Building Out a VMware Deployment

Now that we know how to launch networking and services in AWS, we will discuss deploying a similar setup in a **VMware** environment and talk through the core VMware modules.

In this chapter, we will cover the following topics:

* An introduction to VMware
* The VMware REST modules

Technical requirements

This chapter will discuss various components of the VMware family of products and how you can interact with them using Ansible. While there will be example playbook tasks in this chapter, they may need to be more easily transferable to your installation. Because of this, it’s not recommended that you use any examples in this chapter without first reviewing the complete documentation.

An introduction to VMware

With over 25 years of history, VMware has evolved significantly from its origins as a stealth startup. Boasting a revenue of over $13 billion in August 2023, the Vmware product portfolio, which grew to encompass around 30 products, is best known for its hypervisors, and it is a staple in most enterprises, enabling administrators to deploy virtual machines rapidly across various standard x86-based hardware configurations.

However, recent developments have seen significant changes following Broadcom’s acquisition of Vmware in late 2023.

This acquisition has dramatically simplified Vmware’s product portfolio, something that was influenced by customer and partner feedback, allowing users of all sizes to derive more value from VMware solutions. Two notable offerings include VMware Cloud Foundation and VMware vSphere Foundation, each with advanced add-on offers.

The first of the major changes that Broadcom has implemented is transitioning VMware to a subscription-based model. This aligns with the industry standard for cloud consumption and aims to provide continuous innovation, quicker time to value, and predictable investments for customers by phasing out perpetual licenses and replacing them with subscription or term licenses to enable customer and partner success in digital transformations.

There are concerns from the wider industry about Broadcom’s post-acquisition strategy for VMware. There’s speculation that Broadcom may focus on retaining only the largest and most profitable VMware customers and partners. This strategy could lead to a restructuring of VMware’s portfolio to better align with Broadcom’s business objectives, potentially including asset disposals and an even more streamlined product range.

At the time of writing (early 2024), the impact of these changes on VMware’s existing customer base and partner ecosystem is still unknown, with further details expected to emerge throughout the year as Broadcom continues to implement its strategic long-term plans for VMware.

The VMware REST modules

As already mentioned, there were around 30 products in the VMware range, and Ansible had modules that allowed you to interact with many of them.

However, due to product streamlining, we will just concentrate on the vmware.vmware\_rest namespace modules and won’t be looking at any of the community.vmware modules as these will lose all support at some point in 2025.

The difference between the two collections of modules is that, as implied by the name, the vmware.vmware\_rest modules use the VMware REST API to manage resources, whereas the ones in community.vmware use a Python library to interact with the various VMware endpoints to perform tasks.

The modules in the vmware.vmware\_rest namespace are split into three areas:

* **Appliance**: These modules manage your vCenter appliances, which are underlying resources that make up your vCenter deployment
* **Content**: The Content Library modules allow you to manage the services for defining and managing the library’s items, subscription, publication, and storage
* **vCenter**: These modules allow you to manage the workloads, such as virtual machines, running on top of your vCentre deployment

Let’s start by looking at the VMware REST appliance modules.

VMware REST appliance modules

At the time of writing, there are over 60 modules; these are split up into their own clearly labeled areas.

Access modules

To start with, we have the access modules:

* appliance\_access\_consolecli: This module allows you to enable or disable the console-based controlled CLI (TTY1).
* appliance\_access\_consolecli\_info: This module returns the current state of the console-based controlled CLI (TTY1); this will either be enabled or disabled.
* appliance\_access\_dcui: With this module, you can configure the state of the **Direct Console User Interface** (**DCUI** TTY2); again, you only have two options: enabled or disabled.
* appliance\_access\_dcui\_info: As you may have already guessed, this module returns either enabled or disabled for the DCUI TTY2 state.
* appliance\_access\_shell: Again, there isn’t much to this one in that you just change the enabled state of BASH. With this enabled, you will be able to access a BASH shell within the CLI.
* appliance\_access\_shell\_info: This module simply returns BASH access; this will either be enabled or disabled.
* appliance\_access\_ssh: This module sets the enabled state of the SSH-based controlled CLI.
* appliance\_access\_ssh\_info: This module returns the enabled state of the SSH-based controlled CLI.

