Automatisez le déploiement d'appareils virtuels avec VMware et Ansible

Publié: 2024-02-16

Ce guide explique comment créer des playbooks Ansible et des scripts python qui déploient des dispositifs virtuels sur un serveur VMware vSphere et configurent les hôtes ESX/ESXi pour mettre en miroir les données filaires.

Vous devez être familiarisé avec l'administration de VMware et la création de playbooks Ansible. En outre, ces procédures nécessitent les modules VMware Python et les outils de ligne de commande suivants :

PyVmomi 6.7.3

Pour les instructions d'installation, voir http://vmware.github.io/pyvmomi-community-samples/#getting-started.

Outil OVF 4.2.0

Pour les instructions d'installation, voir https://code.vmware.com/web/tool/4.3.0/ovf.

Avant de commencer

- Passez en revue les exigences relatives aux machines virtuelles pour le type d'appareil virtuel que vous déployez :
  - Exigences relatives aux machines virtuelles du capteur
  - Exigences relatives aux machines virtuelles ECA
  - Exigences relatives aux machines virtuelles EXA
  - Exigences relatives à la machine virtuelle de stockage des paquets ExtraHop
- (Recommandé) Effectuez une mise à niveau vers le dernier correctif pour l'environnement vSphere afin d'éviter tout problème connu.

Données Mirror Wire

Vous devez configurer les vSwitches sur les serveurs ESX/ESXi pour qu'ils reflètent les données filaires afin de permettre aux dispositifs virtuels ExtraHop de surveiller le trafic réseau.

Surveiller le trafic externe

Pour surveiller le trafic externe, vous devez créer un deuxième vSwitch sur une interface réseau. Cette interface se connecte à un miroir, un tap ou un agrégateur qui copie le trafic provenant d'un commutateur.

1. Créez des entrées pour les hôtes ESX/ESXi que vous souhaitez configurer dans votre fichier d'inventaire Ansible, comme dans l'exemple suivant :

```
esxis:
  hosts:
    esxi1.example.com:
      vswitch: vSwitch1
      port_group: "Remote Port Mirror"
      vmnic: vmnic1
  vars:
    vcenter: vcenter.example.com
    user: extrahop@vsphere.local
    password: !vault |
          $ANSIBLE_VAULT;1.1;AES256

 32313761623336326566303039613131373465663363326130336435326432666335333739643539
```

```
626638643636356365336330613765383963333466646630a643032303035376137363839656330

6339646532303966653135343766393434373566363830316631653431613463373962623138666 2

32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

Les exemples d'entrées d'inventaire ci-dessus définissent les variables de configuration suivantes :

esxi1.exemple.com

Nom de l'hôte ESX/ESXi sur lequel les dispositifs virtuels sont déployés.

vswitch

Nom du deuxième vSwitch à créer. Nous vous recommandons de nommer le vSwitch en ajoutant un numéro. Par exemple, si `vSwitch1` existe déjà sur votre hôte ESX/ESXi, spécifiez `vSwitch2`.

groupe_port

Nom du groupe de ports à créer sur le vSwitch.

vmnic

Nom de la carte réseau à laquelle ajouter le vSwitch.

vcenter

Le nom d'hôte du serveur vCenter.

utilisateur

Le nom d'un compte utilisateur sur le serveur vCenter.

mot de passe

Le mot de passe du compte utilisateur.

2. Créez une entrée de tâche dans un fichier playbook .yml, comme dans l'exemple suivant :

```
    - name: Mirror Wire Data
      eh_vsphere_external_mirror:
        esxi: "{{ inventory_hostname }}"
        vswitch: "{{ vswitch }}"
        port_group: "{{ port_group }}"
        vmnic: "{{ vmnic }}"
        vcenter: "{{ vcenter }}"
        user: "{{ user }}"
        password: "{{ password }}"
```

3. Ajoutez un script Python dans le répertoire de votre bibliothèque et ouvrez le fichier dans un éditeur de texte.

   Dans cet exemple, le script est nommé `/library/eh_vsphere_external_mirror.py`.

4. Importez tous les modules Python requis par le script.

   Le code suivant importe les modules requis pour le reste de la procédure :

