Strukturen, Playbooks, Vaults - so habe ich es gemacht

Prolog

Ich möchte meine Erfahrungen mit Ansible mit euch teilen und werde hier mal aufführen wie ich Ansible auf meinem Lernweg aufbaue. Bedenkt aber stets, dass ich gerade erst mit Ansible anfange und vorher noch nie etwas mit zentralem Konfigurationsmanagement zu tun hatte. Ich zeige euch hier lediglich meine Überlegungen, die wahrscheinlich hier und da noch verbessert werden müssten. Ein Nachbauen macht ihr immer auf eigene Gefahr hin.

Vorraussetzungen - Ansible Eigenheiten

Was solltet ihr beachten bzw. vorausschauend planen, wenn ihr Ansible einsetzen wollt?

Benutzer für zentrale Verwaltungsaufgaben

Das Arbeiten mit Ansible wird euch viel einfacher fallen, wenn ihr einen einzigen Benutzer auf allen euren Hosts angelegt habt und dieser SUDO Rechte hat. Dann könnt ihr euer Ansible Management einheitlich aufbauen und habt nicht zig verschiedene User, die ihr berücksichtigen müsst.

> Ja, das ist sicherlich Geschmackssache und in manchen Situationen im Bezug auf Sicherheit sollte man das abwandeln und auf mehrere spezifische Benutzer zurückgreifen.

Ordnerstruktur

Ansible Docs:

https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html#directory-layout

Von Ansible wird keine Struktur vorgegeben. Man hat also die freie Entscheidungsgewalt. So habe ich also das hier angefangen zu strukturieren:

```
root@server: /etc/ansible# tree
.
├── ansible.cfg
├── archive
│   ├── azure.bak
│   ├── configure-linuxserver-rsyslog-to-graylog.yml.alt
│   ├── hosts.bak
│   └── install-prometheus-node-exporter.yml.bak
├── azure
├── group_vars
│   ├── production
│   │   ├── vars
│   │   └── vault
│   ├── init-linuxserver
│   │   ├── vars
│   │   └── vault
│   └── linuxserver
│       ├── vars
│       └── vault
├── playbooks
│   ├── packages
│   │   ├── install-production-prometheus-node-exporter.yml
│   │   ├── install-prometheus-node-exporter.yml
│   │   └── install-systemtools.yml
│   ├── services
│   │   └── connect-graylog.yml
│   ├── systemsettings
│   │   ├── set-timezone-europe-berlin.yml
```

```
|   |   └── update-hosts-file.yml
|   └── users
|       └── create-user-webdata.yml
└── vault_pwd
```

Variablen

So sieht eine **vars** Datei für die Gruppe **linuxserver** aus:

```
root@server:/etc/ansible# cat group_vars/linuxserver/vars
---
ansible_connection: ssh
ansible_user: ansibleadmin
ansible_ssh_private_key_file: /home/ansibleadmin/.ssh/id_rsa
ansible_ssh_common_args: '-o StrictHostKeyChecking=no'
ansible_become: yes
ansible_become_method: sudo
ansible_become_user: root
ansible_become_pass: "{{ vault_ansible_become_pass }}"
```

Inventory

Ansible Docs: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Meine Inventory Datei heißt **azure**, weil ich Ansible im Zuge einer Azure Evaluierung einsetze.

```
root@vm-azure-manager:/etc/ansible# cat azure
#Here i was experimenting where do set this option for being applied everywhere.
#ansible_ssh_common_args='-o StrictHostKeyChecking=no'

[prometheus]
prometheus-01

[grafana]
grafana-01

[graylog]
```

```
graylog-01

[es-nodes]
es-node-01

[init-linuxserver:children]
prometheus
grafana
graylog
es-nodes

[linuxserver:children]
prometheus
grafana
graylog
es-nodes

[graylogclients:children]
grafana
prometheus
es-nodes
```

Bisher habe ich folgendes gemeistert:

- Einstellungen in separate Unterverzeichnisse zu kapseln und Gruppen bezogen Einstellungen zu vergeben
- Hosts in Gruppen zu unterteilen
- Gruppen zu verschachteln

Durch die Verschachtelung von Gruppen kann ich meine Hosts kategorisieren und habe eine ähnliche Vorangehensweise wie beim Administrieren von Benutzerrechten, d.h. bei Änderungen den Aufwand zu minimieren, weil man z.B. einen neuen Host nur noch einer Gruppe zuweist oder einen alten aus einer Gruppe löschen muss.

