Automatisieren Sie die Bereitstellung virtueller Appliances mit VMware und Ansible

Veröffentlicht: 2023-09-14

In diesem Handbuch wird erklärt, wie Sie Ansible-Playbooks und Python-Skripte erstellen, die virtuelle Appliances auf einem VMware vSphere-Server bereitstellen und ESX/ESXi-Hosts für die Spiegelung von Wire-Daten konfigurieren.

Sie müssen mit der Verwaltung von VMware und der Erstellung von Ansible-Playbooks vertraut sein. Darüber hinaus erfordern diese Verfahren die folgenden VMware Python-Module und Befehlszeilentools:

PyvMomi 6.7.3

Installationsanweisungen finden Sie unter http://vmware.github.io/pyvmomi-community-samples/#getting-started.

OVF-Werkzeug 4.2.0

Installationsanweisungen finden Sie unter https://code.vmware.com/web/tool/4.3.0/ovf.

Bevor du anfängst

- Überprüfen Sie die VM-Anforderungen für den Typ der virtuellen Appliance, die Sie bereitstellen:
  - Anforderungen an die Sensor-VM
  - ECA-VM-Anforderungen
  - EXA-VM-Anforderungen
  - ExtraHop Packetstore-VM-Anforderungen
- (Empfohlen) Führen Sie ein Upgrade auf den neuesten Patch für die vSphere-Umgebung durch, um bekannte Probleme zu vermeiden.

Daten zu Spiegeldrähten

Sie müssen vSwitches auf ESX/ESXi-Servern so konfigurieren, dass sie wire data spiegeln, damit virtuelle ExtraHop-Appliances den Netzwerkverkehr überwachen können.

Überwachen Sie den externen Verkehr

Um den externen Verkehr zu überwachen, müssen Sie einen zweiten vSwitch auf einer Netzwerkschnittstelle erstellen. Diese Schnittstelle stellt eine Verbindung zu einem Mirror, Tap oder Aggregator her, der den Datenverkehr von einem Switch kopiert.

1. Erstellen Sie Einträge für die ESX/ESXi-Hosts, die Sie konfigurieren möchten, in Ihrer Ansible-Inventardatei, ähnlich dem folgenden Beispiel:

```
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

 32313761623336326566303039613131373465663633261303364353264326663353337396435 39
```

```
62663864363635636533633061376538396333346664666630a643032303035376137363839656330

6339646532303966653135343766393434373566363830316631653431613463373962623138666 2

32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

Die obigen Beispielinventareinträge definieren die folgenden Konfigurationsvariablen:

esxi1.example.com

Der Name des ESX/ESXi-Hosts, auf dem die virtuellen Appliances bereitgestellt werden .

vswitch

Der Name des zweiten zu erstellenden vSwitches. Wir empfehlen, dass Sie den vSwitch benennen, indem Sie eine Zahl anhängen. Zum Beispiel, wenn `vSwitch1` ist bereits auf Ihrem ESX/ESXi-Host vorhanden, geben Sie an `vSwitch2`.

Portgruppe

Der Name der Portgruppe, die auf dem vSwitch erstellt werden soll.

vmnic

Der Name der NIC, zu der der vSwitch hinzugefügt werden soll.

vcenter

Der Hostname des vCenter Server.

Benutzer

Der Name eines Benutzerkonto auf dem vCenter Server.

Passwort

Das Passwort des Benutzerkonto.

2. Erstellen Sie einen Aufgabeneintrag in einer YML-Playbook-Datei, ähnlich dem folgenden Beispiel:

```
    - 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. Fügen Sie Ihrem Bibliotheksverzeichnis ein Python-Skript hinzu und öffnen Sie die Datei in einem Texteditor.

   In diesem Beispiel heißt das Skript `/library/eh_vsphere_external_mirror.py`.

