Red Hat Enterprise Linux 9

Deploying RHEL 9 on Microsoft Azure

Obtaining RHEL system images and creating RHEL instances on Azure

Last Updated: 2024-06-25

Red Hat Enterprise Linux 9 Deploying RHEL 9 on Microsoft Azure

Obtaining RHEL system images and creating RHEL instances on Azure

Legal Notice

Abstract

To use Red Hat Enterprise Linux (RHEL) in a public cloud environment, you can create and deploy RHEL system images on various cloud platforms, including Microsoft Azure. You can also create and configure a Red Hat High Availability (HA) cluster on Azure. The following chapters provide instructions for creating cloud RHEL instances and HA clusters on Azure. These processes include installing the required packages and agents, configuring fencing, and installing network resource agents.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION 3

CHAPTER 1. INTRODUCING RHEL ON PUBLIC CLOUD PLATFORMS 4
1.1. BENEFITS OF USING RHEL IN A PUBLIC CLOUD 4
1.2. PUBLIC CLOUD USE CASES FOR RHEL 5
1.3. FREQUENT CONCERNS WHEN MIGRATING TO A PUBLIC CLOUD 5
1.4. OBTAINING RHEL FOR PUBLIC CLOUD DEPLOYMENTS 6
1.5. METHODS FOR CREATING RHEL CLOUD INSTANCES 7

CHAPTER 2. PUSHING VHD IMAGES TO MICROSOFT AZURE CLOUD 8

CHAPTER 3. DEPLOYING A RED HAT ENTERPRISE LINUX IMAGE AS A VIRTUAL MACHINE ON MICROSOFT AZURE 11
3.1. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON AZURE 11
3.2. UNDERSTANDING BASE IMAGES 12
3.2.1. Using a custom base image 12
3.2.2. Required system packages 12
3.2.3. Azure VM configuration settings 13
3.2.4. Creating a base image from an ISO image 13
3.3. CONFIGURING A CUSTOM BASE IMAGE FOR MICROSOFT AZURE 15
3.3.1. Installing Hyper-V device drivers 15
3.3.2. Making configuration changes required for a Microsoft Azure deployment 16
3.4. CONVERTING THE IMAGE TO A FIXED VHD FORMAT 19
3.5. INSTALLING THE AZURE CLI 20
3.6. CREATING RESOURCES IN AZURE 21
3.7. UPLOADING AND CREATING AN AZURE IMAGE 24
3.8. CREATING AND STARTING THE VM IN AZURE 25
3.9. OTHER AUTHENTICATION METHODS 26
3.10. ATTACHING RED HAT SUBSCRIPTIONS 27
3.11. SETTING UP AUTOMATIC REGISTRATION ON AZURE GOLD IMAGES 27
3.12. CONFIGURING KDUMP FOR MICROSOFT AZURE INSTANCES 28
3.13. ADDITIONAL RESOURCES 31

CHAPTER 4. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER ON MICROSOFT AZURE 32
4.1. THE BENEFITS OF USING HIGH-AVAILABILITY CLUSTERS ON PUBLIC CLOUD PLATFORMS 32
4.2. CREATING RESOURCES IN AZURE 33
4.3. REQUIRED SYSTEM PACKAGES FOR HIGH AVAILABILITY 37
4.4. AZURE VM CONFIGURATION SETTINGS 37
4.5. INSTALLING HYPER-V DEVICE DRIVERS 38
4.6. MAKING CONFIGURATION CHANGES REQUIRED FOR A MICROSOFT AZURE DEPLOYMENT 39
4.7. CREATING AN AZURE ACTIVE DIRECTORY APPLICATION 42
4.8. CONVERTING THE IMAGE TO A FIXED VHD FORMAT 44
4.9. UPLOADING AND CREATING AN AZURE IMAGE 45
4.10. INSTALLING RED HAT HA PACKAGES AND AGENTS 46
4.11. CREATING A CLUSTER 47
4.12. FENCING OVERVIEW 48
4.13. CREATING A FENCING DEVICE 49
4.14. CREATING AN AZURE INTERNAL LOAD BALANCER 51
4.15. CONFIGURING THE LOAD BALANCER RESOURCE AGENT 52
4.16. CONFIGURING SHARED BLOCK STORAGE 53
4.17. ADDITIONAL RESOURCES 57

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. INTRODUCING RHEL ON PUBLIC CLOUD PLATFORMS

Public cloud platforms provide computing resources as a service. Instead of using on-premises hardware, you can run your IT workloads, including Red Hat Enterprise Linux (RHEL) systems, as public cloud instances.

1.1. BENEFITS OF USING RHEL IN A PUBLIC CLOUD

RHEL as a cloud instance located on a public cloud platform has the following benefits over RHEL on-premises physical systems or virtual machines (VMs):

- **Flexible and fine-grained allocation of resources**
  A cloud instance of RHEL runs as a VM on a cloud platform, which typically means a cluster of remote servers maintained by the provider of the cloud service. Therefore, allocating hardware resources to the instance, such as a specific type of CPU or storage, happens on the software level and is easily customizable.

  In comparison to a local RHEL system, you are also not limited by the capabilities of your physical host. Instead, you can choose from a variety of features, based on selection offered by the cloud provider.

- **Space and cost efficiency**
  You do not need to own any on-premises servers to host your cloud workloads. This avoids the space, power, and maintenance requirements associated with physical hardware.

  Instead, on public cloud platforms, you pay the cloud provider directly for using a cloud instance. The cost is typically based on the hardware allocated to the instance and the time you spend using it. Therefore, you can optimize your costs based on your requirements.

- **Software-controlled configurations**
  The entire configuration of a cloud instance is saved as data on the cloud platform, and is controlled by software. Therefore, you can easily create, remove, clone, or migrate the instance. A cloud instance is also operated remotely in a cloud provider console and is connected to remote storage by default.

  In addition, you can back up the current state of a cloud instance as a snapshot at any time. Afterwards, you can load the snapshot to restore the instance to the saved state.

- **Separation from the host and software compatibility**
  Similarly to a local VM, the RHEL guest operating system on a cloud instance runs on a virtualized kernel. This kernel is separate from the host operating system and from the *client* system that you use to connect to the instance.

  Therefore, any operating system can be installed on the cloud instance. This means that on a RHEL public cloud instance, you can run RHEL-specific applications that cannot be used on your local operating system.

  In addition, even if the operating system of the instance becomes unstable or is compromised, your client system is not affected in any way.

Additional resources

- What is public cloud?

- What is a hyperscaler?
- Types of cloud computing
- Public cloud use cases for RHEL
- Obtaining RHEL for public cloud deployments

1.2. PUBLIC CLOUD USE CASES FOR RHEL

Deploying on a public cloud provides many benefits, but might not be the most efficient solution in every scenario. If you are evaluating whether to migrate your RHEL deployments to the public cloud, consider whether your use case will benefit from the advantages of the public cloud.

Beneficial use cases

- Deploying public cloud instances is very effective for flexibly increasing and decreasing the active computing power of your deployments, also known as *scaling up* and *scaling down*. Therefore, using RHEL on public cloud is recommended in the following scenarios:
  - Clusters with high peak workloads and low general performance requirements. Scaling up and down based on your demands can be highly efficient in terms of resource costs.
  - Quickly setting up or expanding your clusters. This avoids high upfront costs of setting up local servers.
- Cloud instances are not affected by what happens in your local environment. Therefore, you can use them for backup and disaster recovery.

Potentially problematic use cases

- You are running an existing environment that cannot be adjusted. Customizing a cloud instance to fit the specific needs of an existing deployment may not be cost-effective in comparison with your current host platform.
- You are operating with a hard limit on your budget. Maintaining your deployment in a local data center typically provides less flexibility but more control over the maximum resource costs than the public cloud does.

Next steps

- Obtaining RHEL for public cloud deployments

Additional resources

- Should I migrate my application to the cloud? Here's how to decide.

1.3. FREQUENT CONCERNS WHEN MIGRATING TO A PUBLIC CLOUD

Moving your RHEL workloads from a local environment to a public cloud platform might raise concerns about the changes involved. The following are the most commonly asked questions.

Will my RHEL work differently as a cloud instance than as a local virtual machine?

In most respects, RHEL instances on a public cloud platform work the same as RHEL virtual machines on a local host, such as an on-premises server. Notable exceptions include:

- Instead of private orchestration interfaces, public cloud instances use provider-specific console interfaces for managing your cloud resources.
- Certain features, such as nested virtualization, may not work correctly. If a specific feature is critical for your deployment, check the feature's compatibility in advance with your chosen public cloud provider.

Will my data stay safe in a public cloud as opposed to a local server?

The data in your RHEL cloud instances is in your ownership, and your public cloud provider does not have any access to it. In addition, major cloud providers support data encryption in transit, which improves the security of data when migrating your virtual machines to the public cloud.

The general security of your RHEL public cloud instances is managed as follows:

- Your public cloud provider is responsible for the security of the cloud hypervisor
- Red Hat provides the security features of the RHEL guest operating systems in your instances
- You manage the specific security settings and practices in your cloud infrastructure

What effect does my geographic region have on the functionality of RHEL public cloud instances?

You can use RHEL instances on a public cloud platform regardless of your geographical location. Therefore, you can run your instances in the same region as your on-premises server.

However, hosting your instances in a physically distant region might cause high latency when operating them. In addition, depending on the public cloud provider, certain regions may provide additional features or be more cost-efficient. Before creating your RHEL instances, review the properties of the hosting regions available for your chosen cloud provider.

1.4. OBTAINING RHEL FOR PUBLIC CLOUD DEPLOYMENTS

To deploy a RHEL system in a public cloud environment, you need to:

1. Select the optimal cloud provider for your use case, based on your requirements and the current offer on the market.
   The cloud providers currently certified for running RHEL instances are:
   - Amazon Web Services (AWS)
   - Google Cloud Platform (GCP)
   - Microsoft Azure

   NOTE

   This document specifically talks about deploying RHEL on Microsoft Azure.
2. Create a RHEL cloud instance on your chosen cloud platform. For more information, see Methods for creating RHEL cloud instances .
3. To keep your RHEL deployment up-to-date, use Red Hat Update Infrastructure (RHUI).

Additional resources

- RHUI documentation
- Red Hat Open Hybrid Cloud

1.5. METHODS FOR CREATING RHEL CLOUD INSTANCES

To deploy a RHEL instance on a public cloud platform, you can use one of the following methods:

Create a system image of RHEL and import it to the cloud platform.

- To create the system image, you can use the RHEL image builder or you can build the image manually.
- This method uses your existing RHEL subscription, and is also referred to as *bring your own subscription* (BYOS).
- You pre-pay a yearly subscription, and you can use your Red Hat customer discount.
- Your customer service is provided by Red Hat.
- For creating multiple images effectively, you can use the **cloud-init** tool.

Purchase a RHEL instance directly from the cloud provider marketplace.

- You post-pay an hourly rate for using the service. Therefore, this method is also referred to as *pay as you go* (PAYG).
- Your customer service is provided by the cloud platform provider.