```
from ansible.module_utils.basic import AnsibleModule
from pyVim import connect
from pyVmomi import vim
import ssl
```

5. Créez un objet AnsibleModule et importez des variables depuis le fichier .yml :

```
module = AnsibleModule(
    argument_spec = dict(
        esxi = dict(required=True),
        vswitch = dict(required=True),
```

```
        port_group = dict(required=True),
        vmnic = dict(required=True),
        vcenter = dict(required=True),
        user = dict(required=True),
        password = dict(required=True, no_log=True)
    ),
    supports_check_mode=False
)

esxi = module.params['esxi']
vswitch = module.params['vswitch']
port_group = module.params['port_group']
vmnic = module.params['vmnic']
vcenter = module.params['vcenter']
user = module.params['user']
password = module.params['password']
```

6. Établissez une connexion au serveur vCenter :

```
context = ssl.SSLContext(ssl.PROTOCOL_SSLv23)
try:
    si = connect.SmartConnect(host=vcenter,
                              user=user,
                              pwd=password,
                              sslContext=context)
except:
    module.fail_json(msg='Could not connect to vcenter.')
```

7. Récupérez un objet qui représente le serveur ESX/ESXi :

```
content = si.RetrieveContent()
object_view = content.viewManager.CreateContainerView(content.rootFolder,
 [], True)
for obj in object_view.view:
    if isinstance(obj, vim.HostSystem):
        if obj.name.lower() == esxi.lower():
            object_view.Destroy()
            host_system = obj
object_view.Destroy()
host_network_system = host_system.configManager.networkSystem
```

8. Configurez et ajoutez le nouveau commutateur virtuel :

```
switch_spec = vim.host.VirtualSwitch.Specification()
switch_spec.numPorts = 120
switch_spec.bridge = vim.host.VirtualSwitch.BondBridge(nicDevice=[vmnic])
host_network_system.AddVirtualSwitch(vswitchName=vswitch,
 spec=switch_spec)
```

9. Configurez le nouveau groupe de ports :

```
spec = vim.host.PortGroup.Specification()
spec.name = port_group
spec.vswitchName = vswitch
spec.vlanId = 4095
security_policy = vim.host.NetworkPolicy.SecurityPolicy()
security_policy.allowPromiscuous = True
security_policy.forgedTransmits = True
security_policy.macChanges = True
spec.policy = vim.host.NetworkPolicy(security=security_policy)
```

10. Ajoutez le groupe de ports :

```
host_network_system.AddPortGroup(portgrp=spec)
```

Surveillez le trafic intra-VM

Pour permettre à un dispositif virtuel de surveiller le trafic intra-VM, vous devez créer un groupe de ports supplémentaire sur le commutateur virtuel par défaut d'un hôte ESX/ESXi.

1. Créez des entrées pour les hôtes ESX/ESXi que vous souhaitez configurer dans votre fichier d'inventaire Ansible, comme dans l'exemple suivant :

```
esxis:
  hosts:
    esxi1.example.com:
      local_port_mirror: "Remote Port Mirror"
  vars:
    vcenter: vcenter.example.com
    user: extrahop@vsphere.local
    password: !vault |
          $ANSIBLE_VAULT;1.1;AES256

 32313761623336326566303039613131373465663363326130336435326432666335333739643539

 62663864363635636533633061376538396333346664666630a643032303035376137363839656330

 63396465323039666531353437663934343735663638303166316534316134633739626231386662

 32343630636365636650a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

Les exemples d'entrées d'inventaire ci-dessus définissent les variables de configuration suivantes :

esxi1.exemple.com
: Nom de l'hôte ESX/ESXi sur lequel les dispositifs virtuels sont déployés.

miroir port_local
: Nom du groupe de ports à créer sur le vSwitch.

vcenter
: Le nom d'hôte du serveur vCenter.

utilisateur
: Le nom d'un compte utilisateur sur le serveur vCenter.

mot de passe
: Le mot de passe du compte utilisateur.