Ansible Vault

Ansible Docs: https://docs.ansible.com/ansible/latest/user_guide/vault.html

Ich habe am Anfang die Passwörter für meine User, die Ansible benutzen sollte, klar lesbar in meine Konfig-Dateien geschrieben. Da ich davon aber absolut kein Freund bin, habe ich mich also

zeitnah darum gekümmert und herausgefunden, dass es die sog. Ansible-Vault gibt. Vault heißt Tresor. Also ein verschlüsselter Container, der eure Passwörter und sonstige Einstellungen beinhalten kann. Ich habe aktuell nur ein Passwort darin gespeichert.

Wie ihr schon anhand meiner Ordnerstruktur gesehen habt, gibt es jeweils in jedem Unterordner einer Gruppe eine Datei namens "vault". Das sind meine geschützten Bereiche für die jeweilige Gruppe.

So hier habe ich es gemacht:

1. Eine Schlüsseldatei "vault_pwd" erstellt und in Klartext den Schlüssel eingetragen, der global zum Ver- und Entschlüsseln benutzt werden soll.

   > Man muss keinen globalen Schlüssel verwenden. Das kann man so einstellen wie man es möchte.

   1. `root@server:# nano vault_pwd`
   2. Schlüssel eintragen

2. Globale Schlüsseldatei in der ansbile.cfg aktiviert
   1. `vault_password_file = /etc/ansible/vault_pwd`

3. zu verschlüsselnde Variablen mit einem Editor in die jeweilige vault Datei eintragen
4. vault Datei verschlüsseln: `ansible-vault encrypt vault`
   1. > Da wir einen globalen Schlüssel aktiviert haben, wird dieser automatisch zum Verschlüsseln benutzt, sodass ihr keinen Schlüssel an der Stelle eingeben müsst.
   2. So habe ich z.B. das Passwort hinterlegt, dass benutzt wird um SUDO Rechte zu aktivieren:
      1. `vault_ansible_become_pass: DAS-PASSWORT`
         1. Ansible empfiehlt als Variablennamen der zu verschlüsselnden Variable ein "vault_" als Präfix zu verwenden. Ich finde das gut, da man so gleich den Bezug der Verwendung herstellen kann.

5. Passwort in unverschlüsselter Variable aus verschlüsselter Variable aufrufen
   1. So sieht der Aufruf dann in meiner "vars" Datei aus: `ansible_become_pass: "{{ vault_ansible_become_pass }}"`
   2. Die Syntax nennt sich ***jinja2***

Alles was ich jetzt mit Ansible mache, funktioniert mit verschlüsselten Passwörtern.

Playbooks

Ansible Docs: https://docs.ansible.com/ansible/latest/user_guide/playbooks.html

Ansible-Systembenutzer bereitstellen

Funktionen

- User mit Home Verzeichnis anlegen
- SSH Key hinterlegen
- SUDO Rechte vergeben

Probleme

Man benötigt natürlich auf allen Hosts für das erstmalige Bereitstellen mit Ansible einen Systemuser mit SUDO Rechten. Da ich für Ansible Arbeiten einen eigenen Systemuser nutzen möchte, habe ich als erstes einen neuen user auf allen Hosts angelegt. Dafür habe ich den User benutzt, den man mit der Installation von Debian anlegen muss.