NOTE

For detailed instructions on using various methods to deploy RHEL instances On Microsoft Azure, see the following chapters in this document.

Additional resources

- What is a golden image?
- Configuring and managing cloud-init for RHEL 9

CHAPTER 2. PUSHING VHD IMAGES TO MICROSOFT AZURE CLOUD

You can create **.vhd** images by using RHEL image builder. Then, you can push the output **.vhd** images to a Blob Storage of the Microsoft Azure Cloud service provider.

Prerequisites

- You have root access to the system.
- You have access to the RHEL image builder interface of the RHEL web console.
- You created a blueprint. See Creating a RHEL image builder blueprint in the web console interface.
- You have a Microsoft Storage Account created.
- You have a writable Blob Storage prepared.

Procedure

1. In the RHEL image builder dashboard, select the blueprint you want to use.
2. Click the **Images** tab.
3. Click **Create Image** to create your customized **.vhd** image.
   The **Create image** wizard opens.
   a. Select **Microsoft Azure (.vhd)** from the **Type** drop-down menu list.
   b. Check the **Upload to Azure** checkbox to upload your image to the Microsoft Azure Cloud.
   c. Enter the **Image Size** and click **Next**.
4. On the **Upload to Azure** page, enter the following information:
   a. On the Authentication page, enter:
      i. Your **Storage account** name. You can find it on the **Storage account** page, in the Microsoft Azure portal.
      ii. Your **Storage access key**. You can find it on the **Access Key** Storage page.
      iii. Click **Next**.
   b. On the **Authentication** page, enter:
      i. The image name.
      ii. The **Storage container**. It is the blob container to which you will upload the image. Find it under the **Blob service** section, in the Microsoft Azure portal.
      iii. Click **Next**.
5. On the **Review** page, click **Create**. The RHEL image builder and upload processes start.
   Access the image you pushed into **Microsoft Azure Cloud**.

6. Access the Microsoft Azure portal.
7. In the search bar, type "storage account" and click **Storage accounts** from the list.
8. On the search bar, type "Images" and select the first entry under **Services**. You are redirected to the **Image dashboard**.
9. On the navigation panel, click **Containers**.
10. Find the container you created. Inside the container is the **.vhd** file you created and pushed by using RHEL image builder.

Verification

1. Verify that you can create a VM image and launch it.
    a. In the search bar, type images account and click **Images** from the list.
    b. Click **+Create**.
    c. From the dropdown list, choose the resource group you used earlier.
    d. Enter a name for the image.
    e. For the **OS type**, select **Linux**.
    f. For the **VM generation**, select **Gen 2**.
    g. Under **Storage Blob**, click **Browse** and click through the storage accounts and container until you reach your VHD file.
    h. Click **Select** at the end of the page.
    i. Choose an Account Type, for example, **Standard SSD**.
    j. Click **Review + Create** and then **Create**. Wait a few moments for the image creation.
2. To launch the VM, follow the steps:
    a. Click **Go to resource**.
    b. Click **+ Create VM** from the menu bar on the header.
    c. Enter a name for your virtual machine.
    d. Complete the **Size** and **Administrator account** sections.
    e. Click **Review + Create** and then **Create**. You can see the deployment progress. After the deployment finishes, click the virtual machine name to retrieve the public IP address of the instance to connect by using SSH.
    f. Open a terminal to create an SSH connection to connect to the VM.

Additional resources

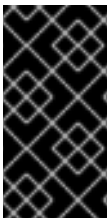
- Microsoft Azure Storage Documentation .

- Create a Microsoft Azure Storage account .
- Open a case on Red Hat Customer Portal .
- Help + support.
- Contacting Red Hat.

CHAPTER 3. DEPLOYING A RED HAT ENTERPRISE LINUX IMAGE AS A VIRTUAL MACHINE ON MICROSOFT AZURE

To deploy a Red Hat Enterprise Linux 9 (RHEL 9) image on Microsoft Azure, follow the information below. This chapter:

- Discusses your options for choosing an image
- Lists or refers to system requirements for your host system and virtual machine (VM)
- Provides procedures for creating a custom VM from an ISO image, uploading it to Azure, and launching an Azure VM instance



IMPORTANT

You can create a custom VM from an ISO image, but Red Hat recommends that you use the *Red Hat Image Builder* product to create customized images for use on specific cloud providers. With Image Builder, you can create and upload an Azure Disk Image (VHD format). See Composing a Customized RHEL System Image for more information.

For a list of Red Hat products that you can use securely on Azure, refer to Red Hat on Microsoft Azure.

Prerequisites

- Sign up for a Red Hat Customer Portal account.
- Sign up for a Microsoft Azure account.

3.1. RED HAT ENTERPRISE LINUX IMAGE OPTIONS ON AZURE

The following table lists image choices for RHEL 9 on Microsoft Azure, and notes the differences in the image options.

Table 3.1. Image options

Image option	Subscriptions	Sample scenario	Considerations
Deploy a Red Hat Gold Image.	Use your existing Red Hat subscriptions.	Select a Red Hat Gold Image on Azure. For details on Gold Images and how to access them on Azure, see the Red Hat Cloud Access Reference Guide.	The subscription includes the Red Hat product cost; you pay Microsoft for all other instance costs.
Deploy a custom image that you move to Azure.	Use your existing Red Hat subscriptions.	Upload your custom image and attach your subscriptions.	The subscription includes the Red Hat product cost; you pay Microsoft for all other instance costs.

Image option	Subscriptions	Sample scenario	Considerations
Deploy an existing Azure image that includes RHEL.	The Azure images include a Red Hat product.	Choose a RHEL image when you create a VM by using the Azure console, or choose a VM from the Azure Marketplace.	You pay Microsoft hourly on a *pay-as-you-go* model. Such images are called "on-demand." Azure provides support for on-demand images through a support agreement. Red Hat provides updates to the images. Azure makes the updates available through the Red Hat Update Infrastructure (RHUI).

Additional resources

- Using Red Hat Gold Images on Microsoft Azure
- Azure Marketplace
- Billing options in the Azure Marketplace
- Red Hat Enterprise Linux Bring-Your-Own-Subscription Gold Images in Azure
- Red Hat Cloud Access Reference Guide

3.2. UNDERSTANDING BASE IMAGES

This section includes information about using preconfigured base images and their configuration settings.

3.2.1. Using a custom base image

To manually configure a virtual machine (VM), first create a base (starter) VM image. Then, you can modify configuration settings and add the packages the VM requires to operate on the cloud. You can make additional configuration changes for your specific application after you upload the image.

To prepare a cloud image of RHEL, follow the instructions in the sections below. To prepare a Hyper-V cloud image of RHEL, see the Prepare a Red Hat-based virtual machine from Hyper-V Manager .

3.2.2. Required system packages

To create and configure a base image of RHEL, your host system must have the following packages installed.

Table 3.2. System packages

Package	Repository	Description
libvirt	rhel-9-for-x86_64-appstream-rpms	Open source API, daemon, and management tool for managing platform virtualization
virt-install	rhel-9-for-x86_64-appstream-rpms	A command-line utility for building VMs
libguestfs	rhel-9-for-x86_64-appstream-rpms	A library for accessing and modifying VM file systems
guestfs-tools	rhel-9-for-x86_64-appstream-rpms	System administration tools for VMs; includes the **virt-customize** utility

3.2.3. Azure VM configuration settings

Azure VMs must have the following configuration settings. Some of these settings are enabled during the initial VM creation. Other settings are set when provisioning the VM image for Azure. Keep these settings in mind as you move through the procedures. Refer to them as necessary.

Table 3.3. VM configuration settings

Setting	Recommendation
ssh	ssh must be enabled to provide remote access to your Azure VMs.
dhcp	The primary virtual adapter should be configured for dhcp (IPv4 only).
Swap Space	Do not create a dedicated swap file or swap partition. You can configure swap space with the Windows Azure Linux Agent (WALinuxAgent).
NIC	Choose **virtio** for the primary virtual network adapter.
encryption	For custom images, use Network Bound Disk Encryption (NBDE) for full disk encryption on Azure.

3.2.4. Creating a base image from an ISO image

The following procedure lists the steps and initial configuration requirements for creating a custom ISO image. Once you have configured the image, you can use the image as a template for creating additional VM instances.

Prerequisites

- Ensure that you have enabled your host machine for virtualization. See Enabling virtualization in RHEL 9 for information and procedures.

Procedure

1. Download the latest Red Hat Enterprise Linux 9 DVD ISO image from the Red Hat Customer Portal.
2. Create and start a basic Red Hat Enterprise Linux VM. For instructions, see Creating virtual machines.
   a. If you use the command line to create your VM, ensure that you set the default memory and CPUs to the capacity you want for the VM. Set your virtual network interface to **virtio**. For example, the following command creates a **kvmtest** VM by using the **rhel-9.0-x86_64-kvm.qcow2** image:

```
# virt-install \
    --name kvmtest --memory 2048 --vcpus 2 \
    --disk rhel-9.0-x86_64-kvm.qcow2,bus=virtio \
    --import --os-variant=rhel9.0
```

   b. If you use the web console to create your VM, follow the procedure in Creating virtual machines using the web console, with these caveats:
      - Do not check **Immediately Start VM**.
      - Change your **Memory** size to your preferred settings.
      - Before you start the installation, ensure that you have changed **Model** under **Virtual Network Interface Settings** to **virtio** and change your **vCPUs** to the capacity settings you want for the VM.
3. Review the following additional installation selection and modifications.
   - Select **Minimal Install** with the **standard RHEL** option.
   - For **Installation Destination**, select **Custom Storage Configuration**. Use the following configuration information to make your selections.
     - Verify at least 500 MB for **/boot**.
     - For file system, use xfs, ext4, or ext3 for both **boot** and **root** partitions.
     - Remove swap space. Swap space is configured on the physical blade server in Azure by the WALinuxAgent.
   - On the **Installation Summary** screen, select **Network and Host Name**. Switch **Ethernet** to **On**.
4. When the install starts:
   - Create a **root** password.
   - Create an administrative user account.
5. When installation is complete, reboot the VM and log in to the root account.
6. Once you are logged in as **root**, you can configure the image.

3.3. CONFIGURING A CUSTOM BASE IMAGE FOR MICROSOFT AZURE

To deploy a RHEL 9 virtual machine (VM) with specific settings in Azure, you can create a custom base image for the VM. The following sections describe additional configuration changes that Azure requires.

3.3.1. Installing Hyper-V device drivers

Microsoft provides network and storage device drivers as part of their Linux Integration Services (LIS) for Hyper-V package. You may need to install Hyper-V device drivers on the VM image prior to provisioning it as an Azure virtual machine (VM). Use the **lsinitrd | grep hv** command to verify that the drivers are installed.

