansible.builtin*** collection that we used most of the time comes bundled with all Ansible versions from ***2.8*** forward. If you get error message saying ***"msg": "the connection plugin 'ansible.posix.authorized_key' was not found"*** then execute below command. Ansible comes bundled with the ***ansible-galaxy*** tool that we can use to install additional content.

```
[root@srv-01]# ansible-galaxy collection install ansible.posix
```

Run Playbook

To run the playbook, execute the below command. As I don't like the idea of giving ansible user a passwordless sudo access on target host that's why I am using an argument with capital ***-K*** which ask for a password. It lets you enter the ansible admin user password for the remote host and execute the playbook as admin.

> ***-k***, *–ask-pass: ask for connection password (We have already defined connection password through variable in inventory file.)* ***-K***, *–ask-become-pass: ask for privilege escalation password.*

```
[root@srv-01]# ansible-playbook create_user.yaml -i inventory.ini -K
```

```
[root@srv-01]# ansible-playbook create_user.yaml -i inventory.ini -K
BECOME password:

PLAY [Create New User] ****************************************************************************

TASK [Create User] ********************************************************************************
changed: [srv-02]
changed: [srv-03]
changed: [srv-04]

TASK [Deploy SSH Public Key] **********************************************************************
changed: [srv-02]
changed: [srv-03]
changed: [srv-04]

PLAY RECAP ****************************************************************************************
srv-02              : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
srv-03              : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
srv-04              : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Above output shows us summary of our playbook. 2 task are **OK** without any errors and **2 changes** has been done. If you execute above command again then nothing will be changed as ansible is an idempotent.

```
[root@srv-01]# ansible-playbook create_user.yaml -i inventory.ini -K
BECOME password:

PLAY [Create New User] ******************************************************************************

TASK [Create User] **********************************************************************************
ok: [srv-02]
ok: [srv-03]
ok: [srv-04]

TASK [Deploy SSH Public Key] ************************************************************************
ok: [srv-02]
ok: [srv-03]
ok: [srv-04]

PLAY RECAP ******************************************************************************************
srv-02              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
srv-03              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
srv-04              : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Now new user with sudo privileged is created in remote hosts. User’s SSH Public Key is added in the remote authorized_keys file.

Tags: Ansible

Categories: Automation

COMMENTS