Playbook "create-user-ansibleadmin.yml

```
root@server:/etc/ansible# cat playbooks/users/create-user-ansibleadmin.yml
---
- hosts: init-linuxserver
  remote_user: initial-user-from-fresh-install-with-sudo-rights
  become: yes
  become_method: sudo
  become_user: root

  tasks:
    - name: Ansible create user ansibleadmin with home directory
      user:
       name: ansibleadmin
       createhome: yes

    - name: create directory .ssh
      file:
```

```
        path: /home/ansibleadmin/.ssh
        state: directory
        owner: ansibleadmin
        group: ansibleadmin
        mode: 0700

    - name: Ansible copy the ssh public key of the user ansibleadmin to the remote host
      copy:
        src: "/home/ansibleadmin/.ssh/id_rsa.pub"
        dest: "/home/ansibleadmin/.ssh/authorized_keys"
        owner: ansibleadmin
        group: ansibleadmin

    - name: Add user ansibleadmin to sudo group and grant sudo rights
      user:
        name: ansibleadmin
        groups: sudo
```

Debian Paket mit apt installieren

Playbook: "install-systemtools.yml"

```
root@server:/etc/ansible# cat playbooks/packages/install-systemtools.yml
---
- hosts: linuxserver
  tasks:
    - name: install lsb-release
      apt: name=lsb-release state=present

    - name: install lsof
      apt: name=lsof state=present
```

Playbook: Azure CLI installieren

```
---
- hosts: azure-master
  tasks:
    - name: Install apt-transport-https
      apt: name=apt-transport-https state=present
```

```
    - name: Add Azure Repository to sources.list.d
      apt_repository:
       repo: deb [arch=amd64] https://packages.microsoft.com/repos/azure-cli/ stretch
main
       state: present
       filename: azure-cli

    - name: Add Microsoft´s signature key
      apt_key:
       url: https://packages.microsoft.com/keys/microsoft.asc
       state: present

    - name: make an apt-update before be able to find azure-cli package
      apt:
       update_cache: yes

    - name: Install Azure CLI
      apt: name=azure-cli state=present
```

Playbook: Docker CE installieren

```
---
- hosts: docker-vm
  tasks:
    - name: Add Azure Repository to sources.list.d
      apt_repository:
       repo: deb [arch=amd64] https://download.docker.com/linux/debian stretch stable
       state: present
       filename: docker-ce

    - name: Add Docker CE signature key
      apt_key:
       url: https://download.docker.com/linux/debian/gpg
       state: present

    - name: make an apt-update before be able to find azure-cli package
      apt:
       update_cache: yes
```

```
    - name: Install Docker CE
      apt: name=docker-ce state=present
```

Datei kopieren - hosts Datei bereitstellen

Playbook "update-hosts-file.yml"

```
root@server:/etc/ansible# cat playbooks/systemsettings/update-hosts-file.yml
---
- hosts: linuxserver
# steht alles in den "Vars" Dateien
#  remote_user: ansibleadmin
#  become: yes
#  become_method: sudo
#  become_user: root

  tasks:
    - name: copy the hosts file from host "server" to the group linuxserver
      copy:
       src: "/etc/hosts"
       dest: "/etc/hosts"
       owner: root
       group: root
```

Hintergrund

Ich arbeite in meiner testumgebung mit den Hosts Dateien, weil ich auf die Schnelle kein DNS aufbauen konnte. Zum Einen bin ich kein Experte für DNS (erst einmal damit rumprobiert) und Zweitens wollte ich so schnell wie möglich alle Hosts mit Namen anstelle von IP Adressen ansprechen.

Zeitzone einstellen

Playbook "set-timezone-europe-berlin.yml"

```
root@server:/etc/ansible# cat playbooks/systemsettings/set-timezone-europe-berlin.yml
---
- hosts: linuxserver
```

```
  tasks:
    - name: set timezone to Europe/Berlin
      timezone:
       name: Europe/Berlin
```

Hintergrund

Da ich die Debian VMs in Azure über den RessourcenManager von Azure bereitstellen lassen habe, gab es nachträglich Einstellungen, die angepasst werden mussten. Dazu gehörte u.a. die Zeitzone.

Revision #12
Created 20 June 2018 08:33:41 by David
Updated 20 January 2020 08:57:40 by David