Red Hat Container Development Kit 2.4
Installation Guide

Guide to installing Red Hat Container Development Kit

Brian Brock Robert Krátký Chris Negus
Red Hat Developer Group Documentation Team

Red Hat Container Development Kit 2.4 Installation Guide

Guide to installing Red Hat Container Development Kit

Brian Brock
bbrock@redhat.com

Robert Krátký
rkratky@redhat.com

Chris Negus

Legal Notice

Abstract

This guide contains instructions on how to install Red Hat Container Development Kit on Microsoft Windows, macOS, and Red Hat Enterprise Linux. Information and hands-on guidance is provided for obtaining all prerequisites for each of the supported platforms. You will also learn how to launch the Container Development Environment provided by Red Hat Container Development Kit, so that you can start to develop containerized applications.

Table of Contents

CHAPTER 1. INTRODUCING RED HAT CONTAINER DEVELOPMENT KIT

Red Hat Container Development Kit is a platform for developing containerized applications — it is a set of tools that enables developers to quickly and easily set up an environment for developing and testing containerized applications on the Red Hat Enterprise Linux platform.

- Container Development Kit provides a personal Container Development Environment you can install on your own laptop, desktop, or server system. The Container Development Environment is provided in the form of a Red Hat Enterprise Linux virtual machine. The Container Development Environment itself can also be installed in a virtual machine.
- Container Development Kit includes the same container-development and run-time tools used to create and deploy containers for large data centers.
- Container Development Kit offers an easy installation method that results in virtual machines created from pre-configured Vagrant boxes and Vagrantfiles running on your local system.
- Container Development Kit is available for Microsoft Windows, Mac OS X, and Linux operating systems, thus allowing developers to use their favorite platform while producing applications ready to be deployed in the Red Hat Enterprise Linux ecosystem.

Container Development Kit is a part of the Red Hat Developers program, which provides tools, resources, and support for developers who wish to utilize Red Hat solutions and products to create applications, both locally and in the cloud. For additional information and to register to become a part of the program, visit developers.redhat.com.

1.1. UNDERSTANDING CONTAINER DEVELOPMENT KIT DOCUMENTATION

- The Red Hat Container Development Kit 2.4 Release Notes and Known Issues contains information about the current release of the product as well as a list of known problems that users may encounter when using it.
- The Red Hat Container Development Kit 2.4 Installation Guide contains instructions for installing the Container Development Environment provided by Red Hat Container Development Kit on your chosen system.
- The Red Hat Container Development Kit 2.4 Getting Started Guide contains instructions on how to start using the Container Development Environment to develop Red Hat Enterprise Linux-based containers using tools and services such as **OpenShift Container Platform**, **Docker**, **Eclipse**, and various command-line tools.
- Report issues with Red Hat Container Development Kit or request new features using the **CDK** project at https://issues.jboss.org/projects/CDK.

1.2. WHAT IS IN CONTAINER DEVELOPMENT KIT

Once the Container Development Environment from Container Development Kit is running on your system, you can begin exploring the contents. Some services and tools run automatically when you boot the virtual machine while others require configuration. Here is a list of some of those features:

- **OpenShift Container Platform**: A containerized version of OpenShift Container Platform is included in the Container Development Environment. OpenShift Container Platform provides developers with a platform for creating, provisioning, managing, and scaling container-based

applications for cloud environments. Once the OpenShift container is running, you can use a web console from your browser or work from the command line with the **oc** command to develop container projects.

- **Docker**: The Docker project develops the basic container format and the **docker** command for working with containers that are included in the Container Development Environment. The Environment is configured to have the **docker** daemon start automatically when you boot the virtual machine. With the **docker** command, you can build, run, start, stop, investigate, and otherwise work with individual containers.

