

SD-WAN virtual|engine

Deployment Guide

11 January 2022

The SD-WAN virtual|engine provides the same functionalities as the SD-WAN physical ip|engine.

SD-WAN virtual|engines are positioned in virtualized environments, in the branch (uCPE or server) and in the Data Center. They are fully interoperable with SD-WAN physical ip|engines and are similarly managed by the SD-WAN Orchestrator platform in the Cloud.

This document describes how to deploy the virtual|engine models in some different contexts (used hypervisor and targeted deployment mode).

Contents

Introduction	3
SD-WAN virtual engine series models	3
Deployment Modes.....	3
Multiwan.....	3
Multipath.....	3
Supported Hypervisors	4
VMware ESXi (version 5.5 and later)	4
Linux KVM.....	4
Deploying on VMware ESXi Hypervisor.....	5
Requirements and Limitations.....	5
Preparing the Deployment.....	5
Deploying a virtual engine instance on VMware ESXi	6
Additional step for adding a 5 th network interface.....	11
Deploying on KVM	14
Requirements and Limitations.....	14
Preparing the Deployment.....	14

Deploying a virtual engine instance on KVM.....	15
Creating a virtual engine instance with 4 interfaces.....	15
Creating a virtual engine instance with 5 interfaces.....	16
Checks and start-up.....	17
Adding a 5 th interface to a virtual engine instance.....	17
Local configuration of the SD-WAN virtual engine.....	18
Configuration Drive for the SD-WAN virtual engine.....	18
Definition.....	18
Preparing the configuration text file.....	18
Building the config drive ISO file.....	20
Using config drive with the virtual engine.....	21
On VMware ESXi.....	21
On KVM.....	25
License and Serial Number of the SD-WAN virtual engine.....	26
Retrieving the Serial Number on VMware ESXi.....	26
On the virtual engine.....	26
In vSphere Client.....	26
In ESXi Shell.....	26
Retrieving the Serial Number on KVM.....	26
On the virtual engine.....	26
On Linux KVM host.....	27
Appendix.....	28

Introduction

SD-WAN virtual|engine series models

The SD-WAN virtual|engine series includes the three following models: v-ipe-S, v-ipe-M and v-ipe-L.

These models require the following allocation of resources:

	v-ipe-S	v-ipe-M	v-ipe-L
vCPUs	1	2	4
RAM (GB)	2	4	8
Storage size (GB) system+cache	1 + 30	1 + 60	1 + 100
Disk type	HDD 7200rpm	HDD 7200rpm	SSD
Number of network interfaces	4 or 5	4 or 5	4 or 5

The series supports auto-scaling, i.e. the capability for an instance to change its model in an automated way, when its allocated resources are modified (and after the instance has been restarted).

The three models are intended for use in different contexts, mainly depending on the size of the site.

	v-ipe-S	v-ipe-M	v-ipe-L
Targeted site	Branch office		Central site or small datacenter
Number of users	< 100	< 300	< 5,000

Deployment Modes

As mentioned above, all models can be instantiated with either 4 or 5 virtual network interfaces. The number of network interfaces has an impact on the deployment modes available.

All virtual|engine models are deployed in-path, either in multiwan or multipath mode. As with physical ip|engines, each network interface can be of type Layer 2 or Layer 3. Refer to the appropriate SD-WAN documentation for more information on multiwan and multipath modes, the type of interface (Bridge or Router) and the associated Use Cases.

Multiwan

This mode is available if the virtual|engine instance is created with 4 or 5 virtual network interfaces, although the 5th interface will not be used. The 4 virtual network interfaces used (1 LAN interface and from 1 to 3 WAN interfaces) must be created on the virtual|engine instance in the following non-customizable order: lan1, wan1, wan2 and wan3.

Multipath

This mode is available if 5 virtual network interfaces (2 LAN interfaces and 2 or 3 WAN interfaces) are created on the virtual|engine instance in the following non-customizable order: lan1, wan1, wan2, wan3 and lan2.

Notes

If less than 3 WAN interfaces are needed (e.g. 1 LAN + 2 WANs or 2 LANs + 2 WANs), then the unnecessary virtual network interfaces could be connected to a “dummy” virtual switch (which is not connected to any physical NIC or network).

It is possible to switch a virtual|engine instance from multiwan mode to multipath mode and vice versa.

For switching from multiwan mode to multipath mode, the virtual|engine instance must have 5 virtual network interfaces. If it is not the case, then a 5th interface (lan2) must be added.

When switching from multipath to multiwan, it is not required to remove the useless 5th interface (lan2).

Supported Hypervisors

The SD-WAN virtual|engine series can be deployed on the following platforms:

VMware ESXi (version 5.5 and later)

SD-WAN virtual|engine series is supported on VMware ESXi versions 5.5 and later. For details, see [Deploying on VMware ESXi](#).

Linux KVM

SD-WAN virtual|engine series has been successfully tested on:

- Ubuntu 16.04 and 18.04
- CentOS/RHEL 7.7 and 8.1

For details, see [Deploying on KVM](#).

Notes

Although this document does not include specific information related to deployments in the following environments, the SD-WAN virtual|engine series is known to be compatible with them:

- OpenStack,
- Adva Ensemble Connector,
- Cisco NFVIS with Cisco ENCS series.

SD-WAN virtual|engine series is not available for IaaS environments like Azure, AWS and GCP.

Deploying on VMware ESXi Hypervisor

Requirements and Limitations

The host CPU must be x86-based Intel or AMD CPU with virtualization extension.

See [virtual|engine series models](#) for the resource requirements of the virtual|engine model to be deployed.

See [Deployment modes](#) for explanations on the number of virtual network interfaces that can be created on a virtual|engine instance. The mapping between the network interfaces presented in ESXi and the virtual|engine interfaces is fixed, as follows:

Network Adapter numbering	virtual engine interfaces	
	4 interfaces	5 interfaces
1	lan1	lan1
2	wan1	wan1
3	wan2	wan2
4	wan3	wan3
5	X	lan2

Only **vmxnet3** paravirtual driver is supported.

SR-IOV and PCI-passthrough are not supported.

“Promiscuous mode” and **“Forged Transmits”** must be set to Accept on the vSwitch port groups that are connected to:

- a virtual|engine WAN interface configured in Bridge mode (Bridge or Hybrid deployment modes)
- a virtual|engine LAN interface (whichever deployment mode is used)

Preparing the Deployment

Three files are distributed, one per model of the virtual|engine series and named as follows:

- v-ipe-S_v[version].ova
- v-ipe-M_v[version].ova
- v-ipe-L_v[version].ova

1. Download these OVA files from the Support website and identify the one that corresponds to the model of virtual|engine that is to be instantiated.
2. Create and configure virtual switches and port groups according to the requirements mentioned above. You will need 4 or 5 virtual switches and port groups, depending on the targeted deployment mode.

In the case an interface will not be used, the associated virtual switch and port group may be “dummy” ones, i.e. they are not linked to any real network nor any physical NICs.

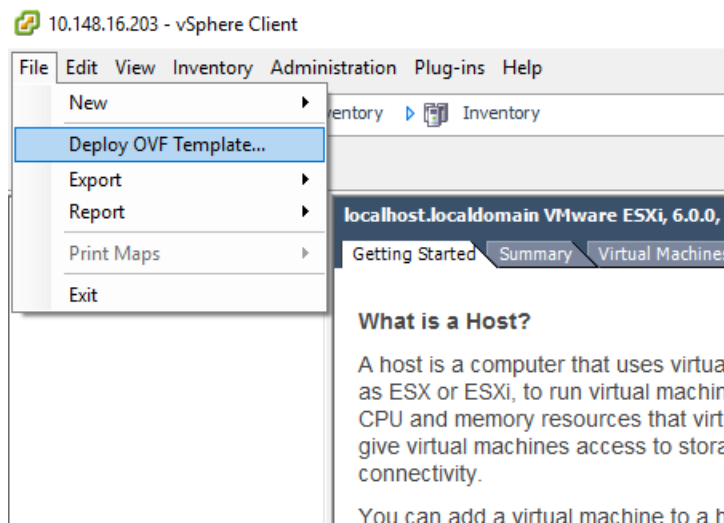
Deploying a virtual|engine instance on VMware ESXi

The following guidelines use VMware vSphere Client, but VMware ESXi Web Client might be used as well, without any restriction.

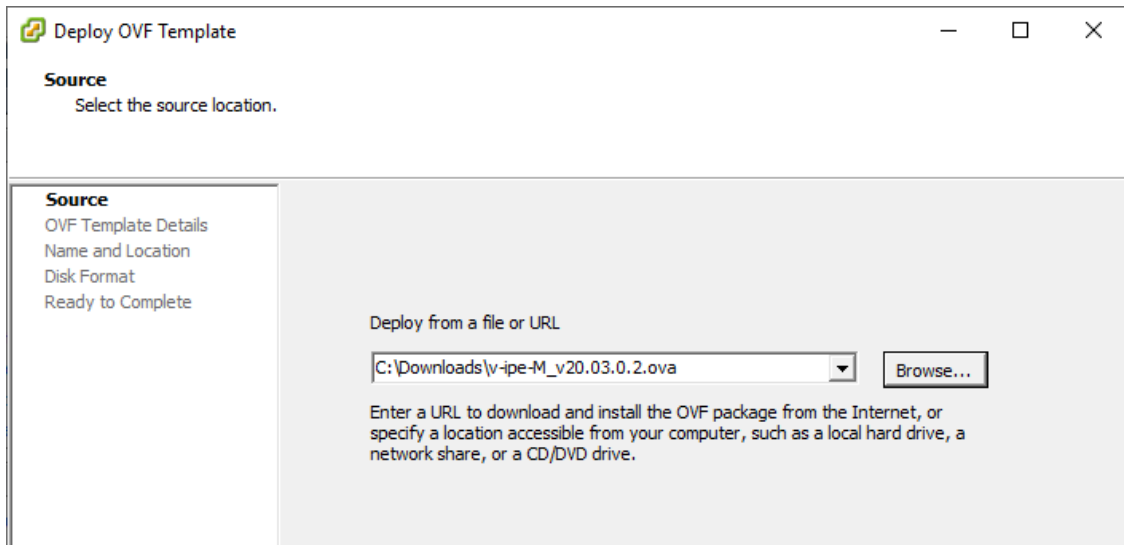
1. Open VMware vSphere Client, enter the IP address or host name of your VMware server and your user name & password and then click Login.