As already mentioned, each of these modules either allows you to set the state of the access system or returns the currently configured state:

- name: "Enable SSH access"

  vmware.vmware\_rest.appliance\_access\_ssh:

    enabled: true

  register: access\_ssh\_result

CopyExplain

Each of the non-info modules has a single value of enabled, which accepts either true or false, as demonstrated earlier.

Health info modules

The next grouping of modules only returns information about the health of your system:

* appliance\_health\_applmgmt\_info
* appliance\_health\_database\_info
* appliance\_health\_databasestorage\_info
* appliance\_health\_load\_info
* appliance\_health\_mem\_info
* appliance\_health\_softwarepackages\_info
* appliance\_health\_storage\_info
* appliance\_health\_swap\_info
* appliance\_health\_system\_info

You would call one of the modules like this:

- name: "Get the system health status"

  vmware.vmware\_rest.appliance\_health\_system\_info:

  register: health\_system\_result

CopyExplain

This would return the current health of whichever of the services you are querying.

Infraprofile modules

Here, we have just two modules:

* appliance\_infraprofile\_configs: This module exports the selected profile
* appliance\_infraprofile\_configs\_info: This module lists all the registered profiles

The only valid state for the appliance\_infraprofile\_configs module is export:

- name: "Export the ApplianceManagement profile"

  vmware.vmware\_rest.appliance\_infraprofile\_configs:

    state: "export"

    profiles:

    - "ApplianceManagement"

  register: infraprofile\_configs\_result

CopyExplain

Here’s the output is JSON containing the profile for the selected configuration. In the preceding example, this is ApplianceManagement.

Local accounts modules

Here, we have three modules:

* appliance\_localaccounts\_globalpolicy
* appliance\_localaccounts\_globalpolicy\_info
* appliance\_localaccounts\_info

These modules allow you to set and query the global policy and return information on all or just one of the local accounts.

Monitoring modules

While there are only two modules here, they can be powerful when you combine them:

* appliance\_monitoring\_info: This module returns a list of monitors
* appliance\_monitoring\_query: This module allows you to query the monitors

Here’s an example query:

- name: "Query the monitoring backend"

  vmware.vmware\_rest.appliance\_monitoring\_query:

    start\_time: "2024-01-01 09:00:00+00:00"

    end\_time: "2024-01-01 10:00:00+00:00"

    names:

    - "mem.total"

    interval: "MINUTES5"

    function: "AVG"

  register: mem\_total\_result

CopyExplain

As you can see, with the preceding task, we are querying the total memory in 5 minutes, which averages between 9 A.M. and 10 A.M. on January 1, 2024.

Networking modules

This is where things start to get a little more complicated; each of the modules has an *info* equivalent where highlighted:

* appliance\_networking (plus info): This module resets and restarts network configuration on all interfaces. It also renews the DHCP leases for DHCP IP addresses.
* appliance\_networking\_dns\_domains (plus info): This module is used to manage the DNS search domains.
* appliance\_networking\_dns\_hostname (plus info): This module configures the **fully qualified domain name** (**FQDN**) hostname.
* appliance\_networking\_dns\_servers (plus info): This module can manage the DNS server configuration.
* appliance\_networking\_firewall\_inbound (plus info): This module sets an ordered list of firewall rules.
* appliance\_networking\_interfaces\_info: This module fetches information on a single network interface.
* appliance\_networking\_interfaces\_ipv4 (plus info): This module manages the IPv4 network configuration for the named network interface.
* appliance\_networking\_interfaces\_ipv6 (plus info): This module manages the IPv6 network configuration for the named network interface.
* appliance\_networking\_noproxy (plus info): This module configures servers for which no proxy configuration should be applied.
* Appliance\_networking\_proxy (plus info): This module configures which proxy server to use for the specified protocol.