Procedure

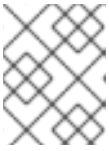
1. Enter the following **grep** command to determine if the required Hyper-V device drivers are installed.

```
# lsinitrd | grep hv
```

In the example below, all required drivers are installed.

```
# lsinitrd | grep hv
drwxr-xr-x   2 root     root            0 Aug 12 14:21 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root        31272 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/hv/hv_vmbus.ko.xz
-rw-r--r--   1 root     root        25132 Aug 11 08:46 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/net/hyperv/hv_netvsc.ko.xz
-rw-r--r--   1 root     root         9796 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/scsi/hv_storvsc.ko.xz
```

If all the drivers are not installed, complete the remaining steps.



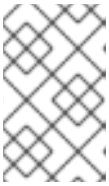
NOTE

An **hv_vmbus** driver may exist in the environment. Even if this driver is present, complete the following steps.

2. Create a file named **hv.conf** in **/etc/dracut.conf.d**.

3. Add the following driver parameters to the **hv.conf** file.

```
add_drivers+=" hv_vmbus "
add_drivers+=" hv_netvsc "
add_drivers+=" hv_storvsc "
add_drivers+=" nvme "
```



NOTE

Note the spaces before and after the quotes, for example, **add_drivers+=" hv_vmbus "**. This ensures that unique drivers are loaded in the event that other Hyper-V drivers already exist in the environment.

4. Regenerate the **initramfs** image.

```
# dracut -f -v --regenerate-all
```

Verification

1. Reboot the machine.
2. Run the **lsinitrd | grep hv** command to verify that the drivers are installed.

3.3.2. Making configuration changes required for a Microsoft Azure deployment

Before you deploy your custom base image to Azure, you must perform additional configuration changes to ensure that the virtual machine (VM) can properly operate in Azure.

Procedure

1. Log in to the VM.
2. Register the VM, and enable the Red Hat Enterprise Linux 9 repository.

```
# subscription-manager register --auto-attach
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```

3. Ensure that the **cloud-init** and **hyperv-daemons** packages are installed.

```
# dnf install cloud-init hyperv-daemons -y
```

4. Create **cloud-init** configuration files that are needed for integration with Azure services:
   a. To enable logging to the Hyper-V Data Exchange Service (KVP), create the **/etc/cloud/cloud.cfg.d/10-azure-kvp.cfg** configuration file and add the following lines to that file.

```
reporting:
    logging:
        type: log
    telemetry:
        type: hyperv
```

   b. To add Azure as a datasource, create the **/etc/cloud/cloud.cfg.d/91-azure_datasource.cfg** configuration file, and add the following lines to that file.

```
datasource_list: [ Azure ]
datasource:
    Azure:
        apply_network_config: False
```

5. To ensure that specific kernel modules are blocked from loading automatically, edit or create the **/etc/modprobe.d/blocklist.conf** file and add the following lines to that file.

```
blacklist nouveau
blacklist lbm-nouveau
blacklist floppy
blacklist amdgpu
blacklist skx_edac
blacklist intel_cstate
```

6. Modify **udev** network device rules:

   a. Remove the following persistent network device rules if present.

   ```
   # rm -f /etc/udev/rules.d/70-persistent-net.rules
   # rm -f /etc/udev/rules.d/75-persistent-net-generator.rules
   # rm -f /etc/udev/rules.d/80-net-name-slot-rules
   ```

   b. To ensure that Accelerated Networking on Azure works as intended, create a new network device rule **/etc/udev/rules.d/68-azure-sriov-nm-unmanaged.rules** and add the following line to it.

   ```
   SUBSYSTEM=="net", DRIVERS=="hv_pci", ACTION=="add",
   ENV{NM_UNMANAGED}="1"
   ```

7. Set the **sshd** service to start automatically.

   ```
   # systemctl enable sshd
   # systemctl is-enabled sshd
   ```

8. Modify kernel boot parameters:

   a. Open the **/etc/default/grub** file, and ensure the **GRUB_TIMEOUT** line has the following value.

   ```
   GRUB_TIMEOUT=10
   ```

   b. Remove the following options from the end of the **GRUB_CMDLINE_LINUX** line if present.

   ```
   rhgb quiet
   ```

   c. Ensure the **/etc/default/grub** file contains the following lines with all the specified options.

   ```
   GRUB_CMDLINE_LINUX="loglevel=3 crashkernel=auto console=tty1 console=ttyS0
   earlyprintk=ttyS0 rootdelay=300"
   GRUB_TIMEOUT_STYLE=countdown
   GRUB_TERMINAL="serial console"
   GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --
   stop=1"
   ```

   d. Regenerate the **grub.cfg** file.

   - On a BIOS-based machine:
     - In RHEL 9.2 and earlier:

     ```
     # grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

- In RHEL 9.3 and later:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

- On a UEFI-based machine:
  - In RHEL 9.2 and earlier:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

  - In RHEL 9.3 and later:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg --update-bls-cmdline
```

    If your system uses a non-default location for **grub.cfg**, adjust the command accordingly.

9. Configure the Windows Azure Linux Agent (**WALinuxAgent**):
   a. Install and enable the **WALinuxAgent** package.

```
# dnf install WALinuxAgent -y
# systemctl enable waagent
```

   b. To ensure that a swap partition is not used in provisioned VMs, edit the following lines in the **/etc/waagent.conf** file.

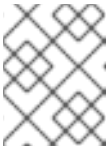
```
Provisioning.DeleteRootPassword=y
ResourceDisk.Format=n
ResourceDisk.EnableSwap=n
```

10. Prepare the VM for Azure provisioning:
    a. Unregister the VM from Red Hat Subscription Manager.

```
# subscription-manager unregister
```

    b. Clean up the existing provisioning details.

```
# waagent -force -deprovision
```



NOTE

This command generates warnings, which are expected because Azure handles the provisioning of VMs automatically.

    c. Clean the shell history and shut down the VM.

```
# export HISTSIZE=0
# poweroff
```

3.4. CONVERTING THE IMAGE TO A FIXED VHD FORMAT

All Microsoft Azure VM images must be in a fixed **VHD** format. The image must be aligned on a 1 MB boundary before it is converted to VHD. To convert the image from **qcow2** to a fixed **VHD** format and align the image, see the following procedure. Once you have converted the image, you can upload it to Azure.

Procedure

1. Convert the image from **qcow2** to **raw** format.

```
$ qemu-img convert -f qcow2 -O raw <image-name>.qcow2 <image-name>.raw
```

2. Create a shell script with the following content.

```
#!/bin/bash
MB=$((1024 * 1024))
size=$(qemu-img info -f raw --output json "$1" | gawk 'match($0, /"virtual-size": ([0-9]+),/, val)
{print val[1]}')
rounded_size=$((($size/$MB + 1) * $MB))
if [ $(($size % $MB)) -eq  0 ]
then
 echo "Your image is already aligned. You do not need to resize."
 exit 1
fi
echo "rounded size = $rounded_size"
export rounded_size
```

3. Run the script. This example uses the name **align.sh**.

```
$ sh align.sh <image-xxx>.raw
```

   - If the message *"Your image is already aligned. You do not need to resize."* displays, proceed to the following step.
   - If a value displays, your image is not aligned.

4. Use the following command to convert the file to a fixed **VHD** format.
   The sample uses qemu-img version 2.12.0.

```
$ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
<image.xxx>.vhd
```

   Once converted, the **VHD** file is ready to upload to Azure.

5. If the **raw** image is not aligned, complete the following steps to align it.

   a. Resize the **raw** file by using the rounded value displayed when you ran the verification script.

```
$ qemu-img resize -f raw <image-xxx>.raw <rounded-value>
```

   b. Convert the **raw** image file to a **VHD** format.
   The sample uses qemu-img version 2.12.0.

```
$ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
<image.xxx>.vhd
```

Once converted, the **VHD** file is ready to upload to Azure.

3.5. INSTALLING THE AZURE CLI

Complete the following steps to install the Azure command line interface (Azure CLI 2.1). Azure CLI 2.1 is a Python-based utility that creates and manages VMs in Azure.

Prerequisites

- You need to have an account with Microsoft Azure before you can use the Azure CLI.
- The Azure CLI installation requires Python 3.x.

Procedure

1. Import the Microsoft repository key.

```
$ sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

2. Create a local Azure CLI repository entry.

```
$ sudo sh -c 'echo -e "[azure-cli]\nname=Azure
CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
cli\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/yum.repos.d/azure-cli.repo'
```

3. Update the **dnf** package index.

```
$ dnf check-update
```

4. Check your Python version (**python --version**) and install Python 3.x, if necessary.

```
$ sudo dnf install python3
```

5. Install the Azure CLI.

```
$ sudo dnf install -y azure-cli
```

6. Run the Azure CLI.

```
$ az
```

Additional resources

- Azure CLI
- Azure CLI command reference

3.6. CREATING RESOURCES IN AZURE

Complete the following procedure to create the Azure resources that you need before you can upload the **VHD** file and create the Azure image.

Procedure

1. Authenticate your system with Azure and log in.

```
$ az login
```

NOTE

If a browser is available in your environment, the CLI opens your browser to the Azure sign-in page. See Sign in with Azure CLI for more information and options.

2. Create a resource group in an Azure region.

```
$ az group create --name <resource-group> --location <azure-region>
```

Example:

```
[clouduser@localhost]$ az group create --name azrhelclirsgrp --location southcentralus
{
  "id": "/subscriptions//resourceGroups/azrhelclirsgrp",
  "location": "southcentralus",
  "managedBy": null,
  "name": "azrhelclirsgrp",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

3. Create a storage account. See SKU Types for more information about valid SKU values.

```
$ az storage account create -l <azure-region> -n <storage-account-name> -g <resource-group> --sku <sku_type>
```

Example:

```
[clouduser@localhost]$ az storage account create -l southcentralus -n azrhelclistact -g azrhelclirsgrp --sku Standard_LRS
{
  "accessTier": null,
  "creationTime": "2017-04-05T19:10:29.855470+00:00",
  "customDomain": null,
  "encryption": null,
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Storage/storageAccounts/azrhelclistact",
  "kind": "StorageV2",
  "lastGeoFailoverTime": null,
```

```
  "location": "southcentralus",
  "name": "azrhelclistact",
  "primaryEndpoints": {
    "blob": "https://azrhelclistact.blob.core.windows.net/",
    "file": "https://azrhelclistact.file.core.windows.net/",
    "queue": "https://azrhelclistact.queue.core.windows.net/",
    "table": "https://azrhelclistact.table.core.windows.net/"
},
"primaryLocation": "southcentralus",
"provisioningState": "Succeeded",
"resourceGroup": "azrhelclirsgrp",
"secondaryEndpoints": null,
"secondaryLocation": null,
"sku": {
  "name": "Standard_LRS",
  "tier": "Standard"
},
"statusOfPrimary": "available",
"statusOfSecondary": null,
"tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```


4. Get the storage account connection string.

```
$ az storage account show-connection-string -n <storage-account-name> -g <resource-group>
```

Example:


```
[clouduser@localhost]$ az storage account show-connection-string -n azrhelclistact -g azrhelclirsgrp
{
  "connectionString":
"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=azrhelclistact
AccountKey=NreGk...=="
}
```


5. Export the connection string by copying the connection string and pasting it into the following command. This string connects your system to the storage account.