2. Créez une entrée de tâche dans un fichier playbook .yml, comme dans l'exemple suivant :

```
    - name: Mirror Intra-VM Wire Data
      when: local_port_mirror is defined
      eh_vsphere_intra_mirror:
        esxi: "{{ inventory_hostname }}"
        local_port_mirror: "{{ local_port_mirror }}"
        vcenter: "{{ vcenter }}"
        user: "{{ user }}"
        password: "{{ password }}"
```

3. Ajoutez un script Python dans le répertoire de votre bibliothèque et ouvrez le fichier dans un éditeur de texte.

   Dans cet exemple, le script est nommé `/library/eh_vsphere_intra_mirror.py`.

4. Importez tous les modules Python requis par le script.
   Le code suivant importe les modules requis pour le reste de la procédure :

```
from ansible.module_utils.basic import AnsibleModule
from pyVim import connect
from pyVmomi import vim
import ssl
```

5. Créez un objet AnsibleModule et importez des variables depuis le fichier .yml :

```
module = AnsibleModule(
    argument_spec = dict(
        esxi = dict(required=True),
        local_port_mirror = dict(required=True),
        vcenter = dict(required=True),
        user = dict(required=True),
        password = dict(required=True, no_log=True)
    ),
    supports_check_mode=False
)

esxi = module.params['esxi']
local_port_mirror = module.params['local_port_mirror']
vcenter = module.params['vcenter']
user = module.params['user']
password = module.params['password']
```

6. Établissez une connexion au serveur vCenter :

```
context = ssl.SSLContext(ssl.PROTOCOL_SSLv23)
try:
    si = connect.SmartConnect(host=vcenter,
                              user=user,
                              pwd=password,
                              sslContext=context)
except:
    module.fail_json(msg='Could not connect to vcenter.')
```

7. Récupérez un objet qui représente le serveur ESX/ESXi :

```
content = si.RetrieveContent()
object_view = content.viewManager.CreateContainerView(content.rootFolder,
 [], True)
for obj in object_view.view:
    if isinstance(obj, vim.HostSystem):
        if obj.name.lower() == esxi.lower():
            object_view.Destroy()
            Host_sytem = obj
object_view.Destroy()
host_network_system = host_system.configManager.networkSystem
```

8. Configurez le nouveau groupe de ports :

```
spec = vim.host.PortGroup.Specification()
spec.name = local_port_mirror
spec.vswitchName = 'vSwitch0'
spec.vlanId = 4095
network_policy = vim.host.NetworkPolicy()
network_policy.security = vim.host.NetworkPolicy.SecurityPolicy()
network_policy.security.allowPromiscuous = True
security_policy.forgedTransmits = True
security_policy.macChanges = True
```

```
spec.policy = network_policy
```

9. Ajoutez le nouveau groupe de ports :

```
host_network_system.AddPortGroup(spec)
```

Déployez le fichier OVA

L'exemple suivant montre comment configurer un playbook et une bibliothèque de tâches pour déployer le fichier OVA pour un sonde VM. Le fichier OVA doit être téléchargé sur la machine sur laquelle Ansible est exécuté.

Vous pouvez télécharger les fichiers OVA ExtraHop à partir du Portail client ExtraHop.