The time and date modules

The following modules affect the time and date settings in some way:

* appliance\_ntp (plus info): This module manages the NTP server configuration
* appliance\_system\_time\_info: This module gets the system time
* appliance\_system\_time\_timezone (plus info): This module sets the time zone
* appliance\_timesync module (plus info): This module configures time sync mode

The remaining modules

The remaining modules cover appliance configuration and management:

* appliance\_services (plus info): You can use this module to restart a given service
* appliance\_shutdown (plus info): This module allows you to cancel a pending shutdown action
* appliance\_system\_globalfips (plus info): Using this module, you can enable or disable Global FIPS mode for the appliance
* appliance\_system\_storage (plus info): This module resizes all partitions to 100% of the disk size
* appliance\_system\_version\_info: This module gets version information
* appliance\_update\_info: This module gets the status of an appliance update
* appliance\_vmon\_service (plus info): This module lists details of services managed by vmon

This concludes the appliance section. Next, we’ll look at content modules.

VMware REST content modules

There are a small number of modules that allow you to manage and gather information on your content libraries:

* content\_configuration (plus info): This module updates the configuration
* content\_library\_item\_info: This module returns {@link ItemModel} when provided with an identifier
* content\_locallibrary (plus info): This module creates a new local library
* content\_subscribedlibrary (plus info): This module creates a new subscription

vCenter modules

This is where the more interesting things happen. Using these modules, you can launch, configure, and manage the entire life cycle of your virtual machines. Before we look at virtual machines, we’ll take a look at some of the supporting vCenter modules.

Supporting vCenter modules

These supporting modules allow you to manage things such as data centers, folders, data stores, and resource pools hosted within your vCenter:

* vcenter\_cluster\_info: This module retrieves information about the cluster corresponding to {@param.name cluster\_name}
* vcenter\_datacenter (plus info): This module adds a new data center to your vCenter inventory
* vcenter\_datastore\_info: This module fetches information about the data store using {@param.name datastore\_name}
* vcenter\_folder\_info: This module retrieves information on up to 1,000 folders in vCenter matching {@link FilterSpec} that the user you are connecting as has permission to see
* vcenter\_host (plus info): This module can be used to add a new standalone host to your vCenter
* vcenter\_network\_info: This module returns information about the first 1,000 visible networks in vCenter matching {@link FilterSpec}, depending on your permissions
* vcenter\_ovf\_libraryitem: This module is used to create an item in the content library from a virtual machine or virtual appliance
* vcenter\_resourcepool (plus info): This module deploys a resource pool
* vcenter\_storage\_policies\_info: This module fetches information about the storage policies available in vCenter; it returns a maximum of 1,024 results

Virtual machine modules

The final group of modules deals with creating and managing virtual machines and their associated resources. Let’s start by looking at the main module, vcenter\_vm.

The vcenter\_vm module is used to create virtual machines. For example, a basic task would look like this:

- name: "Create a Virtual Machine"

  vmware.vmware\_rest.vcenter\_vm:

    placement:

      cluster: "{{ lookup('vmware.vmware\_rest.cluster\_moid', '/learnansible\_dc/host/learnansible\_cluster') }}"

folder: "{{ lookup('vmware.vmware\_rest.folder\_moid', '/learnansible\_dc/vm') }}"

      resource\_pool: "{{ lookup('vmware.vmware\_rest.resource\_pool\_moid', '/learnansible\_dc/host/learnansible\_cluster/Resources') }}"

    name: "LearnAnsibleVM"

    guest\_OS: "UBUNTU\_64"

    hardware\_version: "VMX\_11"

    memory:

      hot\_add\_enabled: true

      size\_MiB: 4000

  register: LearnAnsibleVM\_output

CopyExplain

As you can see, we are using a few of the different lookup modules to find the cluster, data store, folder, and resource pool IDs – if we had this information, we could provide the IDs directly.