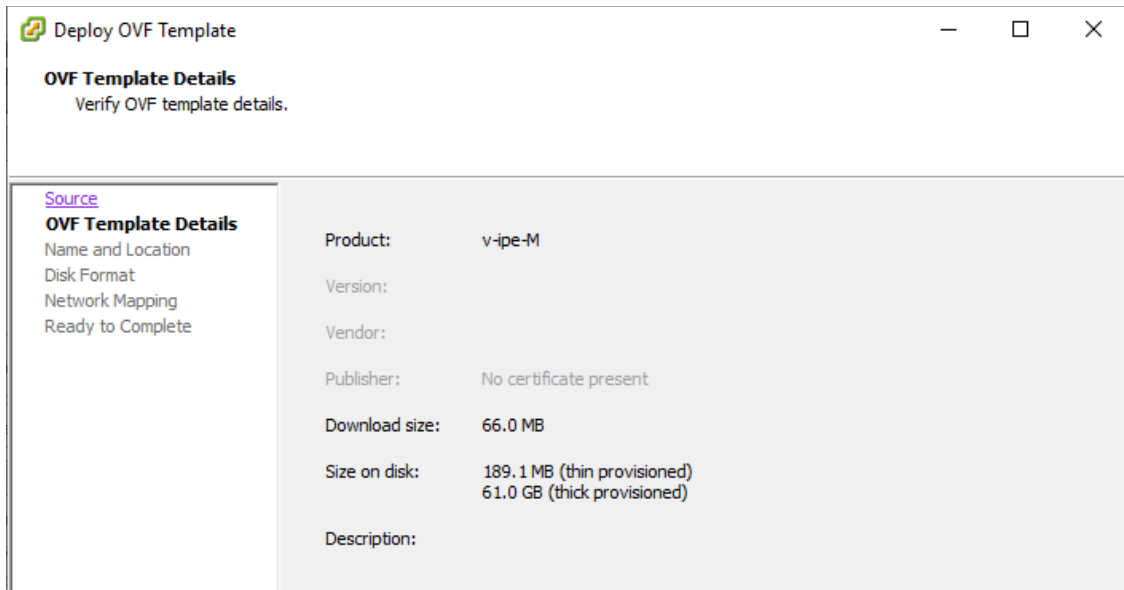
2. Select File > Deploy OVF Template...



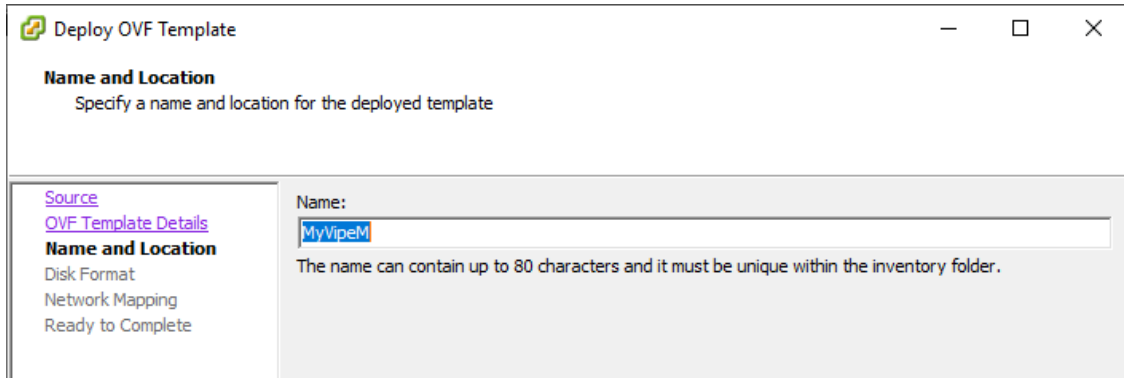
- Click Browse... and select the OVA file corresponding to the model of virtual|engine to install, open it and then click Next >.



- Verify that the OVA file corresponds to the model that you want to deploy (check "Product" information) and then click Next >. On the screenshot below, a v-ipe-M is being deployed.



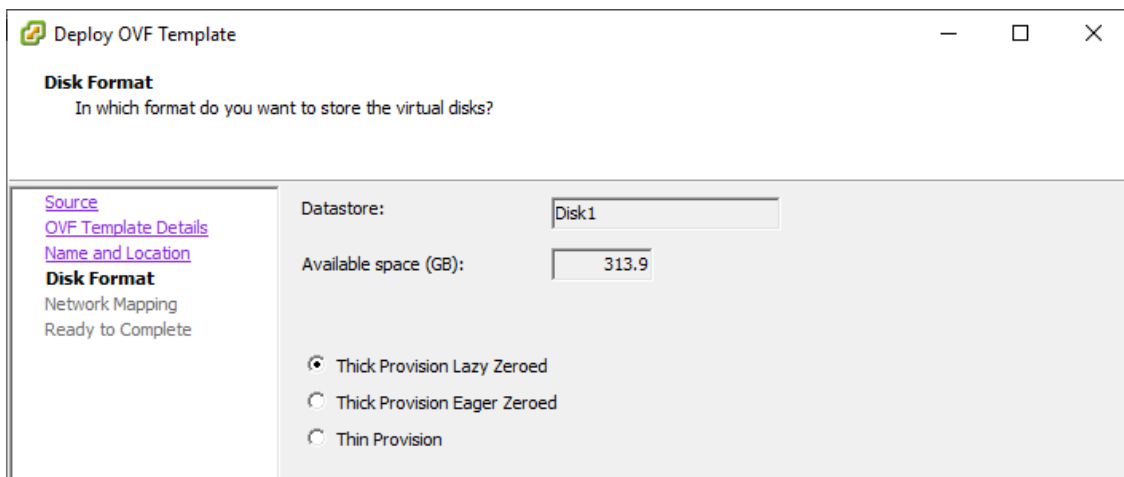
5. Enter the name of the virtual|engine instance to create and then click Next >.



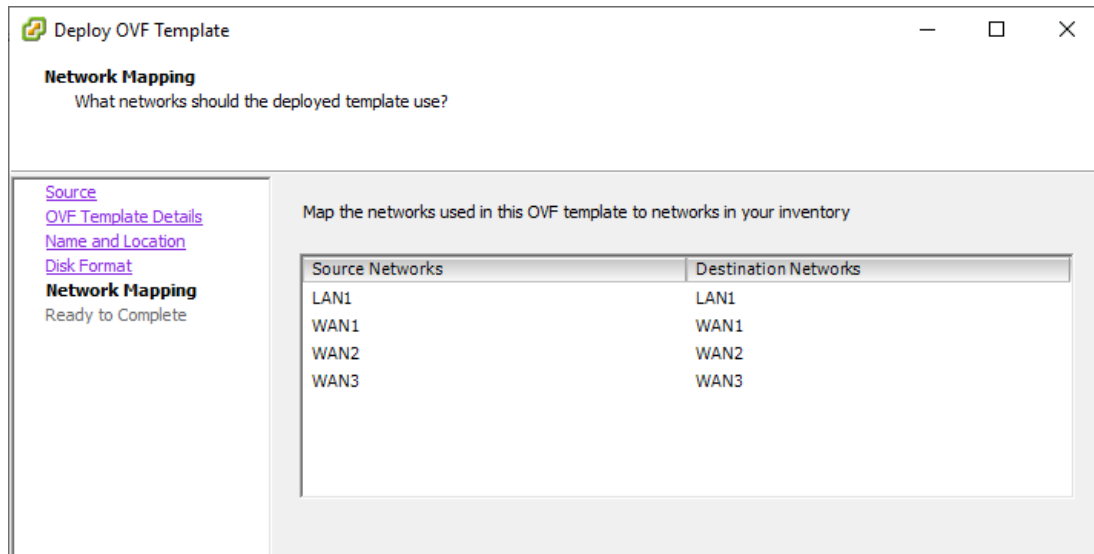
6. On the “Disk Format” page, choose a datastore, in which the virtual|engine and its virtual disk files (system and cache) will be stored. Select a provision mode, depending on the usage of the virtual|engine instance (Thick provision is recommended) and then click Next >.