1. Créez des entrées pour les appareils virtuels dans votre fichier d'inventaire Ansible, comme dans l'exemple suivant :

```
vms:
  hosts:
    exampleTestEDA:
      app_type: EDA
      datacenter: "Example Datacenter"
      folder: "VM Folder"
      esxi: esxi1.example.com
      datastore: DATA
      ova: ../firmware_images/extrahop-eda-1100v-vmware-7.9.0.2924.ova
      add_disk: 250
  vars:
    vcenter: vcenter.example.com
    user: extrahop@vsphere.local
    password: !vault |
          $ANSIBLE_VAULT;1.1;AES256

 32313761623336326566303039613131373465663363326130336435326432666335333739643539

 62663864363635636533633061376538396333346664666630a643032303035376137363839656330

 63396465323039666531353437663934343735663638303166316534316134633739626231386662

 32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

Les entrées d'inventaire ci-dessus définissent les variables de configuration suivantes :

Exemple EDA
: Le nom de la machine virtuelle.

type_application
: Type d'appareil virtuel.

datacenter
: Le centre de données qui contient l'hôte ESX/ESXi.

dossier
: Le dossier des machines virtuelles du centre de données.

esxi
: Nom de l'hôte ESX/ESXi sur lequel les machines virtuelles sont déployées.

banque de données
: Nom de la banque de données qui contiendra la machine virtuelle.

ovules

Le chemin relatif du fichier OVA ExtraHop.

vcenter

Le nom d'hôte du serveur vCenter.

utilisateur

Le nom d'un compte utilisateur sur le serveur vCenter.

mot de passe

Le mot de passe du compte utilisateur.

2. Créez une entrée de tâche dans un playbook .yml, comme dans l'exemple suivant :

```
    - name: Deploy ExtraHop OVA to vSphere
      eh_vsphere_deploy:
        appliance: "{{ inventory_hostname }}"
        app_type: "{{ app_type }}"
        datacenter: "{{ datacenter }}"
        folder: "{{ folder }}"
        esxi: "{{ esxi }}"
        datastore: "{{ datastore }}"
        ova: "{{ ova }}"
        vcenter: "{{ vcenter }}"
        user: "{{ user }}"
        password: "{{ password }}"
```

3. Ajoutez un script Python dans le répertoire de votre bibliothèque et ouvrez le fichier dans un éditeur de texte.

   Dans cet exemple, le script est nommé `/library/eh_vsphere_deploy.py`.

4. Importez tous les modules Python requis par le script.

   Le code suivant importe les modules requis pour le reste de la procédure :

```
from ansible.module_utils.basic import AnsibleModule
import subprocess
import urllib
```

5. Créez un objet AnsibleModule et importez des variables depuis le fichier .yml :

```
module = AnsibleModule(
    argument_spec = dict(
        appliance = dict(required=True),
        app_type = dict(required=True),
        datacenter = dict(required=True),
        folder = dict(required=True),
        esxi = dict(required=True),
        ova = dict(required=True),
        datastore = dict(required=True),
        vcenter = dict(required=True),
        user = dict(required=True),
        password = dict(required=True, no_log=True)
    ),
    supports_check_mode=False
)

appliance = module.params['appliance']
app_type = module.params['app_type']
datastore = module.params['datastore']
ova = module.params['ova']
attrs = {
    'vcenter': module.params['vcenter'],
    'user': module.params['user'],
```

```
    'password': module.params['password'],
    'datacenter': module.params['datacenter'],
    'folder': module.params['folder'],
    'esxi': module.params['esxi']
}
```

6. Créez la chaîne d'hôte qui spécifie l'emplacement de l'hôte ESX/ESXi dans vCenter :

```
           attrs = dict((k, urllib.quote(v)) for k, v in attrs.items())
host_str = 'vi://{user}:{password}@{vcenter}/{datacenter}/host/{folder}/
{esxi}'.format(**attrs)
```

Pour plus d'informations sur la création d'une chaîne hôte, consultez la documentation de l'outil OVF sur le site de VMware : https://vdc-download.vmware.com/vmwb-repository/dcr-public/bb505ca7-88b5-4b11-aff4-f59125ab27bc/f3d05149-23e9-4ac2-8f99-0c851a8a5231/ovftool-430-userguide.pdf

7. Créez un tableau pour définir le `ovtool` commande :

