# Automate virtual appliance deployment with VMware and Ansible

Published: 2022-09-27

This guide explains how to create Ansible playbooks and python scripts that deploy virtual appliances to a VMware vSphere server and configure ESX/ESXi hosts to mirror wire data.

You must have familiarity administering VMware and creating Ansible playbooks. Additionally, these procedures require the following VMware Python modules and command line tools:

**pyVmomi 6.7.3**

For installation instructions, see http://vmware.github.io/pyvmomi-community-samples/#getting-started.

**OVF Tool 4.2.0**

For installation instructions, see https://code.vmware.com/web/tool/4.3.0/ovf.

**Before you begin**

- Review VM requirements for the type of virtual appliance you are deploying:
  - Sensor VM requirements
  - ECA VM requirements
  - EXA VM requirements
  - ExtraHop packetstore VM requirements
- (Recommended) Upgrade to the latest patch for the vSphere environment to avoid any known issues.

## Mirror wire data

You must configure vSwitches on ESX/ESXi servers to mirror wire data to enable ExtraHop virtual appliances to monitor network traffic.

### Monitor external traffic

To monitor external traffic, you must create a second vSwitch on a network interface. This interface connects to a mirror, tap, or aggregator that copies traffic from a switch.

1. Create entries for the ESX/ESXi hosts that you want to configure in your Ansible inventory file, similar to the following example:

```
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

 62663864363635636533633061376538396333346664666630a643032303035376137363839656330

 63396465323039666531353437663934343735663638303166316534316134633739626231386662
```

```
32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 86133
```

The example inventory entries above define the following configuration variables:

**esxi1.example.com**
The name of the ESX/ESXi host that the virtual appliances are deployed to.

**vswitch**
The name of the second vSwitch to create. We recommend that you name the vSwitch by appending a number. For example, if `vSwitch1` already exists on your ESX/ESXi host, specify `vSwitch2`.

**port_group**
The name of the port group to create on the vSwitch.

**vmnic**
The name of the NIC to add the vSwitch to.

**vcenter**
The hostname of the vCenter server.

**user**
The name of a user account on the vCenter server.

**password**
The password of the user account.

2. Create a task entry in a .yml playbook file similar to the following example:

```
    - 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. Add a Python script to your library directory and open the file in a text editor.
In this example, the script is named `/library/eh_vsphere_external_mirror.py`.
4. Import any Python modules the script requires.
The following code imports the modules required for the rest of the procedure:

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

5. Create an AnsibleModule object and import variables from the .yml file:

```
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. Establish a connection to the vCenter server:

```
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. Retrieve an object that represents the ESX/ESXi server:

```
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. Configure and add the new virtual switch:

```
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. Configure the new port group:

```
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. Add the port group:

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

## Monitor intra-VM traffic

To enable a virtual appliance to monitor intra-VM traffic, you must create an additional port group on the default virtual switch of an ESX/ESXi host.

1. Create entries for the ESX/ESXi hosts you want to configure in your Ansible inventory file, similar to the following example:

```
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

 32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

   The example inventory entries above define the following configuration variables:

   **esxi1.example.com**
   : The name of the ESX/ESXi host that the virtual appliances are deployed to.

   **local_port_mirror**
   : The name of the port group to create on the vSwitch.

   **vcenter**
   : The hostname of the vCenter server.

   **user**
   : The name of a user account on the vCenter server.

   **password**
   : The password of the user account.

2. Create a task entry in a .yml playbook file, similar to the following example:

```
    - 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. Add a Python script to your library directory and open the file in a text editor.

   In this example, the script is named `/library/eh_vsphere_intra_mirror.py`.

4. Import any Python modules the script requires.

   The following code imports the modules required for the rest of the procedure:

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

```
import ssl
```

5. Create an AnsibleModule object and import variables from the .yml file:

```
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. Establish a connection to the vCenter server:

```
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. Retrieve an object that represents the ESX/ESXi server:

```
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. Configure the new port group:

```
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. Add the new port group:

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

## Deploy the OVA file

The following example shows how to set up a playbook and task library to deploy the OVA file for a sensor VM. The OVA file must be downloaded to the machine where Ansible is running.

You can download ExtraHop OVA files from the ExtraHop Customer Portal.

1. Create entries for virtual appliances in your Ansible inventory file, similar to the following example:

```
vms:
  hosts:
    exampleTestEDA:
      app_type: EDA
      datacenter: "Example Datacenter"
      folder: "VM Folder"
      esxi: esxi1.example.com
      datastore: DATA
      ova: ../firmware_images/extrahop-eda-1000v-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

  32343630636365636650a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

The above inventory entries define the following configuration variables:

**exampleEDA**
: The name of the VM.

**app_type**
: The type of virtual appliance.

**datacenter**
: The datacenter that contains the ESX/ESXi host.

**folder**
: The folder of VMs on the datacenter.

**esxi**
: The name of the ESX/ESXi host that the VM are deployed to.

**datastore**
: The name of the datastore that will contain the VM.

**ova**
: The relative path of the ExtraHop OVA file.

**vcenter**
: The hostname of the vCenter server.

**user**
: The name of a user account on the vCenter server.

**password**

The password of the user account.

2. Create a task entry in a .yml playbook, similar to the following example:

```
    - 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. Add a Python script to your library directory and open the file in a text editor.
   In this example, the script is named `/library/eh_vsphere_deploy.py`.
4. Import any Python modules the script requires.
   The following code imports the modules required for the rest of the procedure:

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

5. Create an AnsibleModule object and import variables from the .yml file:

```
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. Create the host string that specifies the location of the ESX/ESXi host in 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)
```

For more information about creating a host string, see the OVF Tool documentation on the VMware site: https://vdc-download.vmware.com/vmwb-repository/dcr-public/bb505ca7-88b5-4b11-aff4-f59125ab27bc/f3d05149-23e9-4ac2-8f99-0c851a8a5231/ovftool-430-userguide.pdf

7. Create an array to define the `ovtool` command:

```
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:** This code defines two network mappings for sensors and packetstores. `VM Network` maps to the VM management network on the ESX/ESXi host. `Remote Port Mirror` maps to the network that mirrors wire data. The names of the target networks must match the networks on your ESX/ESXi host. For example, if you configured a network named `Local Port Mirror` to mirror intra-VM traffic, specify:

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

For EDA 2000v and 6100v appliances, you can add additional network mappings to the above array for Remote Port Mirror 2 and Remote Port Mirror 3, for example:

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

8. Run the `ovftool` command with the `subprocess.Popen` class:

```
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")
```

## Add a disk in VMware (optional)

For sensors that are licensed for packet capture and all EXAs, you must configure an additional disk for the virtual appliance.

1. Create entries for virtual appliances in your Ansible inventory file similar to the following example:

```
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

  62663864363635636533633306137653839633334666466630a643032303035376137363839656330

  63396465323039666531353437663934343735663638303166316534316134633739626231386662

  32343630636365636 50a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

The above inventory entries define the following configuration variables:

**exampleEXA**
: The name of the VM.

**add_disk**
: The size in GB of the disk to add. The following size values are valid:

| Virtual appliance | Size |
|---|---|
| EDA 1000v | 250 |
| EDA 1100v | 250 |
| EDA 2000v | 250 |
| EDA 6100v | 500 |
| EXA-XS | 250 or less |
| EXA-S | 500 or less |
| EXA-M | 1000 or less |
| EXA-L | 2000 or less |

**app_type**
: The type of virtual appliance.

**datacenter**
: The datacenter that contains the ESX/ESXi host.

**folder**
: The folder of VMs on the datacenter.

**esxi**
: The name of the ESX/ESXi host that the VM is deployed to.

**datastore**
: The name of the datastore that will contain the VM.

**ova**
: The relative path of the ExtraHop OVA file.

**vcenter**
: The hostname of the vCenter server.

**user**

The name of a user account on the vCenter server.

**password**

The password of the user account.

2. Create a task entry in a .yml playbook similar to the following example:

```
    - 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. Add a Python script to your library directory and open the file in a text editor.

   In this example, the script is named `/library/eh_vsphere_add_disk.py`.

4. Import any Python modules the script requires.

   The following code imports the modules required for the rest of the procedure:

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

5. Create an AnsibleModule object and import variables from the .yml file:

```
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. Translate the size of the PCAP disk from GB into KB:

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

7. Establish a connection to the vCenter server:

```
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. Retrieve an object that represents the VM that you are adding the disk to.

```
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. Find the location to add the virtual disk by traversing all devices on the VM. The unit number of the disk should be the unit number of the last virtual disk on the device plus one. Also, retrieve the object that represents the iSCSI controller for the VM:

```
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. Configure the disk specifications:

```
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. Add the disk:

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

## Increase the packetstore disk on an ExtraHop packetstore (optional)

The following example shows how to increase the capacity of the packetstore disk on an ETA 1150v appliance.

**Note:** The example does not apply to ETA 6150 appliances, which require a packetstore disk between 1 TB and 25 TB.

1. Create entries for virtual appliances in your Ansible inventory file similar to the following example:

```
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

 32343630636365636650a32633634333439363833643330306362363637623164393432396139333 8
          6133
```

2. Create a task entry in a .yml playbook file similar to the following example:

```
   - 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. Add a Python script to your library directory and open the file in a text editor.

   In this example, the script is named `library/eh_vsphere_increase_disk.py`.

4. Import any Python modules the script requires.

   The following code imports the modules required for the rest of the procedure:

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

5. Create an AnsibleModule object and import variables from the .yml file:

```
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. Translate the size of the disk from GB into KB:

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

7. Establish a connection to the vCenter server:

```
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. Retrieve an object that represents the VM of the 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. Retrieve the path of file that backs the disk you want to resize:

```
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. Retrieve an object that represents the datacenter that the VM is on:

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

11. Increase the size of the disk:

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