The available modes are:

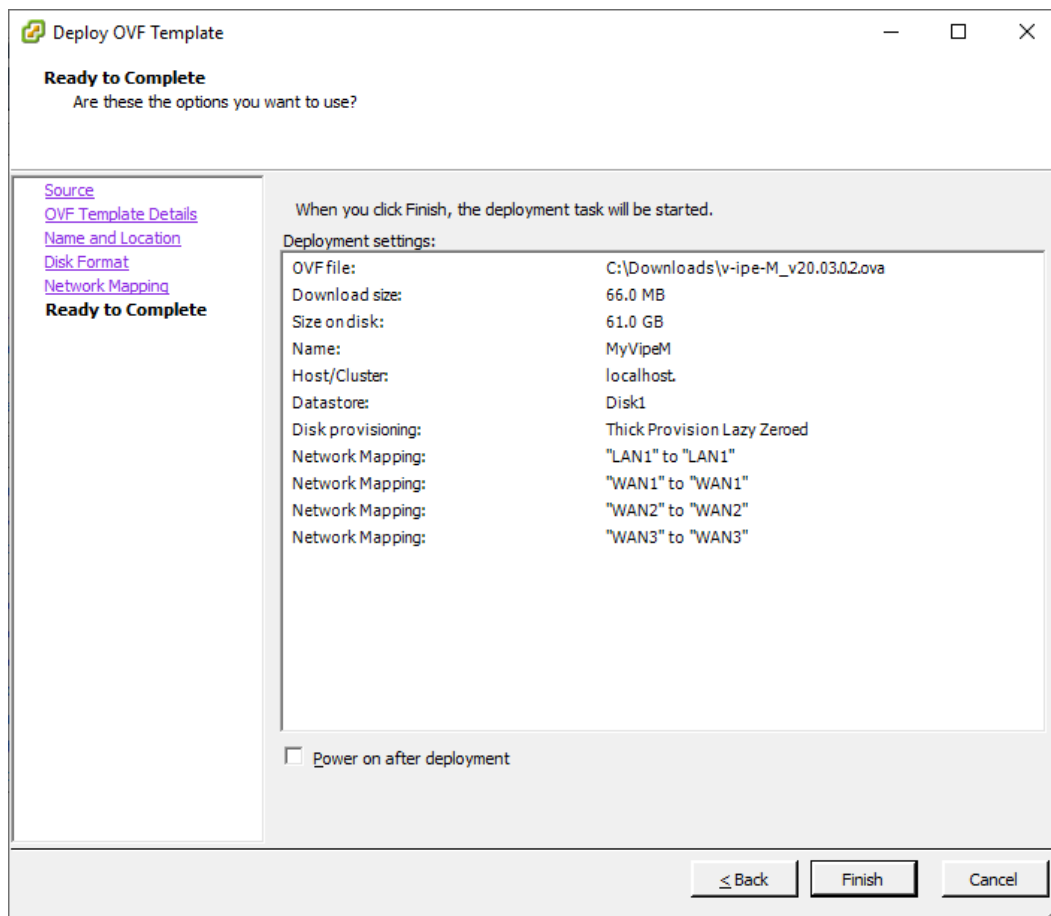
- Thick Provision Lazy Zeroed: recommended mode if you do not enable WAN Optimization. A lazy-zeroed thick disk has all its space allocated at the time of creation, but each block is zeroed only on first write; this results in a shorter creation time – about half a minute – but reduced performance the first time a block is written. Subsequent writes, however, have the same performance as on eager-zeroed thick disks.
- Thick Provision Eager Zeroed: recommended mode if you plan to enable WAN Optimization. An eager-zeroed thick disk has all its space allocated and zeroed out at the time of creation. This results in the best performance, even on the first write in each block but it increases the time it takes to create the disk.
- Thin Provision: this mode is not recommended as there may not be enough resources at the time of the first write. Unlike thick disks, thin disks do not have all their space allocated at the time of creation, but upon first write.



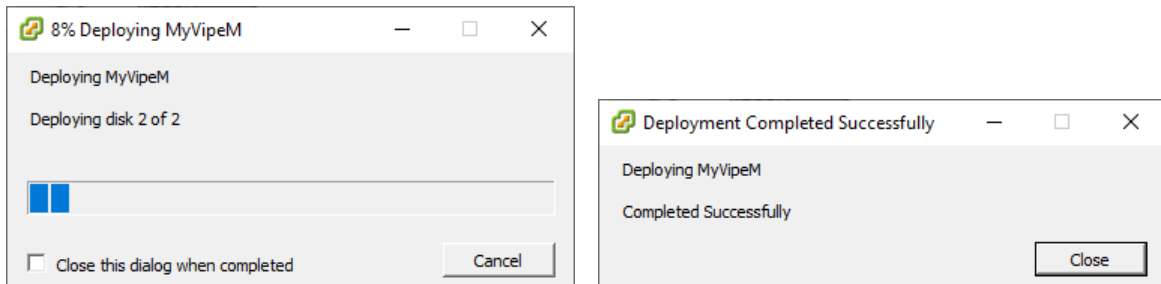
- On the “Network Mapping” page, map the Destination Networks (defined in ESXi) with the 4 Source Networks (LAN1, WAN1, WAN2, WAN3).
If a 5th interface is needed, it will be added manually as a later step.



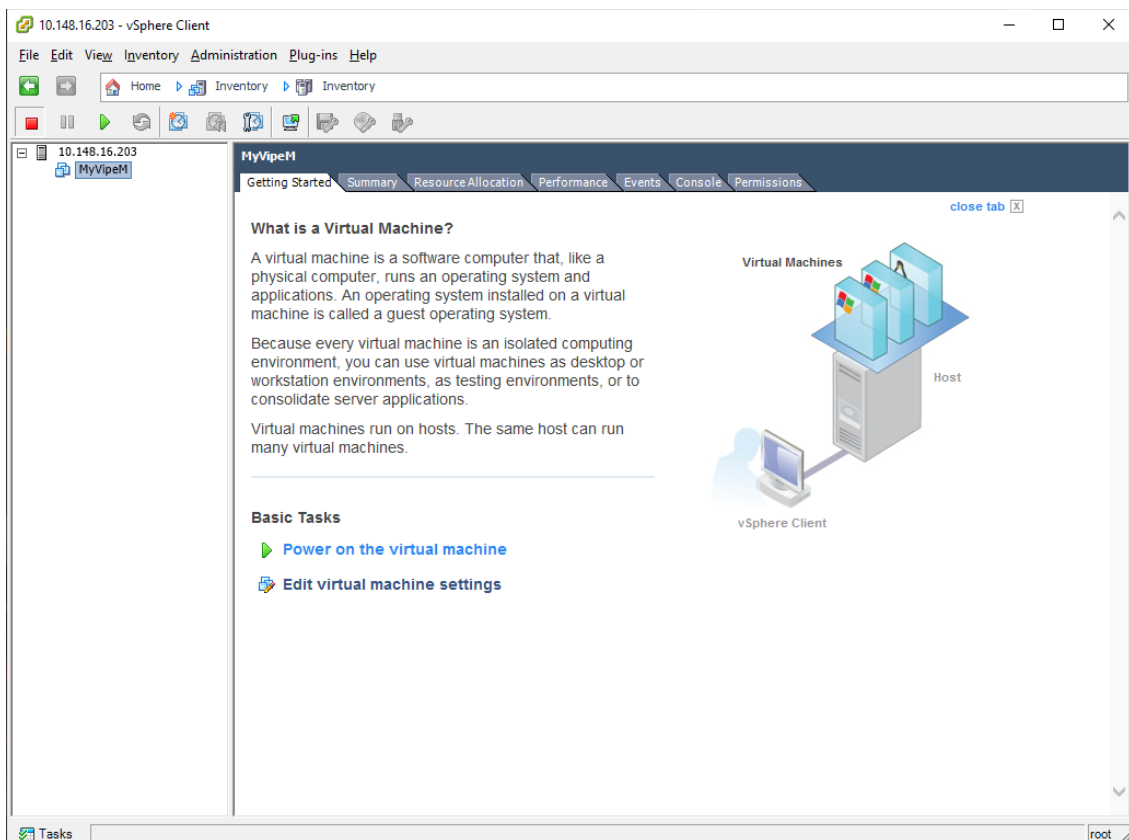
- On the “Ready to Complete” page, review the deployment details.
If you plan to add a 5th network interface or use the Configuration Drive feature, make sure that the checkbox “Power on after deployment” is unticked. Click Finish.



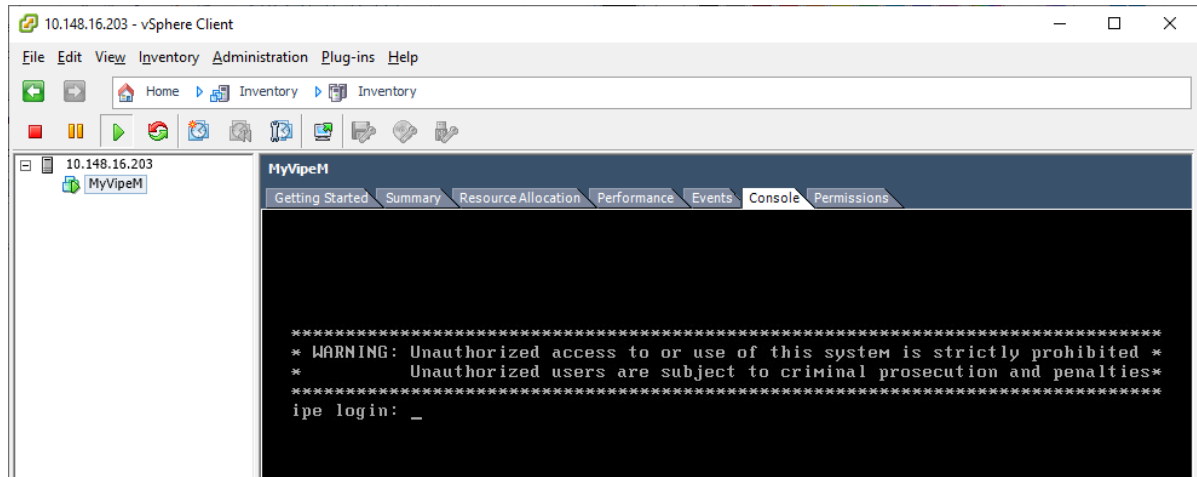
- A dialog box shows deployment progress and informs you when it has successfully completed. Click Close.



- Refer to [Configuration Drive for SD-WAN virtual|engine](#) for explanations regarding the way to use `config drive` to preconfigure the virtual|engine instance.
- Refer to [Additional step for adding a 5th network interface](#), if needed.
- The virtual|engine instance can now be powered on.