- **Kubernetes**: To orchestrate containers in what are referred to as pods, Container Development Kit comes with all the features needed to run a Kubernetes cluster. Within the Container Development Environment, you can use Kubernetes directly with your own container-development tools, or use the version of Kubernetes that is integrated in OpenShift. When started without OpenShift, Kubernetes is set up to run as an all-in-one Kubernetes master (for managing pods) and node (for running pods).

  Kubernetes includes features for replicating pods (to scale up applications on the fly) and services (for interconnecting sets of containers). When you start the Vagrantfile for Kubernetes, the virtual machine starts with all required Kubernetes services running.

- **Linking to Eclipse**: The Container Development Environment lets you connect to it from the Eclipse IDE with **Linux Tools** and the **Docker Tooling** plug-in. This allows Container Development Kit users to manage containers from a graphical interface on their host system.

1.3. WHERE YOU CAN RUN CONTAINER DEVELOPMENT KIT

Red Hat Container Development Kit was designed to let you do your container development using the same computer on which you do your other work. Container Development Kit can be installed on the following systems:

- **Microsoft Windows**: You can use a 64-bit version of Microsoft Windows to install Container Development Kit. Windows 7 or later is required. Virtualization support must be included and turned on in the BIOS.

- **macOS**: You can use an Intel-based Apple Mac to install and run Container Development Kit. The computer should have at least 4 GB of RAM and be running a recent, 64-bit version of macOS, such as 10.11 (El Capitan). Virtualization support must be included and turned on in the BIOS.

- **Red Hat Enterprise Linux**: The latest version of Red Hat Enterprise Linux 7 is recommended for installing Container Development Kit. A 64-bit computer with at least 4 GB of RAM and virtualization support is required.

See the Container Development Kit installation procedure for each system for more detailed hardware and software requirements.

1.4. OBTAINING AND SETTING UP CONTAINER DEVELOPMENT KIT

Container Development Kit is available from the Red Hat Customer Portal to anyone with a valid Red Hat Enterprise Linux Developer Subscription. Joining the Red Hat Developers program also provides a path to getting Container Development Kit.

This guide provides the following instructions for installing Container Development Kit to begin developing containerized applications:

- Choosing your development system (Microsoft Windows, macOS, or Red Hat Enterprise Linux).
- Making the Container Development Kit Vagrant box and Vagrantfiles available on your system.
- Starting the Container Development Environment from the Vagrant box with the selected configuration (OpenShift or stand-alone Kubernetes).

1.5. INSTALLATION COMPONENTS

Regardless of which platform you choose as the workstation for using Container Development Kit, you will use Vagrant to start and manage it. There are a few things you should know about Vagrant before you get started:

- Vagrant is an open-source tool for using light-weight, portable, and consistent development environments.
- Virtual machines that are packaged for use with Vagrant are called boxes.
- Container Development Kit is delivered with two different Vagrantfiles to configure Container Development Kit software components in different ways.

While this guide contains separate chapters for installing on Microsoft Windows, macOS, and Linux systems, some of the components and steps are the same for all installation types. For example, the Red Hat Container Tools ZIP file you download includes Vagrantfiles that are used on all platforms. Likewise, Vagrant boxes you use are tied to the hypervisor (such as VirtualBox or libvirt) rather than the operating system. The installation and use of Vagrant plugins is also the same on all operating systems.

Now that you understand the basics of what is inside Container Development Kit, choose from one of the next chapters to learn how to obtain Container Development Kit and install it on a Microsoft Windows, macOS, or Red Hat Enterprise Linux system.

1.6. ADDITIONAL RESOURCES

- Red Hat Container Development Kit 2.4 Getting Started Guide: Steps you through your first experiences using Container Development Kit.
- Getting Started with Containers: Describes the basics of setting up Docker and Kubernetes (all-in-one or cluster) to run containers. It also covers basic storage setup, Kubernetes troubleshooting, starting containers with systemd, and running super-privileged containers.