```
$ export AZURE_STORAGE_CONNECTION_STRING="<storage-connection-string>"
```

Example:


```
[clouduser@localhost]$ export
AZURE_STORAGE_CONNECTION_STRING="DefaultEndpointsProtocol=https;EndpointSuffi
x=core.windows.net;AccountName=azrhelclistact;AccountKey=NreGk...=="
```


6. Create the storage container.

```
$ az storage container create -n <container-name>
```

Example:

```
[clouduser@localhost]$ az storage container create -n azrhelclistcont
{
  "created": true
}
```

7. Create a virtual network.

```
$ az network vnet create -g <resource group> --name <vnet-name> --subnet-name <subnet-name>
```

Example:

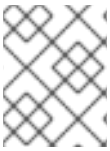
```
[clouduser@localhost]$ az network vnet create --resource-group azrhelclirsgrp --name azrhelclivnet1 --subnet-name azrhelclisubnet1
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
      "10.0.0.0/16"
      ]
  },
  "dhcpOptions": {
    "dnsServers": []
  },
  "etag": "W/\"\"",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azrhelclivnet1",
  "location": "southcentralus",
  "name": "azrhelclivnet1",
  "provisioningState": "Succeeded",
  "resourceGroup": "azrhelclirsgrp",
  "resourceGuid": "0f25efee-e2a6-4abe-a4e9-817061ee1e79",
  "subnets": [
    {
      "addressPrefix": "10.0.0.0/24",
      "etag": "W/\"\"",
      "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azrhelclivnet1/subnets/azrhelclisubnet1",
      "ipConfigurations": null,
      "name": "azrhelclisubnet1",
      "networkSecurityGroup": null,
      "provisioningState": "Succeeded",
      "resourceGroup": "azrhelclirsgrp",
      "resourceNavigationLinks": null,
      "routeTable": null
    }
  ],
  "tags": {},
  "type": "Microsoft.Network/virtualNetworks",
  "virtualNetworkPeerings": null
  }
}
```

Additional resources

- Azure Managed Disks Overview
- SKU Types

3.7. UPLOADING AND CREATING AN AZURE IMAGE

Complete the following steps to upload the **VHD** file to your container and create an Azure custom image.



NOTE

The exported storage connection string does not persist after a system reboot. If any of the commands in the following steps fail, export the connection string again.

Procedure

1. Upload the **VHD** file to the storage container. It may take several minutes. To get a list of storage containers, enter the **az storage container list** command.

```
$ az storage blob upload \
    --account-name <storage-account-name> --container-name <container-name> \
    --type page --file <path-to-vhd> --name <image-name>.vhd
```

Example:

```
[clouduser@localhost]$ az storage blob upload \
--account-name azrhelclistact --container-name azrhelclistcont \
--type page --file rhel-image-{ProductNumber}.vhd --name rhel-image-{ProductNumber}.vhd

Percent complete: %100.0
```

2. Get the URL for the uploaded **VHD** file to use in the following step.

```
$ az storage blob url -c <container-name> -n <image-name>.vhd
```

Example:

```
$ az storage blob url -c azrhelclistcont -n rhel-image-9.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.vhd"
```

3. Create the Azure custom image.

```
$ az image create -n <image-name> -g <resource-group> -l <azure-region> --source <URL>
--os-type linux
```

NOTE

The default hypervisor generation of the VM is V1. You can optionally specify a V2 hypervisor generation by including the option **--hyper-v-generation V2**. Generation 2 VMs use a UEFI-based boot architecture. See Support for generation 2 VMs on Azure for information about generation 2 VMs.

The command may return the error "Only blobs formatted as VHDs can be imported." This error may mean that the image was not aligned to the nearest 1 MB boundary before it was converted to **VHD**.

Example:

```
$ az image create -n rhel9 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.vhd --os-type linux
```

3.8. CREATING AND STARTING THE VM IN AZURE

The following steps provide the minimum command options to create a managed-disk Azure VM from the image. See az vm create for additional options.

Procedure

1. Enter the following command to create the VM.

```
$ az vm create \
   -g <resource-group> -l <azure-region> -n <vm-name> \
   --vnet-name <vnet-name> --subnet <subnet-name> --size Standard_A2 \
   --os-disk-name <simple-name> --admin-username <administrator-name> \
   --generate-ssh-keys --image <path-to-image>
```

NOTE

The option **--generate-ssh-keys** creates a private/public key pair. Private and public key files are created in ~/**.ssh** on your system. The public key is added to the **authorized_keys** file on the VM for the user specified by the **--admin-username** option. See Other authentication methods for additional information.

Example:

```
[clouduser@localhost]$ az vm create \
-g azrhelclirsgrp2 -l southcentralus -n rhel-azure-vm-1 \
--vnet-name azrhelclivnet1 --subnet azrhelclisubnet1 --size Standard_A2 \
--os-disk-name vm-1-osdisk --admin-username clouduser \
--generate-ssh-keys --image rhel9

{
  "fqdns": "",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Compute/virtualMachines/rhe
l-azure-vm-1",
  "location": "southcentralus",
  "macAddress": "",
```

```
    "powerState": "VM running",
    "privateIpAddress": "10.0.0.4",
    "publicIpAddress": "<public-IP-address>",
    "resourceGroup": "azrhelclirsgrp2"
```

Note the **publicIpAddress**. You need this address to log in to the VM in the following step.

2. Start an SSH session and log in to the VM.

```
[clouduser@localhost]$ ssh -i /home/clouduser/.ssh/id_rsa clouduser@<public-IP-address>.
The authenticity of host ',<public-IP-address>' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '<public-IP-address>' (ECDSA) to the list of known hosts.

[clouduser@rhel-azure-vm-1 ~]$
```

If you see a user prompt, you have successfully deployed your Azure VM.

You can now go to the Microsoft Azure portal and check the audit logs and properties of your resources. You can manage your VMs directly in this portal. If you are managing multiple VMs, you should use the Azure CLI. The Azure CLI provides a powerful interface to your resources in Azure. Enter **az --help** in the CLI or see the Azure CLI command reference to learn more about the commands you use to manage your VMs in Microsoft Azure.

3.9. OTHER AUTHENTICATION METHODS

While recommended for increased security, using the Azure-generated key pair is not required. The following examples show two methods for SSH authentication.

Example 1: These command options provision a new VM without generating a public key file. They allow SSH authentication by using a password.

```
$ az vm create \
   -g <resource-group> -l <azure-region> -n <vm-name> \
   --vnet-name <vnet-name> --subnet <subnet-name> --size Standard_A2 \
   --os-disk-name <simple-name> --authentication-type password \
   --admin-username <administrator-name> --admin-password <ssh-password> --image <path-to-image>
```

```
$ ssh <admin-username>@<public-ip-address>
```

Example 2: These command options provision a new Azure VM and allow SSH authentication by using an existing public key file.

```
$ az vm create \
   -g <resource-group> -l <azure-region> -n <vm-name> \
   --vnet-name <vnet-name> --subnet <subnet-name> --size Standard_A2 \
   --os-disk-name <simple-name> --admin-username <administrator-name> \
   --ssh-key-value <path-to-existing-ssh-key> --image <path-to-image>
```

```
$ ssh -i <path-to-existing-ssh-key> <admin-username>@<public-ip-address>
```

3.10. ATTACHING RED HAT SUBSCRIPTIONS

Using the **subscription-manager** command, you can register and attach your Red Hat subscription to a RHEL instance.

Prerequisites

- You must have enabled your subscriptions.

Procedure

1. Register your system.

```
# subscription-manager register --auto-attach
```

2. Attach your subscriptions.
   - You can use an activation key to attach subscriptions. See Creating Red Hat Customer Portal Activation Keys for more information.
   - Alternatively, you can manually attach a subscription by using the ID of the subscription pool (Pool ID). See Attaching and Removing Subscriptions Through the Command Line .
3. **Optional:** To collect various system metrics about the instance in the Red Hat Hybrid Cloud Console, you can register the instance with Red Hat Insights.

```
# insights-client register --display-name <display-name-value>
```

   For information on further configuration of Red Hat Insights, see Client Configuration Guide for Red Hat Insights.

Additional resources

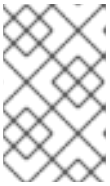
- Creating Red Hat Customer Portal Activation Keys
- Attaching and Removing Subscriptions Through the Command Line
- Using and Configuring Red Hat Subscription Manager
- Client Configuration Guide for Red Hat Insights

3.11. SETTING UP AUTOMATIC REGISTRATION ON AZURE GOLD IMAGES

To make deploying RHEL 9 virtual machines (VM) on Micorsoft Azure faster and more comfortable, you can set up Gold Images of RHEL 9 to be automatically registered to the Red Hat Subscription Manager (RHSM).

Prerequisites

- RHEL 9 Gold Images are available to you in Microsoft Azure. For instructions, see Using Gold Images on Azure.



NOTE

A Microsoft Azure account can only be attached to a single Red Hat account at a time. Therefore, ensure no other users require access to the Azure account before attaching it to your Red Hat one.

Procedure

1. Use the Gold Image to create a RHEL 9 VM in your Azure instance. For instructions, see Creating and starting the VM in Azure .
2. Start the created VM.
3. In the RHEL 9 VM, enable automatic registration.

```
# subscription-manager config --rhsmcertd.auto_registration=1
```

4. Enable the **rhsmcertd** service.

```
# systemctl enable rhsmcertd.service
```

5. Disable the **redhat.repo** repository.

```
# subscription-manager config --rhsm.manage_repos=0
```

6. Power off the VM, and save it as a managed image on Azure. For instructions, see How to create a managed image of a virtual machine or VHD.
7. Create VMs by using the managed image. They will be automatically subscribed to RHSM.

Verification

- In a RHEL 9 VM created using the above instructions, verify the system is registered to RHSM by executing the **subscription-manager identity** command. On a successfully registered system, this displays the UUID of the system. For example:

```
# subscription-manager identity
system identity: fdc46662-c536-43fb-a18a-bbcb283102b7
name: 192.168.122.222
org name: 6340056
org ID: 6340056
```

Additional resources

- Red Hat Gold Images in Azure
- Overview of RHEL images in Azure
- Configuring cloud sources for Red Hat services

3.12. CONFIGURING KDUMP FOR MICROSOFT AZURE INSTANCES

If a kernel crash occurs in a RHEL instance, you can use the **kdump** service to determine the cause of the crash. If **kdump** is configured correctly when your instance kernel terminates unexpectedly, **kdump**

generates a dump file, known as crash dump or a **vmcore** file. You can then analyze the file to find why the crash occurred and to debug your system.

For **kdump** to work on Microsoft Azure instances, you might need to adjust the **kdump** reserved memory and the **vmcore** target to fit VM sizes and RHEL versions.

Prerequisites

- You are using a Microsoft Azure environment that supports **kdump**:
  - Standard_DS2_v2 VM
  - Standard NV16as v4
  - Standard M416-208s v2
  - Standard M416ms v2
- You have **root** permissions on the system.
- Your system meets the requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets .

Procedure

1. Ensure that **kdump** and other necessary packages are installed on your system.