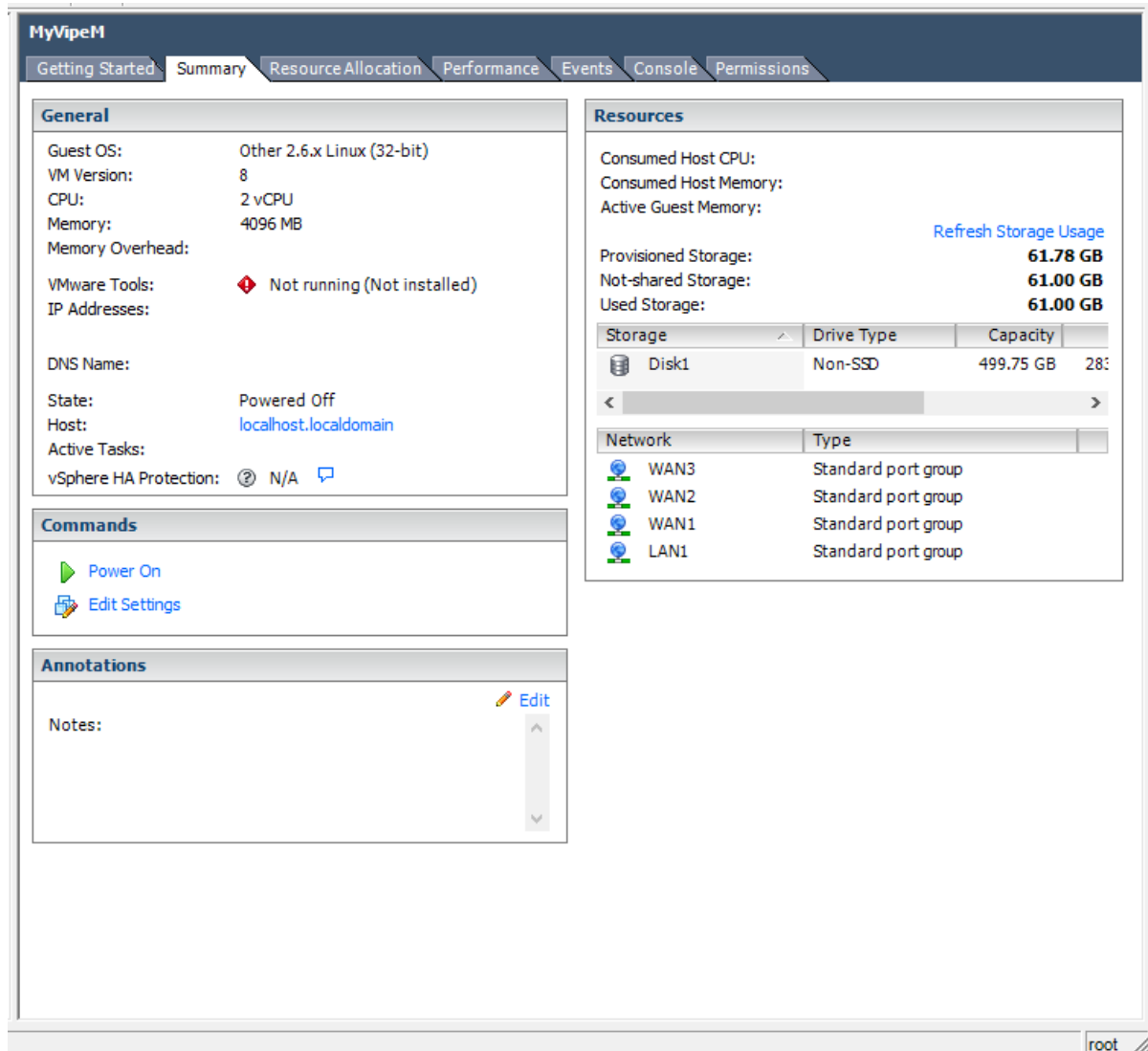
- At this stage, the virtual|engine instance is started and can be accessed. Select the Console tab of the virtual|engine instance and press Enter to activate the virtual console. At the login prompt, you should be able to log in (default credentials are `ipanema/ipanema`).
Note: American English is the keyboard layout by default.



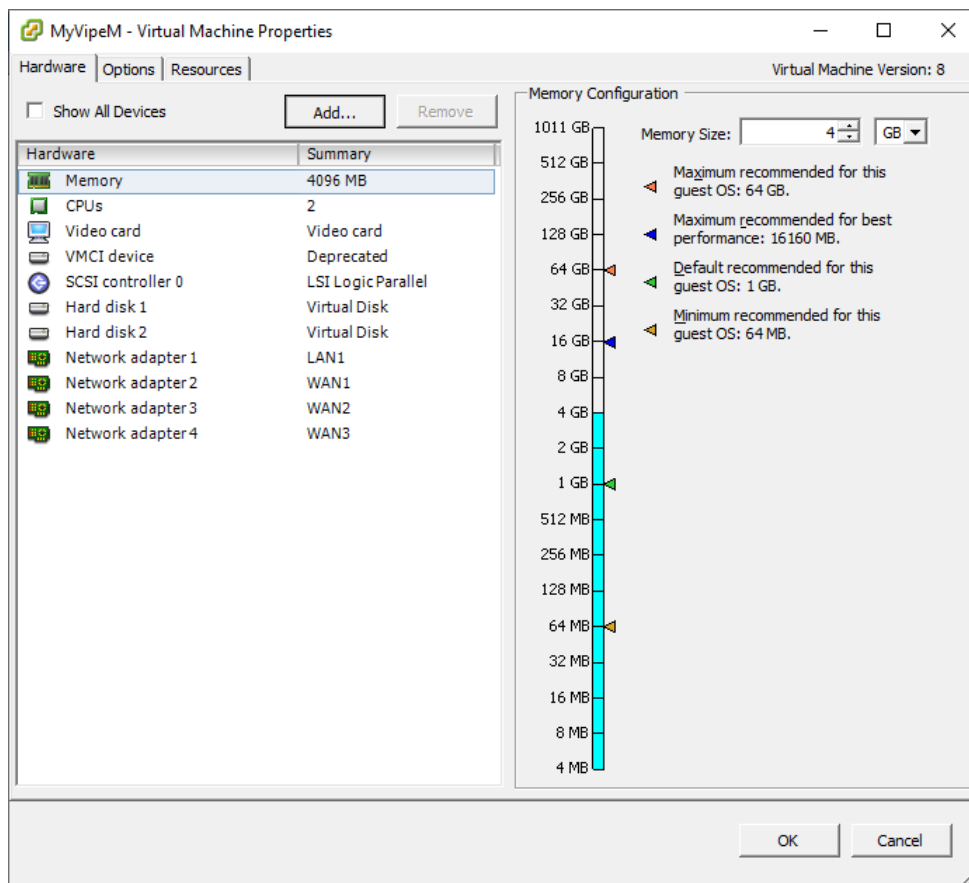
Additional step for adding a 5th network interface

As explained above, a 5th network interface (lan2) is required for multipath mode. After the deployment:

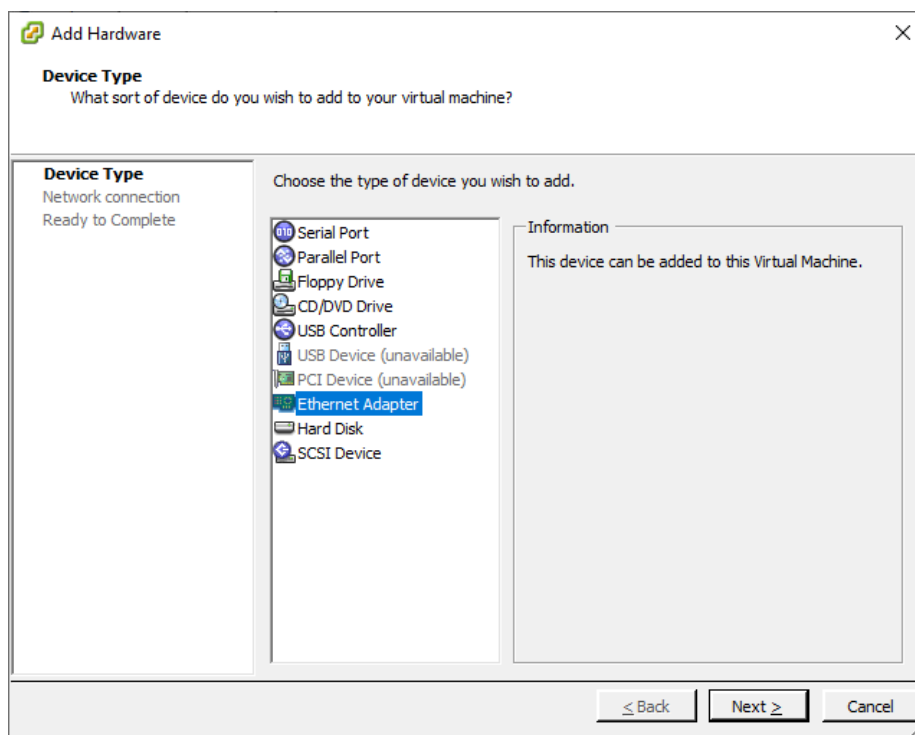
1. Select the Summary tab of the virtual|engine instance, open the Edit Settings page.



2. Select the tab Hardware and click Add....



3. On the “Device Type” page, select “Ethernet Adapter” and click Next >.



- On the “Network Type” page, select “VMXNET 3” as adapter type, choose the network to map with the lan2 interface and enable “Connect at power on”. Click Next >.

Add Hardware

Network Type
What type of network do you want to add?

Device Type
Network connection
Ready to Complete

Adapter Type
Type: VMXNET 3

Adapter choice can affect both networking performance and migration compatibility. Consult the [VMware KnowledgeBase](#) for more information on choosing among the network adapters supported for various guest operating systems and hosts.

Network Connection
Network label: LAN2
Port: N/A

Device Status
 Connect at power on

< Back Next > Cancel

- On the “Ready to Complete” page, review the details of the additional network interface and click Finish.

Add Hardware

Ready to Complete
Review the selected options and click Finish to add the hardware.