CHAPTER 2. UPGRADING CONTAINER DEVELOPMENT KIT

Container Development Kit does not offer any automated upgrade mechanism. To upgrade the software, the current version needs to be uninstalled and a new version installed. Follow the steps outlined Chapter 9, *Uninstalling or Reinstalling Container Development Kit* to uninstall the current version of the Container Development Environment and install a new one.

CHAPTER 3. INSTALLING CONTAINER DEVELOPMENT KIT ON MICROSOFT WINDOWS

Note

The preferred way to install Container Development Kit on Microsoft Windows is to use the all-in-one installer, which is provided by Red Hat Development Suite. See the Red Hat Development Suite Installation Guide for detailed information. To proceed with a manual installation procedure, follow the instruction in this section.

To prepare your Microsoft Windows development system to run the Container Development Kit, the steps are:

1. Download and install **Cygwin**.
2. Download and install **VirtualBox** or enable **Hyper-V**.
3. Download and install **Vagrant**.
4. Download **Red Hat Container Tools** and the Container Development Kit Vagrant box for **VirtualBox** or **Hyper-V**.
5. Install additional **Vagrant** plugins to support Red Hat Subscription Management and other features.

3.1. PREREQUISITES FOR INSTALLING CONTAINER DEVELOPMENT KIT ON MICROSOFT WINDOWS

To run the Container Development Kit on a Microsoft Windows system, you need:

- A 64-bit machine with a minimum of 4 GB of RAM running Windows 7 64-bit or later.
- A minimum of 3 GB of free disk space for virtual machine images.
- Adequate Internet connectivity to download 1—2 GB of software.
- An available Red Hat Enterprise Linux subscription. Note that Red Hat Enterprise Linux subscriptions with self-support do not have access to all of the necessary software for Container Development Kit in all environments.
- A text editor that allows editing text files that do not have extentions, such as **`Vagrantfile`**. Ideally, your text editor should be flexible about line endings as you may encounter files that only have new-lines.
- The **`rsync`** and **`ssh`** command-line utilities need to be installed. The recommended source is the Cygwin project. If you choose to use **`rsync`** and **`ssh`** from an alternate source, there may be some path and command-line incompatibilities that are not addressed in this guide.

3.2. INSTALLING SOFTWARE AND CONFIGURING THE HOST SYSTEM

This chapter describes installation and configuration of software prerequisities for running the Container Development Environment.

Command-line instructions use path names with a tilde, such as **`~/cdk`**. This is a shorthand notation for a path relative to the current user's home directory, **`C:\Users\<username>\cdk`**. The **`%USERPROFILE%`** environment variable contains the Windows path to the user's home directory and is the equivalent to **`~/`** in Linux.

3.2.1. Additional Software Requirements for Microsoft Windows

It is recommended that you install the Cygwin environment from cygwin.com to provide the **`ssh`** and **`rsync`** tools for use by Vagrant. Please note that if you choose to use **`ssh`** and **`rsync`** from a different source, you may run into some incompatibilities in command-line arguments and path names.

An SSH client is required to access Vagrant virtual machines. While a graphical utility that provides SSH, such as PuTTY, can be used, it is preferred to have an SSH client that can be run from the command line. Vagrant manages the SSH environment, allowing you to log into your Vagrant box by simply running the **`vagrant ssh`** command. For this to work, **`ssh`** must be in the Windows path.

When **`rsync`** synchronization has been specified, Vagrant does not start if **`rsync.exe`** is not in the path.

3.2.1.1. Installing Cygwin

Download and install **Cygwin** from the Cygwin Installation page. **Cygwin** installation adds **`ssh.exe`** and **`rsync.exe`** to your path.

To temporarily add the **Cygwin** path for the current **`cmd.exe`** session, use the following command.

```
C:> PATH=%PATH%;C:\cygwin64\bin;
```

To add this location to your path permanently on a Microsoft Windows 10 system, right-click on the desktop and select **Start** → **System** → **Advanced system settings** → **Environment variables**. Then change the **`PATH`** variable as shown.