```
# dnf install kexec-tools
```

2. Verify that the default location for crash dump files is set in the **kdump** configuration file and that the **/var/crash** file is available.

```
# grep -v "#" /etc/kdump.conf

path /var/crash
core_collector makedumpfile -l --message-level 7 -d 31
```

3. Based on the size and version of your RHEL virtual machine (VM) instance, decide whether you need a **vmcore** target with more free space, such as **/mnt/crash**. To do so, use the following table.

Table 3.4. Virtual machine sizes that have been tested with GEN2 VM on Azure

RHEL Version	Standard DS1 v2 (1 vCPU, 3.5GiB)	Standard NV16as v4 (16 vCPUs, 56 GiB)	Standard M416-208s v2 (208 vCPUs, 5700 GiB)	Standard M416ms v2 (416 vCPUs, 11400 GiB)
RHEL 9.0 - RHEL 9.3	Default	Default	Target	Target

- **Default** indicates that **kdump** works as expected with the default memory and the default **kdump** target. The default **kdump** target is **/var/crash**.

- **Target** indicates that **kdump** works as expected with the default memory. However, you might need to assign a target with more free space.

4. If your instance requires it, assign a target with more free space, such as **/mnt/crash**. To do so, edit the **/etc/kdump.conf** file and replace the default path.

```
$ sed s/"path /var/crash"/"path /mnt/crash"
```

   The option path **/mnt/crash** represents the path to the file system in which **kdump** saves the crash dump file.

   For more options, such as writing the crash dump file to a different partition, directly to a device or storing it to a remote machine, see Configuring the kdump target.

5. If your instance requires it, increase the crash kernel size to the sufficient size for **kdump** to capture the **vmcore** by adding the respective boot parameter.
   For example, for a Standard M416-208s v2 VM, the sufficient size is 512 MB, so the boot parameter would be **crashkernel=512M**.

   a. Open the GRUB configuration file and add **crashkernel=512M** to the boot parameter line.

```
# vi /etc/default/grub

GRUB_CMDLINE_LINUX="console=tty1 console=ttyS0 earlyprintk=ttyS0 rootdelay=300
crashkernel=512M"
```

   b. Update the GRUB configuration file.

   - In RHEL 9.2 and earlier:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

   - In RHEL 9.3 and later:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

6. Reboot the VM to allocate separate kernel crash memory to the VM.

Verification

- Ensure that **kdump** is active and running.

```
# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor prese>
   Active: active (exited) since Fri 2024-02-09 10:50:18 CET; 1h 20min ago
  Process: 1252 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCES>
 Main PID: 1252 (code=exited, status=0/SUCCESS)
    Tasks: 0 (limit: 16975)
   Memory: 512B
   CGroup: /system.slice/kdump.service
```

Additional resources

- Installing kdump
- How to troubleshoot kernel crashes, hangs, or reboots with kdump on Red Hat Enterprise Linux
- Memory requirements for kdump

3.13. ADDITIONAL RESOURCES

- Red Hat in the Public Cloud
- Red Hat Cloud Access Reference Guide
- Frequently Asked Questions and Recommended Practices for Microsoft Azure

CHAPTER 4. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER ON MICROSOFT AZURE

To create a cluster where RHEL nodes automatically redistribute their workloads if a node failure occurs, use the Red Hat High Availability Add-On. Such high availability (HA) clusters can also be hosted on public cloud platforms, including Microsoft Azure. Creating RHEL HA clusters on Azure is similar to creating HA clusters in non-cloud environments, with certain specifics.

To configure a Red Hat HA cluster on Azure using Azure virtual machine (VM) instances as cluster nodes, see the following sections. The procedures in these sections assume that you are creating a custom image for Azure. You have a number of options for obtaining the RHEL 9 images you use for your cluster. See Red Hat Enterprise Linux Image Options on Azure for information on image options for Azure.

The following sections provide:

- Prerequisite procedures for setting up your environment for Azure. After you set up your environment, you can create and configure Azure VM instances.
- Procedures specific to the creation of HA clusters, which transform individual nodes into a cluster of HA nodes on Azure. These include procedures for installing the High Availability packages and agents on each cluster node, configuring fencing, and installing Azure network resource agents.

Prerequisites

- Sign up for a Red Hat Customer Portal account .
- Sign up for a Microsoft Azure account with administrator privileges.
- You need to install Azure command line interface (CLI). For more information, see Installing the Azure CLI.

4.1. THE BENEFITS OF USING HIGH-AVAILABILITY CLUSTERS ON PUBLIC CLOUD PLATFORMS

A high-availability (HA) cluster is a set of computers (called *nodes*) that are linked together to run a specific workload. The purpose of HA clusters is to provide redundancy in case of a hardware or software failure. If a node in the HA cluster fails, the Pacemaker cluster resource manager distributes the workload to other nodes and no noticeable downtime occurs in the services that are running on the cluster.

You can also run HA clusters on public cloud platforms. In this case, you would use virtual machine (VM) instances in the cloud as the individual cluster nodes. Using HA clusters on a public cloud platform has the following benefits:

- Improved availability: In case of a VM failure, the workload is quickly redistributed to other nodes, so running services are not disrupted.
- Scalability: Additional nodes can be started when demand is high and stopped when demand is low.
- Cost-effectiveness: With the pay-as-you-go pricing, you pay only for nodes that are running.

- Simplified management: Some public cloud platforms offer management interfaces to make configuring HA clusters easier.

To enable HA on your Red Hat Enterprise Linux (RHEL) systems, Red Hat offers a High Availability Add-On. The High Availability Add-On provides all necessary components for creating HA clusters on RHEL systems. The components include high availability service management and cluster administration tools.

Additional resources

- High Availability Add-On overview

4.2. CREATING RESOURCES IN AZURE

Complete the following procedure to create a region, resource group, storage account, virtual network, and availability set. You need these resources to set up a cluster on Microsoft Azure.

Procedure

1. Authenticate your system with Azure and log in.

```
$ az login
```

NOTE

If a browser is available in your environment, the CLI opens your browser to the Azure sign-in page.

Example:

```
[clouduser@localhost]$ az login

To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code
FDMSCMETZ to authenticate.
 [
   {
     "cloudName": "AzureCloud",
     "id": "Subscription ID",
     "isDefault": true,
     "name": "MySubscriptionName",
     "state": "Enabled",
     "tenantId": "Tenant ID",
     "user": {
       "name": "clouduser@company.com",
       "type": "user"
     }
   }
 ]
```

2. Create a resource group in an Azure region.

```
$ az group create --name resource-group --location azure-region
```

Example:

```
[clouduser@localhost]$ az group create --name azrhelclirsgrp --location southcentralus

{
  "id": "/subscriptions//resourceGroups/azrhelclirsgrp",
  "location": "southcentralus",
  "managedBy": null,
  "name": "azrhelclirsgrp",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

3. Create a storage account.

```
$ az storage account create -l azure-region -n storage-account-name -g resource-group --
sku sku_type --kind StorageV2
```

Example:

```
[clouduser@localhost]$ az storage account create -l southcentralus -n azrhelclistact -g
azrhelclirsgrp --sku Standard_LRS --kind StorageV2

{
  "accessTier": null,
  "creationTime": "2017-04-05T19:10:29.855470+00:00",
  "customDomain": null,
  "encryption": null,
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Storage/storageAccounts/azr
helclistact",
  "kind": "StorageV2",
  "lastGeoFailoverTime": null,
  "location": "southcentralus",
  "name": "azrhelclistact",
  "primaryEndpoints": {
    "blob": "https://azrhelclistact.blob.core.windows.net/",
    "file": "https://azrhelclistact.file.core.windows.net/",
    "queue": "https://azrhelclistact.queue.core.windows.net/",
    "table": "https://azrhelclistact.table.core.windows.net/"
},
"primaryLocation": "southcentralus",
"provisioningState": "Succeeded",
"resourceGroup": "azrhelclirsgrp",
"secondaryEndpoints": null,
"secondaryLocation": null,
"sku": {
  "name": "Standard_LRS",
  "tier": "Standard"
},
"statusOfPrimary": "available",
"statusOfSecondary": null,
"tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```

4. Get the storage account connection string.

```
$ az storage account show-connection-string -n storage-account-name -g resource-group
```

Example:

```
[clouduser@localhost]$ az storage account show-connection-string -n azrhelclistact -g
azrhelclirsgrp
{
  "connectionString":
"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=azrhelclistact
AccountKey=NreGk...=="
}
```

5. Export the connection string by copying the connection string and pasting it into the following command. This string connects your system to the storage account.

```
$ export AZURE_STORAGE_CONNECTION_STRING="storage-connection-string"
```

Example:

```
[clouduser@localhost]$ export
AZURE_STORAGE_CONNECTION_STRING="DefaultEndpointsProtocol=https;EndpointSuffi
x=core.windows.net;AccountName=azrhelclistact;AccountKey=NreGk...=="
```

6. Create the storage container.

```
$ az storage container create -n container-name
```

Example:

```
[clouduser@localhost]$ az storage container create -n azrhelclistcont

{
  "created": true
}
```

7. Create a virtual network. All cluster nodes must be in the same virtual network.

```
$ az network vnet create -g resource group --name vnet-name --subnet-name subnet-name
```

Example:

```
[clouduser@localhost]$ az network vnet create --resource-group azrhelclirsgrp --name
azrhelclivnet1 --subnet-name azrhelclisubnet1
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
      "10.0.0.0/16"
      ]
  },
  "dhcpOptions": {
```

```
    "dnsServers": []
  },
  "etag": "W/\"\"",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1",
  "location": "southcentralus",
  "name": "azrhelclivnet1",
  "provisioningState": "Succeeded",
  "resourceGroup": "azrhelclirsgrp",
  "resourceGuid": "0f25efee-e2a6-4abe-a4e9-817061ee1e79",
  "subnets": [
    {
      "addressPrefix": "10.0.0.0/24",
      "etag": "W/\"\"",
      "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1/subnets/azrhelclisubnet1",
      "ipConfigurations": null,
      "name": "azrhelclisubnet1",
      "networkSecurityGroup": null,
      "provisioningState": "Succeeded",
      "resourceGroup": "azrhelclirsgrp",
      "resourceNavigationLinks": null,
      "routeTable": null
    }
  ],
  "tags": {},
  "type": "Microsoft.Network/virtualNetworks",
  "virtualNetworkPeerings": null
  }
}
```


8. Create an availability set. All cluster nodes must be in the same availability set.