```
cmd = ['ovftool',
       '--disableVerification',
       '--noSSLVerify',
       '-ds=%s' % datastore,
       '-dm=%s' % 'thick',
       '-n=%s' % eda,
       '--net:VM Network=VM Network',
       '--X:logFile=ovflogs.log',
       '--X:logLevel=verbose',
       ova,
       host_str]

if app_type == 'EDA' or app_type == 'ETA':
    cmd.insert(7, '--net:Remote Port Mirror=Remote Port Mirror')
```

Note: Ce code définit deux mappages de réseau pour les capteurs et les magasins de paquets. `VM Network` correspond au réseau de gestion des machines virtuelles sur l'hôte ESX/ESXi. `Remote Port Mirror` correspond au réseau qui reflète les données filaires. Les noms des réseaux cibles doivent correspondre à ceux de votre hôte ESX/ESXi. Par exemple, si vous avez configuré un réseau nommé `Local Port Mirror` pour refléter le trafic intra-VM, spécifiez :

```
--net:Remote Port Mirror=Local Port Mirror
```

Pour les appliances EDA 6100v, vous pouvez ajouter des mappages réseau supplémentaires à la baie ci-dessus pour Remote Port Mirror 2 et Remote Port Mirror 3, par exemple :

```
--net:Remote Port Mirror2=Remote Port Mirror2
```

8. Exécutez le `ovftool` commande avec le `subprocess.Popen` classe :

```
proc = subprocess.Popen(cmd)
proc.communicate()
if proc.returncode != 0:
    module.fail_json(msg='Unable to deploy OVA')
else:
    module.exit_json(changed=True, msg="Deployed OVA")
```

Ajouter un disque dans VMware (facultatif)

Pour capteurs disposant d'une licence pour la capture de paquets et pour tous les EXA, vous devez configurer un disque supplémentaire pour le dispositif virtuel.

1. Créez des entrées pour les appareils virtuels dans votre fichier d'inventaire Ansible, comme dans l'exemple suivant :

```
vms:
  hosts:
   exampleEXA:
      add_disk: 150
      app_type: EXA
      datacenter: "Example Datacenter"
      folder: "VM Folder"
      esxi: esxi1.example.com
      ova: ../firmware_images/extrahop-exa-5100v-xs-vmware-7.9.0.2924.ova
  vars:
    vcenter: vcenter.example.com
    user: extrahop@vsphere.local
    password: !vault |
          $ANSIBLE_VAULT;1.1;AES256

 32313761623336326566303039613131373465663363326130336435326432666335333739643539

 62663864363635636533633061376538396333346664666630a643032303035376137363839656330

 63396465323039666531353437663934343735663638303166316534316134633739626231386662

 3234363063636563650a326336343334393638336433303063623636376231643934323961393338
          6133
```

Les entrées d'inventaire ci-dessus définissent les variables de configuration suivantes :

Exemple EXA

Le nom de la machine virtuelle.

ajouter_disque

Taille en Go du disque à ajouter. Les valeurs de taille suivantes sont valides :

Appliance virtuelle	Taille
EDA 1100 V	250
EDA 6100 V	500
EXA-XS	250 ou moins
EXA-S	500 ou moins
EXA-M	1000 ou moins
EXA-L	2000 ou moins

type_application

Type d'appareil virtuel.

datacenter

Le centre de données qui contient l'hôte ESX/ESXi.

dossier

Le dossier des machines virtuelles du centre de données.

esxi

Nom de l'hôte ESX/ESXi sur lequel la machine virtuelle est déployée.

banque de données

Nom de la banque de données qui contiendra la machine virtuelle.

ovules

Le chemin relatif du fichier OVA ExtraHop.

vcenter

Le nom d'hôte du serveur vCenter.

utilisateur

Le nom d'un compte utilisateur sur le serveur vCenter.

mot de passe

Le mot de passe du compte utilisateur.

2. Créez une entrée de tâche dans un playbook .yml, comme dans l'exemple suivant :