Device Type
Network connection
Ready to Complete

Options:

Hardware type: Ethernet Adapter
Adapter type: VMXNET 3
Network Connection: LAN2
Connect at power on: Yes

< Back Finish Cancel

Deploying on KVM

Requirements and Limitations

The host CPU must be x86-based Intel or AMD CPU with virtualization extension.

See [virtual|engine series models](#) for the resource requirements of the virtual|engine model to be deployed.

See [Deployment modes](#) for explanations about the number of virtual network interfaces that can be created on a virtual|engine instance. The order in which the network interfaces are defined (in the `libvirt` XML definition file or in `virt-manager` for example) is very important, as it is directly related to how the virtual|engine will use these interfaces:

Network interfaces definition order	virtual engine interfaces	
	4 interfaces	5 interfaces
1	lan1	lan1
2	wan1	wan1
3	wan2	wan2
4	wan3	wan3
5	X	lan2

Preparing the Deployment

For KVM, one package is distributed, named as follows: `v-ipe_kvm_v[version].tar.bz2`.

1. Download this package from the Support Web Site and unpack its contents. It should be as follows:

```
v-ipe_kvm_v[version]
+-- conf
|   +-- disk_ide_virtio.xml
|   +-- disk_scsi_virtio.xml
|   +-- v-ipe-L.xml
|   +-- v-ipe-M.xml
|   +-- v-ipe-S.xml
+-- images
|   +-- v-ipe-L_cache.img
|   +-- v-ipe-L.img -> v-ipe-S.img
|   +-- v-ipe-L.qcow2 -> v-ipe-S.qcow2
|   +-- v-ipe-M_cache.img
|   +-- v-ipe-M.img -> v-ipe-S.img
|   +-- v-ipe-M.qcow2 -> v-ipe-S.qcow2
|   +-- v-ipe-S_cache.img
|   +-- v-ipe-S.img
|   +-- v-ipe-S.qcow2
+-- md5sum.txt
+-- scripts
    +-- v-ipe_delete.sh
    +-- v-ipe_deploy.sh
```

This package contains:

- A top directory **v-ipe_kvm_v[version]** that includes all the elements listed below.
- A sub-directory **scripts** that contains scripts for deploying or deleting an instance of virtual|engine.
- A sub-directory **conf** that contains the pieces of the `libvirt` XML definition file that is used for deploying an instance of virtual|engine.

- A sub-directory **images** that contains the following image files:
 - **v-ipe-S.img**: the image in RAW format for the system disk of the v-ipe-S model
 - **v-ipe-S.qcow2**: the image in QCOW2 format for the system disk of the v-ipe-S model
 - For v-ipe-M and v-ipe-L models, no specific images in RAW or QCOW2 formats are provided for their system disks; v-ipe-S images can be used (with correct resource assignment for v-ipe-M or v-ipe-L models). Soft links are provided in the package, for simplifying the deployment script execution
 - **v-ipe-S_cache.img**: the image in QCOW2 format for the cache disk of the v-ipe-S model
 - **v-ipe-M_cache.img**: the image in QCOW2 format for the cache disk of the v-ipe-M model
 - **v-ipe-L_cache.img**: the image in QCOW2 format for the cache disk of the v-ipe-L model
 - A file **md5sum.txt** with the MD5 sum of the images provided in the package.
2. Check the integrity of the images; for example, with the following command that should return OK:

```
$ md5sum -c md5sum.txt
images/v-ipe-L_cache.img: OK
images/v-ipe-M_cache.img: OK
images/v-ipe-S_cache.img: OK
images/v-ipe-S.img: OK
images/v-ipe-S.qcow2: OK
```

3. Create and configure Linux bridges or Open vSwitch bridges that will be used during the creation of the virtual|engine instance. The script used later assumes that the following Linux bridges exist:
- **LAN1**: used for creating the 1st virtual|engine interface, i.e. lan1
 - **WAN1**: used for creating the 2nd virtual|engine interface, i.e. wan1
 - **WAN2**: used for creating the 3rd virtual|engine interface, i.e. wan2
 - **WAN3**: used for creating the 4th virtual|engine interface, i.e. wan3

If bridges have been named differently or Open vSwitch is used, then the definition of the **interface** nodes (in the conf/v-ipe*.xml files) needs to be adjusted accordingly.

In the case of an interface that isn't to be used, the associated bridge could be a "dummy" one, i.e. that isn't linked to any real network or any physical NICs.

If multipath deployment is intended, an additional bridge is necessary for creating the 5th virtual|engine interface, i.e. lan2. This bridge is named **LAN2** in the v-ipe_deploy.sh script and the guidelines below.

Deploying a virtual|engine instance on KVM

The following guidelines assume that a qemu/KVM environment (Ubuntu 18.04 used in the current example) is already ready with `libvirt` installed and Linux or Open vSwitch bridges created. The deployment is performed directly on the Linux KVM host, using a script (`v-ipe_deploy.sh`) that is provided in the virtual|engine package for KVM.

The script (`v-ipe_deploy.sh`) uses `libvirt virsh` command and `libvirt XML` definition files. The script usage is slightly different, depending on the number of network interfaces to be created (4 or 5). This is described hereafter.

Creating a virtual|engine instance with 4 interfaces

On the Linux KVM host, go to the directory where the virtual|engine package has been unpacked and execute the following command:

```
$ sudo ./v-ipe_deploy.sh [model] [driver] [image_format] 1 [instance_name]
```

Where:

- [model] indicates the virtual|engine model: **v-ipe-S**, **v-ipe-M** or **v-ipe-L**
- [driver] indicates the combination of drivers used for disk:
 - **ide**: the disk with the system partition uses driver ide and the disk with the cache partition uses driver virtio
 - **scsi**: both disks (with system and cache partitions) use driver virtio-scsi
- [image_format]: the format of the image to use for the system disk, either **raw** (for RAW format) or **qcow2** (for QCOW2 format)
- [instance_name] is the virtual|engine instance name

In the following example, a v-ipe-M instance named MyVipeM is created with a system disk image in QCOW2 format, virtio-scsi driver for both disks and 4 network interfaces:

```
$ sudo ./v-ipe_deploy.sh v-ipe-M scsi qcow2 1 MyVipeM
```

The script output informs about the deployment and its completion:

```
...
Deploying MyVipeM
...
Congratulations: MyVipeM (v-ipe-M) deployed successfully !
You can now manage it using virt-manager or virsh commands
```

Creating a virtual|engine instance with 5 interfaces

On the Linux KVM host, go to the directory where the virtual|engine package has been unpacked and execute the following command:

```
$ sudo ./v-ipe_deploy.sh [model] [driver] [image_format] 2 [instance_name]
```

Where:

- [model] indicates the virtual|engine model: **v-ipe-S**, **v-ipe-M** or **v-ipe-L**
- [driver] indicates the combination of drivers used for disk:
 - **ide**: the disk with the system partition uses driver ide and the disk with the cache partition uses driver virtio
 - **scsi**: both disks (with system and cache partitions) use driver virtio-scsi
- [image_format]: the format of the image to use for the system disk, either **raw** (for RAW format) or **qcow2** (for QCOW2 format)
- [instance_name] is the virtual|engine instance name

In the following example, a v-ipe-M instance named MyVipeM is created with a system disk image in QCOW2 format, virtio-scsi driver for both disks and 5 network interfaces:

```
$ sudo ./v-ipe_deploy.sh v-ipe-M scsi qcow2 2 MyVipeM
```

The script output informs about the deployment and its completion:

```
...
Deploying MyVipeM
...
Congratulations: MyVipeM (v-ipe-M) deployed successfully !
You can now manage it using virt-manager or virsh commands
```


Checks and start-up

At this stage, the virtual|engine instance is created but not started yet; this can be verified through:

```
$ virsh list -all
  Id      Name                               State
-----
-       MyVipeM                            shut off
```

Refer to the paragraph [Configuration Drive for SD-WAN virtual|engine](#) for explanations regarding the way to use the config drive feature to preconfigure the virtual|engine instance.

Instance can then be started:

```
$ virsh start [instance_name]
```

In the example used previously, it would be:

```
$ virsh start MyVipeM
```

Now, the virtual|engine instance can be accessed through the virtual serial console:

```
$ virsh console [instance_name]
```

In the example used previously, it would be:

```
$ virsh console MyVipeM
```

Press Enter to activate the console. At the login prompt, you should be able to log in (default credentials are ipanema/ipanema).

Adding a 5th interface to a virtual|engine instance

In the case a virtual|engine instance was first created with 4 virtual network interfaces but then needs to be switched from multiwan mode to multipath mode, a 5th interface must be added. The procedure is as follows:

```
$ virsh attach-interface [instance_name] bridge LAN2 --model virtio --
config
```

Finally, start or reboot the virtual|engine instance:

```
$ virsh start [instance_name]
```

or

```
$ virsh reboot [instance_name]
```

Local configuration of the SD-WAN virtual|engine

Like physical SD-WAN ip|engines, virtual|engines are configured through the Cloud-based Orchestrator, but some settings can be configured locally on the appliance. This is especially required when DHCP cannot be used. Consequently, the virtual|engine needs to be manually configured to connect to the Orchestrator and retrieve its definitive configuration.

Refer to the **SD-WAN ip|engine Quick Installation Sheet** for more information on CLI for local configuration and the associated use cases.

virtual|engines can be accessed to execute commands for local configuration through SSH or through the virtual console (see the end of deployment procedure on VMware ESXi or KVM).

Unlike physical SD-WAN ip|engines, virtual|engines may be locally configured in a silent way, via the [Configuration Drive](#) feature.

Configuration Drive for the SD-WAN virtual|engine

Definition

Configuration Drive (aka. config drive) is the mechanism that is provided to silently preconfigure a virtual|engine instance (when local configuration is required – refer to [Local configuration of SD-WAN virtual|engine](#)) and, thus, avoids accessing the device and launching manual commands.