```
$ az vm availability-set create --name MyAvailabilitySet --resource-group MyResourceGroup
```

Example:


```
[clouduser@localhost]$ az vm availability-set create --name rhelha-avset1 --resource-group
azrhelclirsgrp
{
  "additionalProperties": {},
    "id":
"/subscriptions/.../resourceGroups/azrhelclirsgrp/providers/Microsoft.Compute/availabilitySets/rh
elha-avset1",
    "location": "southcentralus",
    "name": “rhelha-avset1",
    "platformFaultDomainCount": 2,
    "platformUpdateDomainCount": 5,

[omitted]
```


Additional resources

- Sign in with Azure CLI
- SKU Types
- Azure Managed Disks Overview

4.3. REQUIRED SYSTEM PACKAGES FOR HIGH AVAILABILITY

The procedure assumes you are creating a VM image for Azure HA that uses Red Hat Enterprise Linux. To successfully complete the procedure, the following packages must be installed.

Table 4.1. System packages

Package	Repository	Description
libvirt	rhel-9-for-x86_64-appstream-rpms	Open source API, daemon, and management tool for managing platform virtualization
virt-install	rhel-9-for-x86_64-appstream-rpms	A command-line utility for building VMs
libguestfs	rhel-9-for-x86_64-appstream-rpms	A library for accessing and modifying VM file systems
guestfs-tools	rhel-9-for-x86_64-appstream-rpms	System administration tools for VMs; includes the **virt-customize** utility

4.4. AZURE VM CONFIGURATION SETTINGS

Azure VMs must have the following configuration settings. Some of these settings are enabled during the initial VM creation. Other settings are set when provisioning the VM image for Azure. Keep these settings in mind as you move through the procedures. Refer to them as necessary.

Table 4.2. VM configuration settings

Setting	Recommendation
ssh	ssh must be enabled to provide remote access to your Azure VMs.
dhcp	The primary virtual adapter should be configured for dhcp (IPv4 only).
Swap Space	Do not create a dedicated swap file or swap partition. You can configure swap space with the Windows Azure Linux Agent (WALinuxAgent).
NIC	Choose **virtio** for the primary virtual network adapter.

Setting	Recommendation
encryption	For custom images, use Network Bound Disk Encryption (NBDE) for full disk encryption on Azure.

4.5. INSTALLING HYPER-V DEVICE DRIVERS

Microsoft provides network and storage device drivers as part of their Linux Integration Services (LIS) for Hyper-V package. You may need to install Hyper-V device drivers on the VM image prior to provisioning it as an Azure virtual machine (VM). Use the **lsinitrd | grep hv** command to verify that the drivers are installed.

Procedure

1. Enter the following **grep** command to determine if the required Hyper-V device drivers are installed.

```
# lsinitrd | grep hv
```

In the example below, all required drivers are installed.

```
# lsinitrd | grep hv
drwxr-xr-x   2 root     root            0 Aug 12 14:21 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root        31272 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/hv/hv_vmbus.ko.xz
-rw-r--r--   1 root     root        25132 Aug 11 08:46 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/net/hyperv/hv_netvsc.ko.xz
-rw-r--r--   1 root     root         9796 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el9.x86_64/kernel/drivers/scsi/hv_storvsc.ko.xz
```

If all the drivers are not installed, complete the remaining steps.

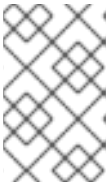
NOTE

An **hv_vmbus** driver may exist in the environment. Even if this driver is present, complete the following steps.

2. Create a file named **hv.conf** in **/etc/dracut.conf.d**.

3. Add the following driver parameters to the **hv.conf** file.

```
add_drivers+=" hv_vmbus "
add_drivers+=" hv_netvsc "
add_drivers+=" hv_storvsc "
add_drivers+=" nvme "
```



NOTE

Note the spaces before and after the quotes, for example, **add_drivers+=" hv_vmbus "**. This ensures that unique drivers are loaded in the event that other Hyper-V drivers already exist in the environment.

4. Regenerate the **initramfs** image.

```
# dracut -f -v --regenerate-all
```

Verification

1. Reboot the machine.
2. Run the **lsinitrd | grep hv** command to verify that the drivers are installed.

4.6. MAKING CONFIGURATION CHANGES REQUIRED FOR A MICROSOFT AZURE DEPLOYMENT

Before you deploy your custom base image to Azure, you must perform additional configuration changes to ensure that the virtual machine (VM) can properly operate in Azure.

Procedure

1. Log in to the VM.
2. Register the VM, and enable the Red Hat Enterprise Linux 9 repository.

```
# subscription-manager register --auto-attach
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```

3. Ensure that the **cloud-init** and **hyperv-daemons** packages are installed.

```
# dnf install cloud-init hyperv-daemons -y
```

4. Create **cloud-init** configuration files that are needed for integration with Azure services:
   a. To enable logging to the Hyper-V Data Exchange Service (KVP), create the **/etc/cloud/cloud.cfg.d/10-azure-kvp.cfg** configuration file and add the following lines to that file.

```
reporting:
    logging:
        type: log
    telemetry:
        type: hyperv
```

   b. To add Azure as a datasource, create the **/etc/cloud/cloud.cfg.d/91-azure_datasource.cfg** configuration file, and add the following lines to that file.

```
datasource_list: [ Azure ]
```

```
datasource:
  Azure:
    apply_network_config: False
```

5. To ensure that specific kernel modules are blocked from loading automatically, edit or create the **/etc/modprobe.d/blocklist.conf** file and add the following lines to that file.

```
blacklist nouveau
blacklist lbm-nouveau
blacklist floppy
blacklist amdgpu
blacklist skx_edac
blacklist intel_cstate
```

6. Modify **udev** network device rules:

   a. Remove the following persistent network device rules if present.

```
# rm -f /etc/udev/rules.d/70-persistent-net.rules
# rm -f /etc/udev/rules.d/75-persistent-net-generator.rules
# rm -f /etc/udev/rules.d/80-net-name-slot-rules
```

   b. To ensure that Accelerated Networking on Azure works as intended, create a new network device rule **/etc/udev/rules.d/68-azure-sriov-nm-unmanaged.rules** and add the following line to it.

```
SUBSYSTEM=="net", DRIVERS=="hv_pci", ACTION=="add",
ENV{NM_UNMANAGED}="1"
```

7. Set the **sshd** service to start automatically.

```
# systemctl enable sshd
# systemctl is-enabled sshd
```

8. Modify kernel boot parameters:

   a. Open the **/etc/default/grub** file, and ensure the **GRUB_TIMEOUT** line has the following value.

```
GRUB_TIMEOUT=10
```

   b. Remove the following options from the end of the **GRUB_CMDLINE_LINUX** line if present.

```
rhgb quiet
```

   c. Ensure the **/etc/default/grub** file contains the following lines with all the specified options.

```
GRUB_CMDLINE_LINUX="loglevel=3 crashkernel=auto console=tty1 console=ttyS0
earlyprintk=ttyS0 rootdelay=300"
GRUB_TIMEOUT_STYLE=countdown
GRUB_TERMINAL="serial console"
GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --
stop=1"
```

d. Regenerate the **grub.cfg** file.

- On a BIOS-based machine:
  - In RHEL 9.2 and earlier:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

  - In RHEL 9.3 and later:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

- On a UEFI-based machine:
  - In RHEL 9.2 and earlier:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

  - In RHEL 9.3 and later:

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg --update-bls-cmdline
```

    If your system uses a non-default location for **grub.cfg**, adjust the command accordingly.

9. Configure the Windows Azure Linux Agent (**WALinuxAgent**):

a. Install and enable the **WALinuxAgent** package.

```
# dnf install WALinuxAgent -y
# systemctl enable waagent
```

b. To ensure that a swap partition is not used in provisioned VMs, edit the following lines in the **/etc/waagent.conf** file.

```
Provisioning.DeleteRootPassword=y
ResourceDisk.Format=n
ResourceDisk.EnableSwap=n
```

10. Prepare the VM for Azure provisioning:

a. Unregister the VM from Red Hat Subscription Manager.

```
# subscription-manager unregister
```

b. Clean up the existing provisioning details.

```
# waagent -force -deprovision
```

NOTE

This command generates warnings, which are expected because Azure handles the provisioning of VMs automatically.

c. Clean the shell history and shut down the VM.

```
# export HISTSIZE=0
# poweroff
```

4.7. CREATING AN AZURE ACTIVE DIRECTORY APPLICATION

Complete the following procedure to create an Azure Active Directory (AD) application. The Azure AD application authorizes and automates access for HA operations for all nodes in the cluster.

Prerequisites

- The Azure Command Line Interface (CLI) is installed on your system.
- You are an Administrator or Owner for the Microsoft Azure subscription. You need this authorization to create an Azure AD application.

Procedure

1. On any node in the HA cluster, log in to your Azure account.

```
$ az login
```

2. Create a **json** configuration file for a custom role for the Azure fence agent. Use the following configuration, but replace *<subscription-id>* with your subscription IDs.

```
{
    "Name": "Linux Fence Agent Role",
    "description": "Allows to power-off and start virtual machines",
    "assignableScopes": [
        "/subscriptions/<subscription-id>"
    ],
    "actions": [
        "Microsoft.Compute/*/read",
        "Microsoft.Compute/virtualMachines/powerOff/action",
        "Microsoft.Compute/virtualMachines/start/action"
    ],
    "notActions": [],
    "dataActions": [],
    "notDataActions": []
}
```

3. Define the custom role for the Azure fence agent. Use the **json** file created in the previous step to do this.

```
$ az role definition create --role-definition azure-fence-role.json

{
  "assignableScopes": [
    "/subscriptions/<my-subscription-id>"
  ],
  "description": "Allows to power-off and start virtual machines",
  "id": "/subscriptions/<my-subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-id>",
```

```
  "name": "<role-id>",
  "permissions": [
   {
    "actions": [
     "Microsoft.Compute/*/read",
     "Microsoft.Compute/virtualMachines/powerOff/action",
     "Microsoft.Compute/virtualMachines/start/action"
    ],
    "dataActions": [],
    "notActions": [],
    "notDataActions": []
   }
  ],
  "roleName": "Linux Fence Agent Role",
  "roleType": "CustomRole",
  "type": "Microsoft.Authorization/roleDefinitions"
}
```

4. In the Azure web console interface, select **Virtual Machine** → Click **Identity** in the left-side menu.
5. Select **On** → Click **Save** → click **Yes** to confirm.
6. Click **Azure role assignments** → **Add role assignment**
7. Select the **Scope** required for the role, for example **Resource Group**.
8. Select the required **Resource Group**.
9. **Optional:** Change the **Subscription** if necessary.
10. Select the **Linux Fence Agent Role** role.
11. Click **Save**.

Verification

- Display nodes visible to Azure AD.

```
# fence_azure_arm --msi -o list
node1,
node2,
[...]
```

If this command outputs all nodes on your cluster, the AD application has been configured successfully.

Additional resources

- View the access a user has to Azure resources
- Create a custom role for the fence agent
- Assign Azure roles by using Azure CLI

4.8. CONVERTING THE IMAGE TO A FIXED VHD FORMAT

All Microsoft Azure VM images must be in a fixed **VHD** format. The image must be aligned on a 1 MB boundary before it is converted to VHD. To convert the image from **qcow2** to a fixed **VHD** format and align the image, see the following procedure. Once you have converted the image, you can upload it to Azure.

Procedure

1. Convert the image from **qcow2** to **raw** format.