```
    - name: Add a disk
      when: add_disk is defined
      eh_vsphere_add_disk:
        vm_name: "{{ inventory_hostname }}"
        add_disk: "{{ add_disk }}"
        vcenter: "{{ vcenter }}"
        user: "{{ user }}"
        password: "{{ password }}"
```

3. Ajoutez un script Python dans le répertoire de votre bibliothèque et ouvrez le fichier dans un éditeur de texte.

   Dans cet exemple, le script est nommé `/library/eh_vsphere_add_disk.py`.

4. Importez tous les modules Python requis par le script.

   Le code suivant importe les modules requis pour le reste de la procédure :

```
from ansible.module_utils.basic import AnsibleModule
from pyVim import connect
from pyVmomi import vim
import ssl
```

5. Créez un objet AnsibleModule et importez des variables depuis le fichier .yml :

```
module = AnsibleModule(
    argument_spec = dict(
        vm_name = dict(required=True),
        add_disk = dict(required=True),
        vcenter = dict(required=True),
        user = dict(required=True),
        password = dict(required=True, no_log=True)
    ),
    supports_check_mode=False
)

vm_name = module.params['vm_name']
add_disk = module.params['add_disk']
vcenter = module.params['vcenter']
user = module.params['user']
password = module.params['password']
```

6. Traduisez la taille du disque PCAP de Go en Ko :

```

```

```
            add_disk = int(add_disk) * 1024 * 1024
```

7. Établissez une connexion au serveur vCenter :

```
context = ssl.SSLContext(ssl.PROTOCOL_SSLv23)
try:
    si = connect.SmartConnect(host=vcenter,
                              user=user,
                              pwd=password,
                              sslContext=context)
except:
    module.fail_json(msg='Could not connect to vcenter.')
```

8. Récupérez un objet qui représente la machine virtuelle à laquelle vous ajoutez le disque.

```
content = si.RetrieveContent()
object_view = content.viewManager.CreateContainerView(content.rootFolder,
                                                       [], True)
for obj in object_view.view:
    if isinstance(obj, vim.VirtualMachine):
        if obj.name.lower() == vm_name.lower():
            object_view.Destroy()
            return obj
object_view.Destroy()
```

9. Trouvez l'emplacement où ajouter le disque virtuel en parcourant tous les périphériques de la machine virtuelle. Le numéro d'unité du disque doit être le numéro d'unité du dernier disque virtuel de l'équipement plus un. Récupérez également l'objet qui représente le contrôleur iSCSI de la machine virtuelle :

```
for dev in vm.config.hardware.device:
    if isinstance(dev, vim.vm.device.VirtualDisk):
        unit_number = int(dev.unitNumber) + 1
        if unit_number == 7:
            unit_number += 1
        if unit_number >= 16:
            module.fail_json(msg="Number of disks not supported")
    if isinstance(dev, vim.vm.device.VirtualSCSIController):
        controller = dev
```

10. Configurez les spécifications du disque :

```
dev_changes = []
diskSpec = vim.vm.device.VirtualDeviceSpec()
diskSpec.fileOperation = "create"
diskSpec.operation = vim.vm.device.VirtualDeviceSpec.Operation.add
diskSpec.device = vim.vm.device.VirtualDisk()
diskSpec.device.backing = vim.vm.device.VirtualDisk.FlatVer2BackingInfo()
diskSpec.device.backing.thinProvisioned = False
diskSpec.device.backing.diskMode = 'persistent'
diskSpec.device.unitNumber = unit_number
diskSpec.device.capacityInKB = add_disk
diskSpec.device.controllerKey = controller.key
dev_changes.append(diskSpec)
vm_spec = vim.vm.ConfigSpec()
vm_spec.deviceChange = dev_changes
```

11. Ajoutez le disque :

```
vm.ReconfigVM_Task(spec=vm_spec)
```

Augmenter le disque de stockage des paquets sur un magasin de paquets ExtraHop (facultatif)

L'exemple suivant montre comment augmenter la capacité du disque de stockage des paquets sur une appliance ETA 1150v.

Note: L'exemple ne s'applique pas aux appareils ETA 6150, qui nécessitent un disque de stockage des paquets compris entre 1 To et 25 To.

1. Créez des entrées pour les appareils virtuels dans votre fichier d'inventaire Ansible, comme dans l'exemple suivant :