Config Drive is essentially an ISO file passed as a CD-ROM drive with the label config-2, which is attached to the virtual|engine instance when it boots. The virtual|engine then mounts this drive and reads a configuration text file to configure itself silently.

This is a one-time configuration mechanism. Once a virtual|engine instance has been configured through config drive, it will neither read nor interpret its content at subsequent boots.

Preparing the configuration text file

The local configuration information that is silently passed to the virtual|engine instance is described in a text file with a specific JSON structure, which is described below.

The configuration file is composed of one “configuration” object, which can nest a “net” object, an “interfaces” array and an “account” object. All these sections are optional.

```
{
  "configuration": {
    "net": {
...
    },
    "interfaces": [
...
    ],
    "account": {
...
    }
  }
}
```

The section “net” corresponds to the configuration of the virtual|engine when deployed in Bridge or Hybrid modes, for example when the virtual|engine’s internal switch cannot be configured through DHCP (see the appropriate paragraph in the **SD-WAN ip|engine Quick Installation Sheet**). The example below provides an exhaustive overview of the configuration possibilities:

```
...
  "net": {
    "mode": "multipath",
```

```

    "gateway": "10.20.0.2",
    "address": "172.18.3.109",
    "mask": "255.255.255.0",
    "vlan_id": "none",
    "mtu": 1500,
    "swmacsrc": "own",
    "hostname": "myname",
    "dns_servers": [
      "10.48.26.1",
      "10.48.26.2"
    ],
    "dns_search": [
      "iv.local"
    ],
    "routes": [
      {
        "net": "10.170.16.0",
        "netmask": "255.255.255.0",
        "gw": "10.170.255.252"
      }
    ]
  },
  ...

```

Explanations:

- mode: optional parameter that indicates the deployment mode (“multiwan” or “multipath”), multiwan being the default
- gateway: optional parameter that sets the default gateway of the virtual|engine (optional)
- address, mask, vlan_id, mtu: the IP address and mask, VLAN ID and MTU of the virtual|engine
 - This group of parameters is optional.
 - If used, at least configure address and mask (i.e. vlan_id and mtu are optional within the group).
- swmacsrc: an optional parameter to set the MAC address of the virtual|engine’s internal switch (default corresponds to lan1’s MAC address); possible values are:
 - def: the default value, switch MAC address is lan1’s MAC address
 - own: switch has its own MAC address (OUI is 00:04:9D)
 - lanN: switch has the same MAC address as lanN (lan1 or lan2)
 - wanN: switch has the same MAC address as wanN (wan1, wan2 or wan3)
- hostname: optional parameter to set the host name of the virtual|engine
- dns_servers: an optional array to configure up to two DNS servers
- dns_search: an optional array to specify the DNS suffix search list
- routes: an optional array of static routes, each of them defined by three mandatory parameters: “net”, “netmask” and “gw”

The section “interfaces” corresponds to the manual configuration of the virtual|engineWAN interfaces. For example, this is needed when DHCP cannot be used to configure a virtual|engine in mode Full Router (see the corresponding paragraph in the **SD-WAN ip|engine Quick Installation Sheet**).

```

...
  "interfaces": [
    {
      "name": "wan1",
      "mode": "bridge"
    },
    {
      "name": "wan2",
      "mode": "router",
      "address": "10.17.16.1",

```

```

    "mask": "255.255.0.0",
    "gateway": "10.17.255.252",
    "dns_server": "10.108.26.1"
  },
  {
    "name": "wan3",
    "mode": "router",
    "address": "10.18.16.1",
    "mask": "255.255.0.0",
    "gateway": "10.18.255.252",
    "dns_server": "10.109.26.1"
  }
],
...

```

When defining an interface, the following rules must be followed:

- name: this is a mandatory information (either “wan1”, “wan2” or “wan3”)
- mode: this is either ‘bridge’ (layer 2 interface) or ‘router’ (layer 3 interface)
- address, mask, gateway, dns_server: the IP address and mask, default gateway and DNS server for the configured WAN interface
 - These parameters are all optional.
 - address and mask need to be set together

The section “account” corresponds to the definition of a password for the “ipanema” account (default password is “ipanema”). There are two alternatives for setting a password, either in clear text (which is not recommended) or in encrypted format. See below for examples:

```

...
  "account": {
    "password": "MyPassword2017?"
  }
...
or
...
  "account": {
    "enc_password":
"$6$mylongsalt$P3PzatX8uxBxweuHoPQErDQsJDja/pqbvMNPXbY/vc2PsFzRbpWu5I9HpBP0XPvonQO
yc.0ql2UORCNu3C1x0"
  }
...

```

The encrypted password can be produced by using a command like the following (salt option is optional):

```
$ mkpasswd --salt=[long_salt] --method=sha-512 [password]
```

Refer to the [Appendix](#) for a complete example of configuration JSON file.

Building the config drive ISO file

Once the configuration file is ready, the config drive ISO file must be built. To do so, it is assumed that you use a Linux host with `mkisofs` tool, but any other OS and tool may be used if the directory structure described below and the config-2 label are respected, when building this ISO file.

The configuration text file with the JSON structure described previously is named `user_data`; the ISO file to build must have the following content:

```

openstack
+-- latest
   +-- user_data

```

This can be achieved this way:

```
$ mkdir -p config-drive/openstack/latest
$ cp user_data config-drive/openstack/latest/
$ mkisofs -R -V config-2 -o configdrive.iso config-drive
```

Note

It is possible to pass the configuration text file to the virtual|engine as an injected file, although it is recommended to pass it as user_data. In this case, the configuration text file must be present in openstack/content/ directory and listed in the openstack/latest/meta_data.json file.

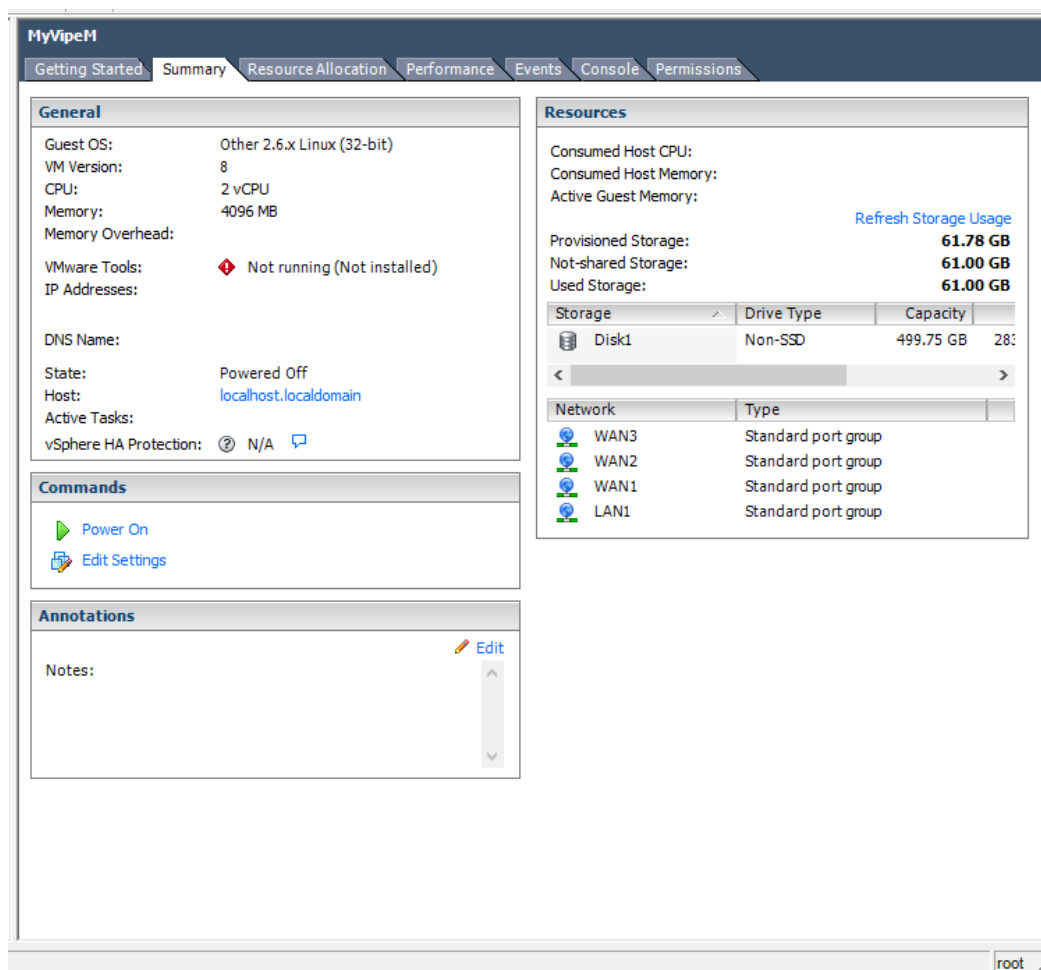
Using config drive with the virtual|engine

Using config drive ISO file depends on the chosen hypervisor.