```
$ qemu-img convert -f qcow2 -O raw <image-name>.qcow2 <image-name>.raw
```

2. Create a shell script with the following content.

```
#!/bin/bash
MB=$((1024 * 1024))
size=$(qemu-img info -f raw --output json "$1" | gawk 'match($0, /"virtual-size": ([0-9]+),/, val)
{print val[1]}')
rounded_size=$((($size/$MB + 1) * $MB))
if [ $(($size % $MB)) -eq  0 ]
then
 echo "Your image is already aligned. You do not need to resize."
 exit 1
fi
echo "rounded size = $rounded_size"
export rounded_size
```

3. Run the script. This example uses the name **align.sh**.

```
$ sh align.sh <image-xxx>.raw
```

   - If the message *"Your image is already aligned. You do not need to resize."* displays, proceed to the following step.
   - If a value displays, your image is not aligned.

4. Use the following command to convert the file to a fixed **VHD** format.
   The sample uses qemu-img version 2.12.0.

```
$ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
<image.xxx>.vhd
```

   Once converted, the **VHD** file is ready to upload to Azure.

5. If the **raw** image is not aligned, complete the following steps to align it.

   a. Resize the **raw** file by using the rounded value displayed when you ran the verification script.

```
$ qemu-img resize -f raw <image-xxx>.raw <rounded-value>
```

   b. Convert the **raw** image file to a **VHD** format.
   The sample uses qemu-img version 2.12.0.

```
$ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
<image.xxx>.vhd
```

Once converted, the **VHD** file is ready to upload to Azure.

4.9. UPLOADING AND CREATING AN AZURE IMAGE

Complete the following steps to upload the **VHD** file to your container and create an Azure custom image.

NOTE

The exported storage connection string does not persist after a system reboot. If any of the commands in the following steps fail, export the connection string again.

Procedure

1. Upload the **VHD** file to the storage container. It may take several minutes. To get a list of storage containers, enter the **az storage container list** command.

```
$ az storage blob upload \
    --account-name <storage-account-name> --container-name <container-name> \
    --type page --file <path-to-vhd> --name <image-name>.vhd
```

Example:

```
[clouduser@localhost]$ az storage blob upload \
--account-name azrhelclistact --container-name azrhelclistcont \
--type page --file rhel-image-{ProductNumber}.vhd --name rhel-image-{ProductNumber}.vhd

Percent complete: %100.0
```

2. Get the URL for the uploaded **VHD** file to use in the following step.

```
$ az storage blob url -c <container-name> -n <image-name>.vhd
```

Example:

```
$ az storage blob url -c azrhelclistcont -n rhel-image-9.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.vhd"
```

3. Create the Azure custom image.

```
$ az image create -n <image-name> -g <resource-group> -l <azure-region> --source <URL>
--os-type linux
```

NOTE

The default hypervisor generation of the VM is V1. You can optionally specify a V2 hypervisor generation by including the option **--hyper-v-generation V2**. Generation 2 VMs use a UEFI-based boot architecture. See Support for generation 2 VMs on Azure for information about generation 2 VMs.

The command may return the error "Only blobs formatted as VHDs can be imported." This error may mean that the image was not aligned to the nearest 1 MB boundary before it was converted to **VHD**.

Example:

```
$ az image create -n rhel9 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.vhd --os-type linux
```

4.10. INSTALLING RED HAT HA PACKAGES AND AGENTS

Complete the following steps on all nodes.

Procedure

1. Launch an SSH terminal session and connect to the VM by using the administrator name and public IP address.

```
$ ssh administrator@PublicIP
```

To get the public IP address for an Azure VM, open the VM properties in the Azure Portal or enter the following Azure CLI command.

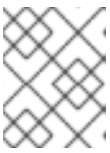
```
$ az vm list -g <resource-group> -d --output table
```

Example:

```
[clouduser@localhost ~] $ az vm list -g azrhelclirsgrp -d --output table
Name    ResourceGroup          PowerState      PublicIps       Location
------  ---------------------  --------------  -------------   --------------
node01  azrhelclirsgrp         VM running      192.98.152.251  southcentralus
```

2. Register the VM with Red Hat.

```
$ sudo -i
# subscription-manager register --auto-attach
```



NOTE

If the **--auto-attach** command fails, manually register the VM to your subscription.

3. Disable all repositories.

```
# subscription-manager repos --disable=*
```

4. Enable the RHEL 9 Server HA repositories.

```
# subscription-manager repos --enable=rhel-9-for-x86_64-highavailability-rpms
```

5. Update all packages.

```
# dnf update -y
```

6. Install the Red Hat High Availability Add-On software packages, along with the Azure fencing agent from the High Availability channel.

```
# dnf install pcs pacemaker fence-agents-azure-arm
```

7. The user **hacluster** was created during the pcs and pacemaker installation in the previous step. Create a password for **hacluster** on all cluster nodes. Use the same password for all nodes.

```
# passwd hacluster
```

8. Add the **high availability** service to the RHEL Firewall if **firewalld.service** is installed.

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --reload
```

9. Start the **pcs** service and enable it to start on boot.

```
# systemctl start pcsd.service
# systemctl enable pcsd.service

Created symlink from /etc/systemd/system/multi-user.target.wants/pcsd.service to
/usr/lib/systemd/system/pcsd.service.
```

Verification

- Ensure the **pcs** service is running.

```
# systemctl status pcsd.service
pcsd.service - PCS GUI and remote configuration interface
Loaded: loaded (/usr/lib/systemd/system/pcsd.service; enabled; vendor preset: disabled)
Active: active (running) since Fri 2018-02-23 11:00:58 EST; 1min 23s ago
Docs: man:pcsd(8)
        man:pcs(8)
Main PID: 46235 (pcsd)
  CGroup: /system.slice/pcsd.service
        └─46235 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
```

4.11. CREATING A CLUSTER

Complete the following steps to create the cluster of nodes.

Procedure

1. On one of the nodes, enter the following command to authenticate the pcs user **hacluster**. In the command, specify the name of each node in the cluster.

```
# pcs host auth <hostname1> <hostname2> <hostname3>
```

Example:

```
[root@node01 clouduser]# pcs host auth node01 node02 node03
Username: hacluster
Password:
node01: Authorized
node02: Authorized
node03: Authorized
```

2. Create the cluster.

```
# pcs cluster setup <cluster_name> <hostname1> <hostname2> <hostname3>
```

Example:

```
[root@node01 clouduser]# pcs cluster setup new_cluster node01 node02 node03

[...]

Synchronizing pcsd certificates on nodes node01, node02, node03...
node02: Success
node03: Success
node01: Success
Restarting pcsd on the nodes in order to reload the certificates...
node02: Success
node03: Success
node01: Success
```

Verification

1. Enable the cluster.

```
[root@node01 clouduser]# pcs cluster enable --all
node02: Cluster Enabled
node03: Cluster Enabled
node01: Cluster Enabled
```

2. Start the cluster.

```
[root@node01 clouduser]# pcs cluster start --all
node02: Starting Cluster...
node03: Starting Cluster...
node01: Starting Cluster...
```

4.12. FENCING OVERVIEW

If communication with a single node in the cluster fails, then other nodes in the cluster must be able to restrict or release access to resources that the failed cluster node may have access to. This cannot be

accomplished by contacting the cluster node itself as the cluster node may not be responsive. Instead, you must provide an external method, which is called fencing with a fence agent.

A node that is unresponsive may still be accessing data. The only way to be certain that your data is safe is to fence the node by using STONITH. STONITH is an acronym for "Shoot The Other Node In The Head," and it protects your data from being corrupted by rogue nodes or concurrent access. Using STONITH, you can be certain that a node is truly offline before allowing the data to be accessed from another node.

Additional resources

- Fencing in Red Hat High Availability Cluster

4.13. CREATING A FENCING DEVICE

Complete the following steps to configure fencing. Complete these commands from any node in the cluster

Prerequisites

You need to set the cluster property **stonith-enabled** to **true**.

Procedure

1. Identify the Azure node name for each RHEL VM. You use the Azure node names to configure the fence device.

```
# fence_azure_arm \
   -l <AD-Application-ID> -p <AD-Password> \
   --resourceGroup <MyResourceGroup> --tenantId <Tenant-ID> \
   --subscriptionId <Subscription-ID> -o list
```

Example:

```
[root@node01 clouduser]# fence_azure_arm \
-l e04a6a49-9f00-xxxx-xxxx-a8bdda4af447 -p
z/a05AwCN0IzAjVwXXXXXXXEWIoeVp0xg7QT//JE=
--resourceGroup azrhelclirsgrp --tenantId 77ecefb6-cff0-XXXX-XXXX-757XXXX9485
--subscriptionId XXXXXXXX-38b4-4527-XXXX-012d49dfc02c -o list

node01,
node02,
node03,
```

2. View the options for the Azure ARM STONITH agent.

```
# pcs stonith describe fence_azure_arm
```

Example:

```
# pcs stonith describe fence_apc
Stonith options:
password: Authentication key
password_script: Script to run to retrieve password
```

WARNING

For fence agents that provide a method option, do not specify a value of cycle as it is not supported and can cause data corruption.

Some fence devices can fence only a single node, while other devices can fence multiple nodes. The parameters you specify when you create a fencing device depend on what your fencing device supports and requires.

You can use the **pcmk_host_list** parameter when creating a fencing device to specify all of the machines that are controlled by that fencing device.

You can use **pcmk_host_map** parameter when creating a fencing device to map host names to the specifications that comprehends the fence device.

3. Create a fence device.

```
# pcs stonith create clusterfence fence_azure_arm
```

Verification

1. Test the fencing agent for one of the other nodes.

```
# pcs stonith fence azurenodename
```

Example:

```
[root@node01 clouduser]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: node01 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Feb 23 11:44:35 2018
Last change: Fri Feb 23 11:21:01 2018 by root via cibadmin on node01

3 nodes configured
1 resource configured

Online: [ node01 node03 ]
OFFLINE: [ node02 ]

Full list of resources:

 clusterfence  (stonith:fence_azure_arm):  Started node01

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

2. Start the node that was fenced in the previous step.

```
# pcs cluster start <hostname>
```

3. Check the status to verify the node started.

```
# pcs status
```

Example:

```
[root@node01 clouduser]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: node01 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Feb 23 11:34:59 2018
Last change: Fri Feb 23 11:21:01 2018 by root via cibadmin on node01

3 nodes configured
1 resource configured

Online: [ node01 node02 node03 ]

Full list of resources:

clusterfence    (stonith:fence_azure_arm):  Started node01

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

Additional resources

- Fencing in a Red Hat High Availability Cluster
- General properties of fencing devices

4.14. CREATING AN AZURE INTERNAL LOAD BALANCER

The Azure internal load balancer removes cluster nodes that do not answer health probe requests.

Perform the following procedure to create an Azure internal load balancer. Each step references a specific Microsoft procedure and includes the settings for customizing the load balancer for HA.