Once the virtual machine has been created, we can use the remaining modules to configure it more or manage its state:

* vcenter\_vm\_guest\_customization: This module applies guest customization to the virtual machine, such as running a script
* vcenter\_vm\_guest\_filesystem\_directories: Using this module, you can create a directory within the guest host operating system
* vcenter\_vm\_guest\_identity\_info: This module fetches information about the guest host
* vcenter\_vm\_guest\_localfilesystem\_info: This module grabs details of the local filesystems in the guest host operating system
* vcenter\_vm\_guest\_networking\_info: This module fetches details about the network configuration within the guest host operating system
* vcenter\_vm\_guest\_networking\_interfaces\_info: This module displays information about the network interfaces in the guest host operating system
* vcenter\_vm\_guest\_networking\_routes\_info: This module displays information about the network routes from within the guest host operating system
* vcenter\_vm\_guest\_operations\_info: This module grabs information about the guest host operating system’s status
* vcenter\_vm\_guest\_power (plus info): This module requests a soft shutdown, standby (suspend), or soft reboot from within the guest host operating system
* vcenter\_vm\_hardware: This module is used to update the hardware settings of the requested virtual machine
* vcenter\_vm\_hardware\_adapter\_sata (plus info): This module configures a virtual SATA adapter
* vcenter\_vm\_hardware\_adapter\_scsi (plus info): This module adds a virtual SCSI adapter
* vcenter\_vm\_hardware\_boot (plus info): This module is used to manage virtual machine boot-related settings
* vcenter\_vm\_hardware\_boot\_device (plus info): This module can set the virtual devices that will be used as the boot drive for your virtual machine
* vcenter\_vm\_hardware\_cdrom (plus info): This module attaches a virtual CD-ROM to your virtual machine
* vcenter\_vm\_hardware\_cpu (plus info): This module manages your virtual machine’s CPU settings
* vcenter\_vm\_hardware\_disk (plus info): This module connects virtual disks to your virtual machine
* vcenter\_vm\_hardware\_ethernet (plus info): This module connects a virtual Ethernet adapter to your virtual machine
* vcenter\_vm\_hardware\_floppy (plus info): This module adds a virtual floppy drive to the virtual machine
* vcenter\_vm\_hardware\_info: This module fetches your virtual machine’s virtual hardware settings information
* vcenter\_vm\_hardware\_memory (plus info): This module configures the memory settings
* vcenter\_vm\_hardware\_parallel (plus info): This module adds a virtual parallel port
* vcenter\_vm\_hardware\_serial (plus info): This module adds a virtual serial port
* vcenter\_vm\_info: This module returns information about your virtual machine
* vcenter\_vm\_libraryitem\_info: This module retrieves information about the library item associated with your virtual machine
* vcenter\_vm\_power (plus info): This module issues a boot, hard shutdown, hard reset, or hard suspend on a guest – that is, it presses the power button on the front
* vcenter\_vm\_storage\_policy (plus info): This module updates the storage policy of your virtual machine’s virtual hard disks
* vcenter\_vm\_storage\_policy\_compliance: This module updates and gathers information on your virtual machine’s storage policy compliance
* vcenter\_vm\_tools (plus info): This module is used to manage the configuration of VMware Tools
* vcenter\_vm\_tools\_installer (plus info): This module attaches the VMware Tools CD installer as a CD-ROM, making it available within the guest host operating system
* vcenter\_vmtemplate\_libraryitems (plus info): This module creates and returns information on items in the content library

As you can see, there is comprehensive support for managing your virtual machine resources using the vmware.vmware\_rest collection, and what’s better is that the modules are all designed to consume the official REST API, meaning that you can safely mix and match how you manage your resources within VMware, regardless of whether you use the CLI, web interface, or Ansible. Everything is managed via the same REST API.

Summary

As you have seen from the very long list of modules, you can do most of the management and configuration tasks you would be doing as a VMware administrator day-to-day using Ansible.

Add to this the modules we looked at in [*Chapter 8*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/8), *Ansible Network Modules*, for managing network equipment, and modules such as the ones that support hardware such as NetApp storage devices.

By doing this, you can build complex playbooks that span the physical devices, VMware elements, and virtual machines running within your on-premises enterprise-level virtualized infrastructure.

As mentioned at the start of this chapter, at the time of writing, there is a lot of upheaval at VMware. This chapter has been written to show the art of the possible rather than be a practical hands-on guide for managing your VMware resources using Ansible. For more details on the current state of the vmware.vmware\_rest collection, go to https://galaxy.ansible.com/ui/repo/published/vmware/vmware\_rest/.

In the next chapter, we will look at how to ensure that our playbooks are following best practices by scanning them for common issues and potential security problems.