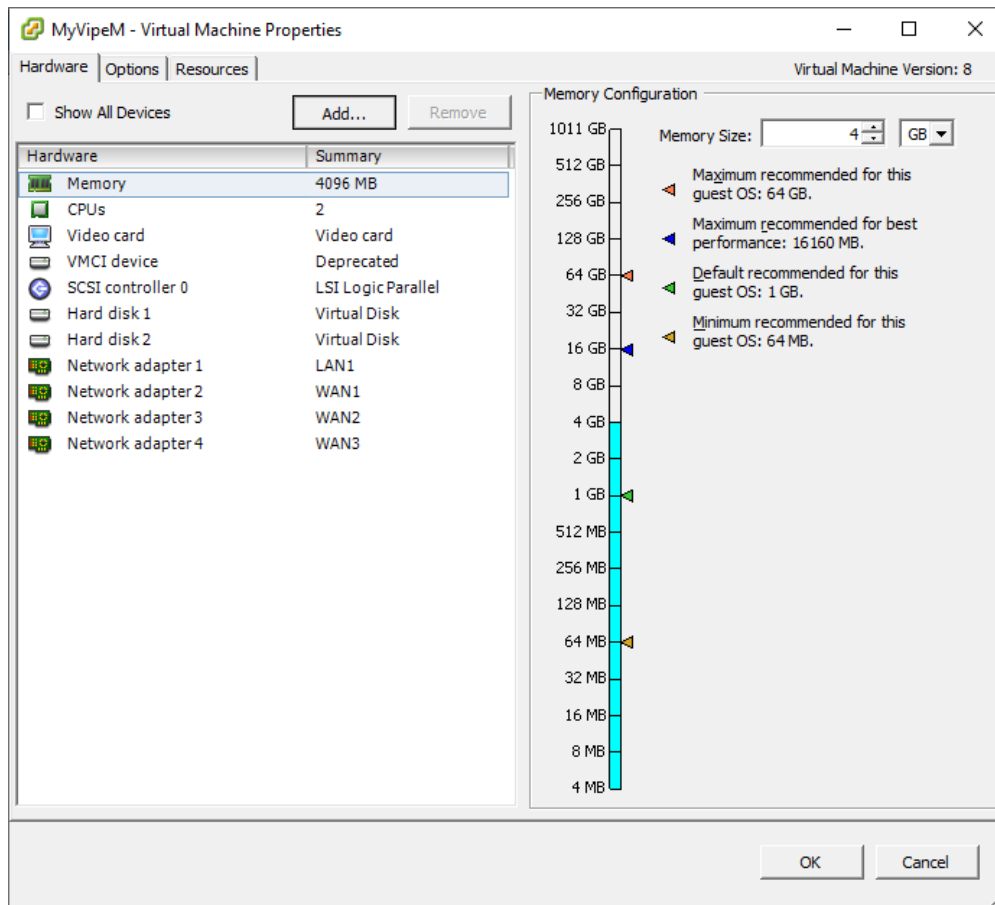
On VMware ESXi

The procedure below describes how config drive can be attached to a virtual|engine instance that has been deployed on VMware ESXi:

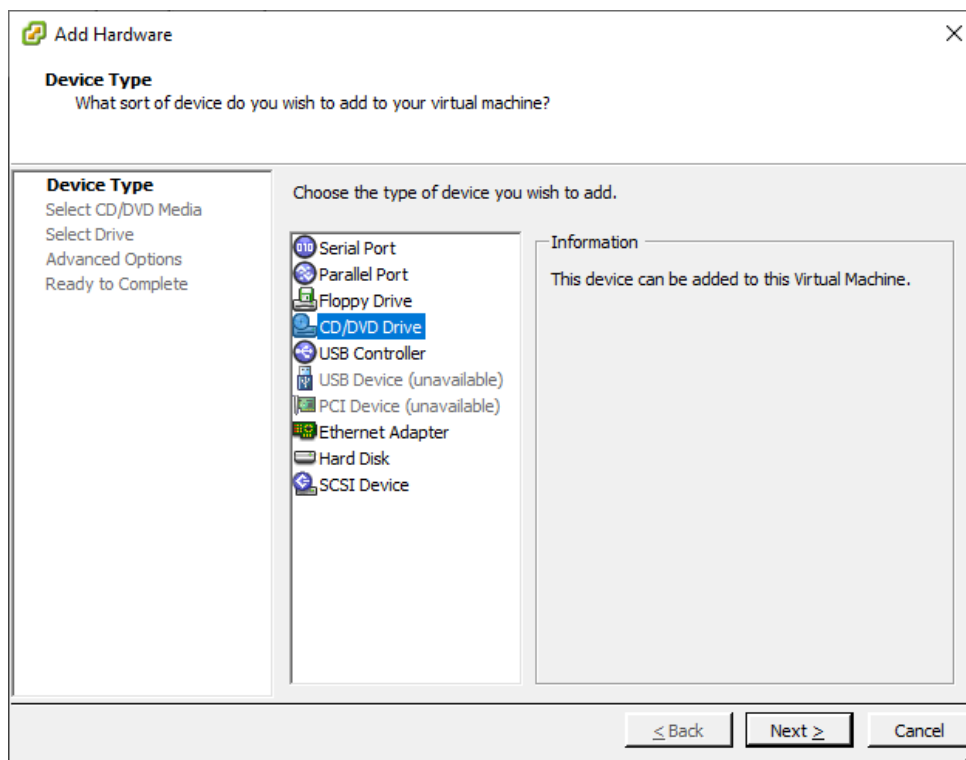
1. Select the Summary tab of the virtual|engine instance, open the Edit Settings page.



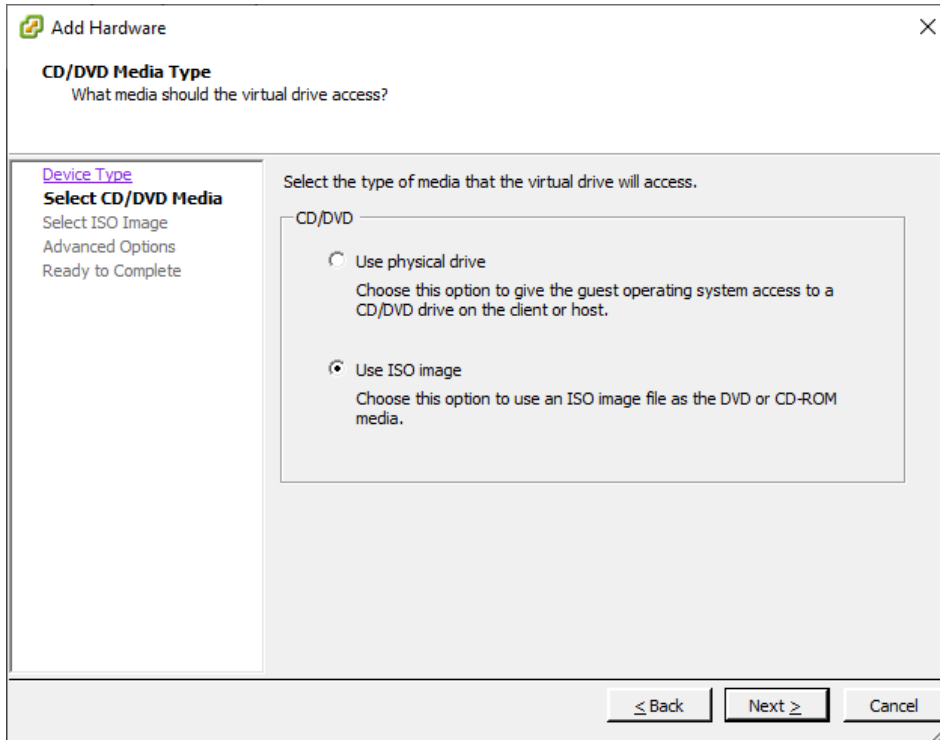
2. Select the tab Hardware and click Add....



3. On the "Device Type" page, select "CD/DVD Drive" and click Next >.



4. On the “CD/DVD Media Type” page, select “Use ISO image” and click Next >.



Add Hardware [Close]

CD/DVD Media Type
What media should the virtual drive access?

[Device Type](#)
Select CD/DVD Media
Select ISO Image
Advanced Options
Ready to Complete

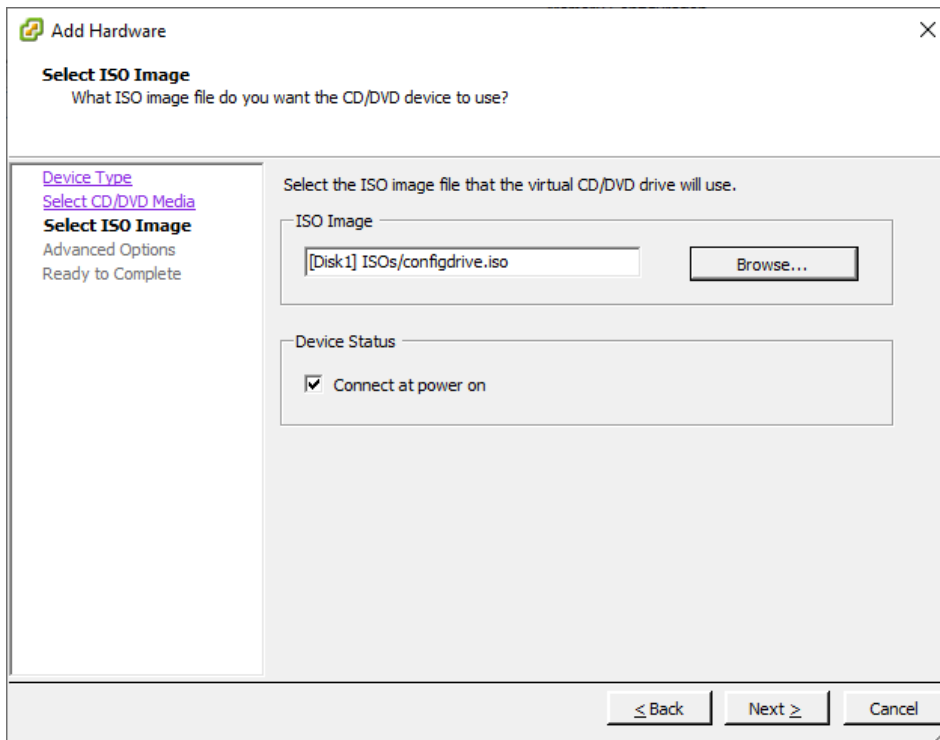
Select the type of media that the virtual drive will access.

CD/DVD

- Use physical drive
Choose this option to give the guest operating system access to a CD/DVD drive on the client or host.
- Use ISO image
Choose this option to use an ISO image file as the DVD or CD-ROM media.