4. Importieren Sie alle Python-Module, die das Skript benötigt.

   Der folgende Code importiert die Module, die für den Rest des Verfahrens erforderlich sind:

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

5. Erstellen Sie ein AnsibleModule-Objekt und importieren Sie Variablen aus der .yml-Datei:

```
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. Stellen Sie eine Verbindung zum vCenter Server her:

```
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. Rufen Sie ein Objekt ab, das den ESX/ESXi-Server darstellt:

```
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. Konfigurieren Sie den neuen virtuellen Switch und fügen Sie ihn hinzu:

```
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. Konfigurieren Sie die neue Portgruppe:

```
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. Fügen Sie die Portgruppe hinzu:

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

Überwachen Sie den internen VM-Verkehr

Damit eine virtuelle Appliance den internen VM-Verkehr überwachen kann, müssen Sie eine zusätzliche Portgruppe auf dem virtuellen Standard-Switch eines ESX/ESXi-Hosts erstellen.

1. Erstellen Sie Einträge für die ESX/ESXi-Hosts, die Sie konfigurieren möchten, in Ihrer Ansible-Inventardatei, ähnlich dem folgenden Beispiel:

```
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
```

Die obigen Beispielinventareinträge definieren die folgenden Konfigurationsvariablen:

esxi1.example.com
: Der Name des ESX/ESXi-Hosts, auf dem die virtuellen Appliances bereitgestellt werden .

local_port_mirror
: Der Name der Portgruppe, die auf dem vSwitch erstellt werden soll.

vcenter
: Der Hostname des vCenter Server.

Benutzer
: Der Name eines Benutzerkonto auf dem vCenter Server.

Passwort
: Das Passwort des Benutzerkonto.

2. Erstellen Sie einen Aufgabeneintrag in einer YML-Playbook-Datei, ähnlich dem folgenden Beispiel:

```
    - 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. Fügen Sie Ihrem Bibliotheksverzeichnis ein Python-Skript hinzu und öffnen Sie die Datei in einem Texteditor.

   In diesem Beispiel heißt das Skript `/library/eh_vsphere_intra_mirror.py`.

4. Importieren Sie alle Python-Module, die das Skript benötigt.
   Der folgende Code importiert die Module, die für den Rest des Verfahrens erforderlich sind:

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

5. Erstellen Sie ein AnsibleModule-Objekt und importieren Sie Variablen aus der .yml-Datei:

```
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. Stellen Sie eine Verbindung zum vCenter Server her:

```
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. Rufen Sie ein Objekt ab, das den ESX/ESXi-Server darstellt:

```
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. Konfigurieren Sie die neue Portgruppe:

```
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. Fügen Sie die neue Portgruppe hinzu:

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

Stellen Sie die OVA-Datei bereit

Das folgende Beispiel zeigt, wie Sie ein Playbook und eine Aufgabenbibliothek einrichten, um die OVA-Datei für ein bereitzustellen Sensor VM. Die OVA-Datei muss auf den Computer heruntergeladen werden, auf dem Ansible ausgeführt wird.

Sie können ExtraHop OVA-Dateien von der ExtraHop Kundenportal.

1. Erstellen Sie Einträge für virtuelle Appliances in Ihrer Ansible-Inventardatei, ähnlich dem folgenden Beispiel:

```
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
```

Die obigen Inventareinträge definieren die folgenden Konfigurationsvariablen:

Beispiel EDA
: Der Name der VM.

App-Typ
: Der Typ der virtuellen Appliance.

Rechenzentrum
: Das Rechenzentrum, das den ESX/ESXi-Host enthält.

Mappe
: Der Ordner mit den VMs im Rechenzentrum.

esxi
: Der Name des ESX/ESXi-Hosts, auf dem die VM bereitgestellt wird.

Datenspeicher
: Der Name des Datenspeichers, der die VM enthalten wird.

Eizellen

Der relative Pfad der ExtraHop-OVA-Datei.

vcenter

Der Hostname des vCenter Server.

Benutzer

Der Name eines Benutzerkonto auf dem vCenter Server.