```
vms:
  hosts:
    exampleETA:
      app_type: ETA
      datacenter: "Example Datacenter"
      folder: "VM Folder"
      esxi: esxi1.example.com
      datastore: DATA
      ova: ../firmware_images/extrahop-eta-1150v-vmware-7.9.0.2924.ova
      increase_disk: 2000
  vars:
    vcenter: vcenter.example.com
    user: extrahop@vsphere.local
    password: !vault |
          $ANSIBLE_VAULT;1.1;AES256

 32313761623336326566303039613131373465663363326130336435326432666335333739643539

 62663864363635636533633061376538396333346664666630a643032303035376137363839656330

 63396465323039666531353437663934343735663638303166316534316134633739626231386662

 32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

2. Créez une entrée de tâche dans un fichier playbook .yml, comme dans l'exemple suivant :

```
   - name: Increase an ETA packetstore
     when: increase_disk is defined
     eh_vsphere_increase_disk:
       vm_name: "{{ inventory_hostname }}"
       increase_disk: "{{ increase_disk }}"
       datacenter: "{{ datacenter }}"
       vcenter: "{{ vcenter }}"
       user: "{{ user }}"
       password: "{{ password }}"
```

3. Ajoutez un script Python dans le répertoire de votre bibliothèque et ouvrez le fichier dans un éditeur de texte.

   Dans cet exemple, le script est nommé `library/eh_vsphere_increase_disk.py`.

4. Importez tous les modules Python requis par le script.

   Le code suivant importe les modules requis pour le reste de la procédure :

```
from ansible.module_utils.basic import AnsibleModule
from pyVim import connect
from pyVmomi import vim
import ssl
```

5. Créez un objet AnsibleModule et importez des variables depuis le fichier .yml :

```
module = AnsibleModule(
    argument_spec = dict(
        vm_name = dict(required=True),
        increase_disk = dict(required=True),
        datacenter = dict(required=True),
        vcenter = dict(required=True),
        user = dict(required=True),
        password = dict(required=True, no_log=True)
    ),
    supports_check_mode=False
)

vm_name = module.params['vm_name']
increase_disk = module.params['increase_disk']
datacenter = module.params['datacenter']
vcenter = module.params['vcenter']
user = module.params['user']
password = module.params['password']
```

6. Traduisez la taille du disque de Go en Ko :

```
increase_disk = int(increase_disk) * 1024 * 1024
```

7. Établissez une connexion avec le serveur vCenter :

```
context = ssl.SSLContext(ssl.PROTOCOL_SSLv23)
try:
    si = connect.SmartConnect(host=vcenter,
                              user=user,
                              pwd=password,
                              sslContext=context)
except:
    module.fail_json(msg='Could not connect to vcenter.')
```

8. Récupérez un objet qui représente la machine virtuelle de l'ETA :

```
content = si.RetrieveContent()
object_view = content.viewManager.CreateContainerView(content.rootFolder,
                                                      [], True)
for obj in object_view.view:
    if isinstance(obj, vim.VirtualMachine):
        if obj.name.lower() == vm_name.lower():
            object_view.Destroy()
            vm = obj

object_view.Destroy()
```

9. Récupérez le chemin du fichier qui sauvegarde le disque que vous souhaitez redimensionner :

```
for dev in vm.config.hardware.device:
    if isinstance(dev, vim.vm.device.VirtualDisk) and
 dev.deviceInfo.label == 'Hard disk 2':
        disk = dev
path = disk.backing.fileName
```

10. Récupérez un objet qui représente le centre de données dans lequel se trouve la machine virtuelle :

```
datacenter_list = si.content.rootFolder.childEntity
for d in datacenter_list:
    if d.name == dcName:
```

```
        datacenter_obj = d
```

11. Augmentez la taille du disque :

```
virtualDiskManager = si.content.virtualDiskManager
virtualDiskManager.ExtendVirtualDisk(path, datacenter_obj, increase_disk,
 False)
```