Prerequisites

Azure control panel

Procedure

1. Create a Basic load balancer . Select **Internal load balancer**, the **Basic SKU**, and **Dynamic** for the type of IP address assignment.

2. Create a back-end address pool . Associate the backend pool to the availability set created while creating Azure resources in HA. Do not set any target network IP configurations.

3. Create a health probe. For the health probe, select **TCP** and enter port **61000**. You can use TCP port number that does not interfere with another service. For certain HA product applications (for example, SAP HANA and SQL Server), you may need to work with Microsoft to identify the correct port to use.

4. Create a load balancer rule. To create the load balancing rule, the default values are prepopulated. Ensure to set **Floating IP (direct server return)** to **Enabled**.

4.15. CONFIGURING THE LOAD BALANCER RESOURCE AGENT

After you have created the health probe, you must configure the **load balancer** resource agent. This resource agent runs a service that answers health probe requests from the Azure load balancer and removes cluster nodes that do not answer requests.

Procedure

1. Install the **nmap-ncat** resource agents on all nodes.

```
# dnf install nmap-ncat resource-agents-cloud
```

   Perform the following steps on a single node.

2. Create the **pcs** resources and group. Use your load balancer FrontendIP for the IPaddr2 address.

```
# pcs resource create resource-name IPaddr2 ip="10.0.0.7" --group cluster-resources-group
```

3. Configure the **load balancer** resource agent.

```
# pcs resource create resource-loadbalancer-name azure-lb port=port-number --group
cluster-resources-group
```

Verification

- Run **pcs status** to see the results.

```
[root@node01 clouduser]# pcs status
```

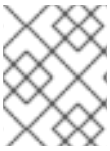
  Example output:

```
Cluster name: clusterfence01
Stack: corosync
Current DC: node02 (version 1.1.16-12.el7_4.7-94ff4df) - partition with quorum
Last updated: Tue Jan 30 12:42:35 2018
Last change: Tue Jan 30 12:26:42 2018 by root via cibadmin on node01

3 nodes configured
3 resources configured

Online: [ node01 node02 node03 ]

Full list of resources:
```

```
clusterfence (stonith:fence_azure_arm):      Started node01
Resource Group: g_azure
    vip_azure  (ocf::heartbeat:IPaddr2):       Started node02
    lb_azure   (ocf::heartbeat:azure-lb):      Started node02

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

4.16. CONFIGURING SHARED BLOCK STORAGE

To configure shared block storage for a Red Hat High Availability cluster with Microsoft Azure Shared Disks, use the following procedure. Note that this procedure is optional, and the steps below assume three Azure VMs (a three-node cluster) with a 1 TB shared disk.



NOTE

This is a stand-alone sample procedure for configuring block storage. The procedure assumes that you have not yet created your cluster.

Prerequisites

- You must have installed the Azure CLI on your host system and created your SSH key(s).
- You must have created your cluster environment in Azure, which includes creating the following resources. Links are to the Microsoft Azure documentation.
  - Resource group
  - Virtual network
  - Network security group(s)
  - Network security group rules
  - Subnet(s)
  - Load balancer (optional)
  - Storage account
  - Proximity placement group
  - Availability set

Procedure

1. Create a shared block volume by using the Azure command **az disk create**.

```
$ az disk create -g <resource_group> -n <shared_block_volume_name> --size-gb
<disk_size> --max-shares <number_vms> -l <location>
```

For example, the following command creates a shared block volume named **shared-block-volume.vhd** in the resource group **sharedblock** within the Azure Availability Zone **westcentralus**.

```
$ az disk create -g sharedblock-rg -n shared-block-volume.vhd --size-gb 1024 --max-shares
3 -l westcentralus

{
  "creationData": {
    "createOption": "Empty",
    "galleryImageReference": null,
    "imageReference": null,
    "sourceResourceId": null,
    "sourceUniqueId": null,
    "sourceUri": null,
    "storageAccountId": null,
    "uploadSizeBytes": null
  },
  "diskAccessId": null,
  "diskIopsReadOnly": null,
  "diskIopsReadWrite": 5000,
  "diskMbpsReadOnly": null,
  "diskMbpsReadWrite": 200,
  "diskSizeBytes": 1099511627776,
  "diskSizeGb": 1024,
  "diskState": "Unattached",
  "encryption": {
    "diskEncryptionSetId": null,
    "type": "EncryptionAtRestWithPlatformKey"
  },
  "encryptionSettingsCollection": null,
  "hyperVgeneration": "V1",
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-
rg/providers/Microsoft.Compute/disks/shared-block-volume.vhd",
  "location": "westcentralus",
  "managedBy": null,
  "managedByExtended": null,
  "maxShares": 3,
  "name": "shared-block-volume.vhd",
  "networkAccessPolicy": "AllowAll",
  "osType": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "sharedblock-rg",
  "shareInfo": null,
  "sku": {
    "name": "Premium_LRS",
    "tier": "Premium"
  },
  "tags": {},
  "timeCreated": "2020-08-27T15:36:56.263382+00:00",
  "type": "Microsoft.Compute/disks",
  "uniqueId": "cd8b0a25-6fbe-4779-9312-8d9cbb89b6f2",
  "zones": null
}
```

2. Verify that you have created the shared block volume by using the Azure command **az disk show**.

```
$ az disk show -g <resource_group> -n <shared_block_volume_name>
```

For example, the following command shows details for the shared block volume **shared-block-volume.vhd** within the resource group **sharedblock-rg**.

```
$ az disk show -g sharedblock-rg -n shared-block-volume.vhd

{
  "creationData": {
    "createOption": "Empty",
    "galleryImageReference": null,
    "imageReference": null,
    "sourceResourceId": null,
    "sourceUniqueId": null,
    "sourceUri": null,
    "storageAccountId": null,
    "uploadSizeBytes": null
  },
  "diskAccessId": null,
  "diskIopsReadOnly": null,
  "diskIopsReadWrite": 5000,
  "diskMbpsReadOnly": null,
  "diskMbpsReadWrite": 200,
  "diskSizeBytes": 1099511627776,
  "diskSizeGb": 1024,
  "diskState": "Unattached",
  "encryption": {
    "diskEncryptionSetId": null,
    "type": "EncryptionAtRestWithPlatformKey"
  },
  "encryptionSettingsCollection": null,
  "hyperVgeneration": "V1",
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-
rg/providers/Microsoft.Compute/disks/shared-block-volume.vhd",
  "location": "westcentralus",
  "managedBy": null,
  "managedByExtended": null,
  "maxShares": 3,
  "name": "shared-block-volume.vhd",
  "networkAccessPolicy": "AllowAll",
  "osType": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "sharedblock-rg",
  "shareInfo": null,
  "sku": {
    "name": "Premium_LRS",
    "tier": "Premium"
  },
  "tags": {},
  "timeCreated": "2020-08-27T15:36:56.263382+00:00",
  "type": "Microsoft.Compute/disks",
```

```
  "uniqueId": "cd8b0a25-6fbe-4779-9312-8d9cbb89b6f2",
  "zones": null
}
```

3. Create three network interfaces by using the Azure command **az network nic create**. Run the following command three times by using a different **<nic_name>** for each.

```
$ az network nic create \
    -g <resource_group> -n <nic_name> --subnet <subnet_name> \
    --vnet-name <virtual_network> --location <location> \
    --network-security-group <network_security_group> --private-ip-address-version IPv4
```

For example, the following command creates a network interface with the name **shareblock-nodea-vm-nic-protected**.

```
$ az network nic create \
    -g sharedblock-rg -n sharedblock-nodea-vm-nic-protected --subnet sharedblock-subnet-protected \
    --vnet-name sharedblock-vn --location westcentralus \
    --network-security-group sharedblock-nsg --private-ip-address-version IPv4
```

4. Create three VMs and attach the shared block volume by using the Azure command **az vm create**. Option values are the same for each VM except that each VM has its own **<vm_name>**, **<new_vm_disk_name>**, and **<nic_name>**.

```
$ az vm create \
    -n <vm_name> -g <resource_group> --attach-data-disks <shared_block_volume_name> \
    --data-disk-caching None --os-disk-caching ReadWrite --os-disk-name <new-vm-disk-name> \
    --os-disk-size-gb <disk_size> --location <location> --size <virtual_machine_size> \
    --image <image_name> --admin-username <vm_username> --authentication-type ssh \
    --ssh-key-values <ssh_key> --nics <nic_name> --availability-set <availability_set> --ppg <proximity_placement_group>
```

For example, the following command creates a VM named **sharedblock-nodea-vm**.

```
$ az vm create \
-n sharedblock-nodea-vm -g sharedblock-rg --attach-data-disks shared-block-volume.vhd \
--data-disk-caching None --os-disk-caching ReadWrite --os-disk-name sharedblock-nodea-vm.vhd \
--os-disk-size-gb 64 --location westcentralus --size Standard_D2s_v3 \
--image /subscriptions/12345678910-12345678910/resourceGroups/sample-azureimagesgroupwestcentralus/providers/Microsoft.Compute/images/sample-azure-rhel-9.3.0-20200713.n.0.x86_64 --admin-username sharedblock-user --authentication-type ssh \
--ssh-key-values @sharedblock-key.pub --nics sharedblock-nodea-vm-nic-protected --availability-set sharedblock-as --ppg sharedblock-ppg

{
  "fqdns": "",
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-rg/providers/Microsoft.Compute/virtualMachines/sharedblock-nodea-vm",
  "location": "westcentralus",
  "macAddress": "00-22-48-5D-EE-FB",
  "powerState": "VM running",
```

```
  "privateIpAddress": "198.51.100.3",
  "publicIpAddress": "",
  "resourceGroup": "sharedblock-rg",
  "zones": ""
}
```

Verification

1. For each VM in your cluster, verify that the block device is available by using the **ssh** command with your VM's IP address.

```
# ssh <ip_address> "hostname ; lsblk -d | grep ' 1T '"
```

For example, the following command lists details including the host name and block device for the VM IP **198.51.100.3**.

```
# ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T '"

nodea
sdb    8:16   0    1T  0 disk
```

2. Use the **ssh** command to verify that each VM in your cluster uses the same shared disk.

```
# ssh <ip_address> "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info
--query=all --name=/dev/{} | grep '^E: ID_SERIAL='"
```

For example, the following command lists details including the host name and shared disk volume ID for the instance IP address **198.51.100.3**.

```
# ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info -
-query=all --name=/dev/{} | grep '^E: ID_SERIAL='"

nodea
E: ID_SERIAL=3600224808dd8eb102f6ffc5822c41d89
```

After you have verified that the shared disk is attached to each VM, you can configure resilient storage for the cluster.

Additional resources

- Configuring a GFS2 file system in a cluster
- Configuring GFS2 file systems

4.17. ADDITIONAL RESOURCES

- Support Policies for RHEL High Availability Clusters - Microsoft Azure Virtual Machines as Cluster Members
- Configuring and Managing High Availability Clusters