Passwort

Das Passwort des Benutzerkonto.

2. Erstellen Sie einen Aufgabeneintrag in einem .yml-Playbook, ähnlich dem folgenden Beispiel:

```
    - 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. Fügen Sie Ihrem Bibliotheksverzeichnis ein Python-Skript hinzu und öffnen Sie die Datei in einem Texteditor.

   In diesem Beispiel heißt das Skript `/library/eh_vsphere_deploy.py`.

4. Importieren Sie alle Python-Module, die das Skript benötigt.

   Der folgende Code importiert die Module, die für den Rest des Verfahrens erforderlich sind:

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

5. Erstellen Sie ein AnsibleModule-Objekt und importieren Sie Variablen aus der .yml-Datei:

```
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. Erstellen Sie die Hostzeichenfolge, die den Speicherort des ESX/ESXi-Hosts in vCenter angibt:

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

Weitere Informationen zum Erstellen einer Hostzeichenfolge finden Sie in der OVF-Tool-Dokumentation auf der VMware-Website: https://vdc-download.vmware.com/vmwb-repository/dcr-public/bb505ca7-88b5-4b11-aff4-f59125ab27bc/f3d05149-23e9-4ac2-8f99-0c851a8a5231/ovftool-430-userguide.pdf

7. Erstellen Sie ein Array zur Definition der `ovtool` Befehl:

```
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')
```

Hinweis Dieser Code definiert zwei Netzwerkzuordnungen für Sensoren und Paketspeicher. `VM Network` ordnet sich dem VM-Verwaltungsnetzwerk auf dem ESX/ESXi-Host zu. `Remote Port Mirror` ordnet sich dem Netzwerk zu, das wire data spiegelt. Die Namen der Zielnetzwerke müssen mit den Netzwerken auf Ihrem ESX/ESXi-Host übereinstimmen. Wenn Sie beispielsweise ein Netzwerk mit dem Namen konfiguriert haben `Local Port Mirror` um den internen VM-Verkehr zu spiegeln, geben Sie Folgendes an:

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

Für EDA 6100v-Appliances können Sie dem obigen Array zusätzliche Netzwerkzuordnungen für Remote Port Mirror 2 und Remote Port Mirror 3 hinzufügen, zum Beispiel:

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

8. Führen Sie den `ovftool` Befehl mit dem `subprocess.Popen` Klasse:

```
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")
```

Fügen Sie eine Festplatte in VMware hinzu (optional)

Für Sensoren die für die PCAP und alle EXAs lizenziert sind, müssen Sie eine zusätzliche Festplatte für die virtuelle Appliance konfigurieren.

1. Erstellen Sie Einträge für virtuelle Appliances in Ihrer Ansible-Inventardatei ähnlich dem folgenden Beispiel:

```
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

 32313761623336326566303039613131373465663633261303364353264326663353337396435 39

 62663864363635636533633061376538396333346664666 30a643032303035376137363839656330

 63396465323039666531353437663934343735663638303166316534316134633739626231386662

 32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

Die obigen Inventareinträge definieren die folgenden Konfigurationsvariablen:

Beispiel EXA
Der Name der VM.

Festplatte hinzufügen
Die Größe der hinzuzufügenden Festplatte in GB. Die folgenden Größenwerte sind gültig:

Virtuelle Appliance	Größe
EDA 1100 v	250
EDA 6100 v	500
EXA-XS	250 oder weniger
EXA-S	500 oder weniger
EXA-M	1000 oder weniger
EXA-L	2000 oder weniger

App-Typ
Der Typ der virtuellen Appliance.

Rechenzentrum
Das Rechenzentrum, das den ESX/ESXi-Host enthält.

Mappe
Der Ordner mit den VMs im Rechenzentrum.

esxi

Der Name des ESX/ESXi-Hosts, auf dem die VM bereitgestellt wird.

Datenspeicher

Der Name des Datenspeichers, der die VM enthalten wird.

Eizellen

Der relative Pfad der ExtraHop-OVA-Datei.

vcenter

Der Hostname des vCenter Server.

Benutzer

Der Name eines Benutzerkonto auf dem vCenter Server.

Passwort

Das Passwort des Benutzerkonto.

2. Erstellen Sie einen Aufgabeneintrag in einem .yml-Playbook, ähnlich dem folgenden Beispiel:

```
    - 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. Fügen Sie Ihrem Bibliotheksverzeichnis ein Python-Skript hinzu und öffnen Sie die Datei in einem Texteditor.

   In diesem Beispiel heißt das Skript `/library/eh_vsphere_add_disk.py`.

4. Importieren Sie alle Python-Module, die das Skript benötigt.

   Der folgende Code importiert die Module, die für den Rest des Verfahrens erforderlich sind:

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

5. Erstellen Sie ein AnsibleModule-Objekt und importieren Sie Variablen aus der .yml-Datei:

```
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. Übersetze die Größe der PCAP-Festplatte von GB in KB:

```

```

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

7. Stellen Sie eine Verbindung zum vCenter Server her:

```
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. Rufen Sie ein Objekt ab, das die VM darstellt, zu der Sie die Festplatte hinzufügen.

```
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. Suchen Sie den Speicherort, an dem das virtuelle Laufwerk hinzugefügt werden soll, indem Sie alle Geräte auf der VM durchqueren. Die Einheitsnummer der Festplatte sollte der Einheitsnummer der letzten virtuellen Festplatte auf dem Gerät plus eins entsprechen. Rufen Sie außerdem das Objekt ab, das den iSCSI-Controller für die VM darstellt:

```
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. Konfigurieren Sie die Festplattenspezifikationen:

```
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. Fügen Sie die Festplatte hinzu:

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

Erhöhen Sie die Packetstore-Festplatte auf einem ExtraHop-Packetstore (optional)

Das folgende Beispiel zeigt, wie die Kapazität der Packetstore-Festplatte auf einer ETA 1150v-Appliance erhöht werden kann.

Hinweis: Das Beispiel gilt nicht für ETA 6150-Appliances, die eine Packetstore-Festplatte zwischen 1 TB und 25 TB benötigen.

1. Erstellen Sie Einträge für virtuelle Appliances in Ihrer Ansible-Inventardatei ähnlich dem folgenden Beispiel:

```
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

 323137616233363265663030396131313734656633633261303364353264326663353337396435396

 626638643636356365336330613765383963333466646630a643032303035376137363839656330

 633964653230396665313534376639343437356636383031663165343161346337396262313866626

 323436306363656365 0a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

2. Erstellen Sie einen Aufgabeneintrag in einer YML-Playbook-Datei, ähnlich dem folgenden Beispiel:

```
   - 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. Fügen Sie Ihrem Bibliotheksverzeichnis ein Python-Skript hinzu und öffnen Sie die Datei in einem Texteditor.

   In diesem Beispiel heißt das Skript `library/eh_vsphere_increase_disk.py`.

4. Importieren Sie alle Python-Module, die das Skript benötigt.

   Der folgende Code importiert die Module, die für den Rest des Verfahrens erforderlich sind:

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

5. Erstellen Sie ein AnsibleModule-Objekt und importieren Sie Variablen aus der .yml-Datei:

```
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. Übersetze die Größe der Festplatte von GB in KB:

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

7. Stellen Sie eine Verbindung zum vCenter Server her:

```
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. Rufen Sie ein Objekt ab, das die VM der ETA darstellt:

```
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. Rufen Sie den Pfad der Datei ab, die die Festplatte unterstützt, deren Größe Sie ändern möchten:

```
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. Rufen Sie ein Objekt ab, das das Rechenzentrum darstellt, in dem sich die VM befindet:

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

```
        datacenter_obj = d
```

11. Erhöhen Sie die Größe der Festplatte:

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