[< Back] [Next >] [Cancel]

5. Select an ISO image from the datastore(s) and enable “Connect at power on”. Click Next >.



Add Hardware [Close]

Select ISO Image
What ISO image file do you want the CD/DVD device to use?

[Device Type](#)
[Select CD/DVD Media](#)
Select ISO Image
Advanced Options
Ready to Complete

Select the ISO image file that the virtual CD/DVD drive will use.

ISO Image

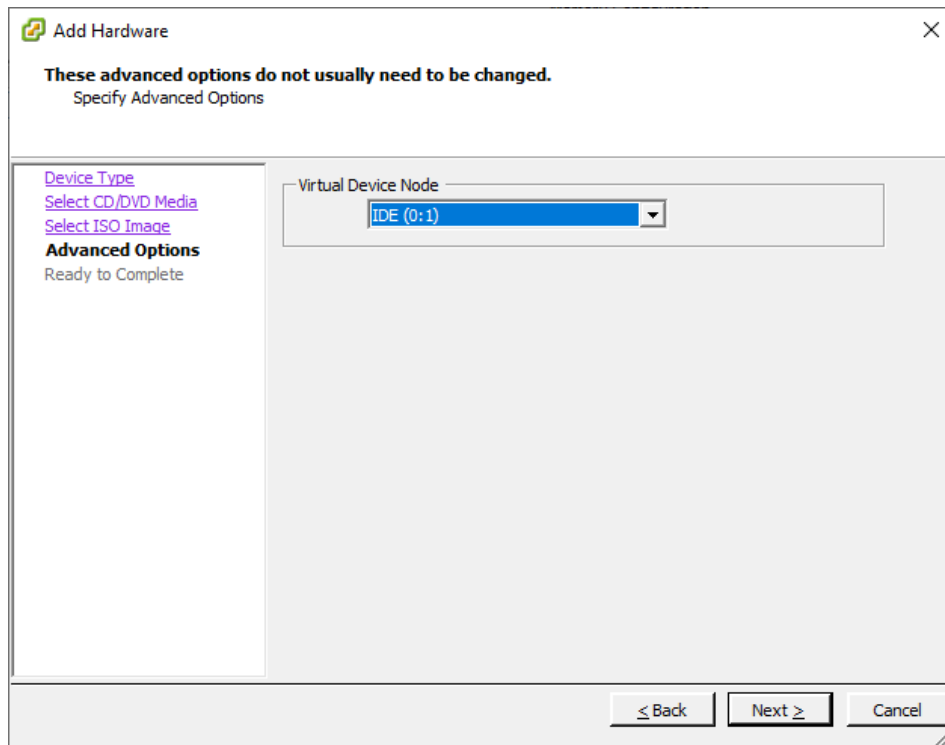
[Disk1] ISOs/configdrive.iso [Browse...]

Device Status

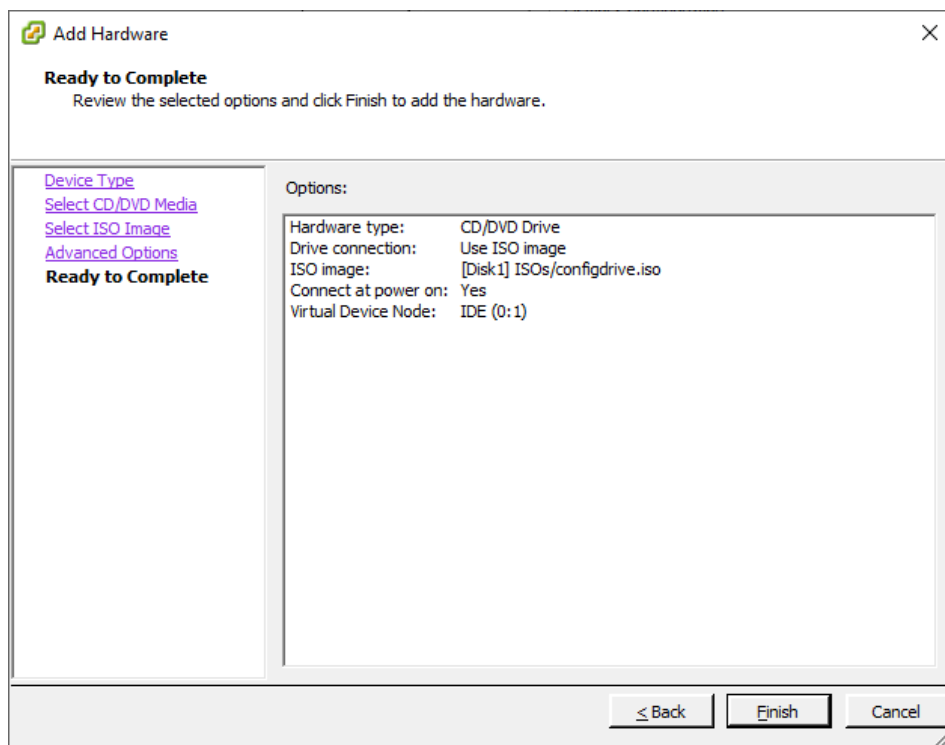
Connect at power on

[< Back] [Next >] [Cancel]

- On the “Advanced Options” page, keep the proposed virtual device node and click Next >.



- On the “Ready to Complete” page, review the details of the additional CD/DVD device and click Finish.



Then, the virtual|engine instance needs to be powered on or rebooted (depending on its current state) to use the config drive.

On KVM

1. After the virtual|engine instance has been deployed on KVM, you can attach the config drive to it by updating its libvirt XML definition:

```
$ virsh edit [instance_name]
```

2. Then, add the following code to the <devices> section:

```
<disk type='file' device='cdrom'>
  <driver name='qemu' type='raw' cache='none' />
  <source file='#path_to_configdrive.iso#' />
  <target dev='hdc' bus='ide' />
  <readonly />
</disk>
```

3. Finally, start or reboot the virtual|engine instance:

```
$ virsh start [instance_name]
```

Or

```
$ virsh reboot [instance_name]
```

License and Serial Number of the SD-WAN virtual|engine

Like physical SD-WAN ip|engines, virtual|engines are not provided locally with a license. However, since virtual|engines are managed by the SD-WAN Cloud-based Orchestrator, they require a Serial Number to be identified by this Orchestrator.

After the virtual|engine instance has been created and started, its Serial Number must be retrieved and communicated to Extreme Networks.

Retrieving the Serial Number on VMware ESXi

There are several ways to retrieve the serial number of a virtual|engine instance that is deployed on VMware ESXi.

On the virtual|engine

You may access the virtual|engine, either through SSH or the virtual console (this last mode is described hereafter).

1. Select the Console tab of the virtual|engine instance and press Enter to activate the virtual console.
2. At the login prompt, you should be able to log in (default credentials are `ipanema/ipanema`).
3. Then, run the following command:

```
[ipe]$ version
```

The virtual|engine serial number is reported as S/N in the command output.

In vSphere Client

1. Select the VMware ESXi host and choose tab “Virtual Machines”.

The virtual|engine serial number is the UUID of the corresponding virtual machine (column UUID in the displayed table).

In ESXi Shell

1. Connect to the VMware ESXi host through SSH (for example) and run the following command:

```
[root@esxihost:~] esxcli vm process list
```

The virtual|engine serial number is the UUID information of the corresponding virtual machine that is reported in the command output.

Retrieving the Serial Number on KVM

There are several ways to retrieve the serial number of a virtual|engine instance that is deployed on a KVM host.

On the virtual|engine

You may access the virtual|engine, either through SSH or the virtual console (this last mode is described hereafter).

1. Run the command below on the Linux KVM host:

```
$ virsh console [instance_name]
```

2. At the login prompt, you should be able to log in (default credentials are `ipanema/ipanema`).

3. Then, run the following command:

```
[ipe]$ version
```

The virtual|engine serial number is reported as S/N in the command output.

On Linux KVM host

1. On the Linux KVM host, run the following command:

```
$ virsh dominfo [instance_name]
```

The virtual|engine serial number is the UUID information that is reported in the command output.

Appendix

Here is a complete example of configuration text file with a JSON structure – needed in [Using config drive with the virtual|engine](#).

```
{
  "configuration": {
    "net": {
      "mode": "multipath",
      "gateway": "10.20.0.2",
      "address": "172.18.3.109",
      "mask": "255.255.255.0",
      "vlan_id": "none",
      "mtu": 1500,
      "swmacsrc": "own",
      "hostname": "myname",
      "dns_servers": [
        "10.48.26.1",
        "10.48.26.2"
      ],
      "dns_search": [
        "iv.local"
      ],
      "routes": [
        {
          "net": "10.170.16.0",
          "netmask": "255.255.255.0",
          "gw": "10.170.255.252"
        }
      ]
    },
    "interfaces": [
      {
        "name": "wan1",
        "mode": "bridge"
      },
      {
        "name": "wan2",
        "mode": "router",
        "address": "10.17.16.1",
        "mask": "255.255.0.0",
        "gateway": "10.17.255.252",
        "dns_server": "10.108.26.1"
      },
      {
        "name": "wan3",
        "mode": "router",
        "address": "10.18.16.1",
        "mask": "255.255.0.0",
        "gateway": "10.18.255.252",
        "dns_server": "10.109.26.1"
      }
    ],
    "account": {
      "enc_password":
"$6$mylongsalt$PzU3PzatX8uxBxweuHoPQErdQsjDja/pqbvMNPXbY/vc2PsFzRbpWu5I9HpBP0XPvonQO
yc.0ql2UORCNu3C1x0"
    }
  }
}
```

Copyright © 2022 Extreme Networks, Inc. All rights reserved.

Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

Trademarks

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners. For additional information on Extreme Networks trademarks, see:

<http://www.extremenetworks.com/company/legal/trademarks>

Open Source Declarations

Some software files have been licensed under certain open source or third-party licenses. End- user license agreements and open source declarations can be found at:

<https://www.extremenetworks.com/support/policies/open-source-declaration/>