3.2.1.2. Installing and Enabling Hyper-V

1. Install and enable **Hyper-V** support by following instructions at Install Hyper-V on Windows 10.
2. Add a **Virtual Switch** using the **Hyper-V Manager** by following instructions at Create a Virtual Switch.

3.2.1.3. Installing VirtualBox

Download and install **VirtualBox** for Microsoft Windows from virtualbox.org.

Important

Container Development Kit 2.4 is known to not work correctly with **VirtualBox 5.1.x**. If you intend to use **VirtualBox** as your virtualization provider, and you already have **VirtualBox 5.1.x** installed, downgrade your installation to **VirtualBox 5.0.26**.

Optional: Choose a location for storing VirtualBox VM images. By default, these are stored in your home directory (**`~/VirtualBox VMs`**). You will need several gigabytes of space wherever you choose to store these images. To change the default location for VirtualBox images, use the **VirtualBox** → **Preferences** → **General** menu, then change **Default Machine Folder** to the desired location. Documentation for VirtualBox can be found on the virtualbox.org website.

Note that the *Start Menu* entry for **VirtualBox** is **Oracle VM VirtualBox**.

See VirtualBox Documentation for additional information.

3.2.1.4. Installing Vagrant

Vagrant is used to run a Red Hat Enterprise Linux virtual machine with all necessary components for Container Development Kit. Virtualization is provided by installing **VirtualBox** or by enabling **Hyper-V**.

Please note that Vagrant is strictly command-line oriented. All interaction with Vagrant is through the command line from a command prompt. Vagrant does not install any *Start Menu* entries or desktop shortcuts. You must launch Cygwin's Terminal Window, **`cmd.exe`**, or other shell to run Vagrant.

1. Download and install **Vagrant** from https://releases.hashicorp.com/vagrant/. Select the folder with the 1.8.1 release and download the installer in the **`.msi`** format (**`vagrant_1.8.1.msi`**). The 1.7.4 version of **Vagrant** is also supported for use with Red Hat Container Development Kit 2.4, but 1.8.1 is necessary to use the **Hyper-V** hypervisor. The **Vagrant** installer automatically adds **Vagrant** to the path.
2. Download and install Microsoft Visual C++ 2010 SP1 Redistributable Package (x86) (this is a requirement for using **Vagrant 1.8.1**).

3.2.1.5. Obtaining Container Development Kit Components

Downloading from the Red Hat Customer Portal

Download the Container Development Kit components from Red Hat Product Downloads. You must log in to get access to this page. If you are on the right page, you should see "Product Variant: Red Hat Container Development Kit". You need to download the following items:

- Red Hat Container Tools
- RHEL 7.3 Vagrant box for VirtualBox **or**
- RHEL 7.3 Vagrant box for Hyper-V

Note

The page also offers Vagrant box downloads formatted for other virtualization platforms, such as libvirt. You only need to download the box image that matches the virtualization you are using — **VirtualBox** or **Hyper-V**.

Downloading through the Red Hat Developers Program

Alternatively, if you do not have a valid Red Hat Enterprise Linux subscription, you can obtain Container Development Kit by registering with the **Red Hat Developers** program. Follow the instructions at Red Hat Container Development Kit Get Started.

Download Path

The instructions in this guide assume you have saved the downloaded Container Development Kit components in your home directory in **`%USERPROFILE%\Downloads`**. If you used a different directory, adjust the paths accordingly. You need several gigabytes of free space for the Vagrant box images.

3.2.2. Translating Windows Paths and rsync

`rsync` uses POSIX-style path names and cannot use Windows paths that contain drive letters and backslashes directly. Cygwin uses paths that start at **`/cygdrive/drive-letter/`** so, **`C:\file.exe`** translates to **`/cygdrive/c/file.exe`**.

Other ports of Linux utilities use different conventions. For example, the MinGW/MSYS environment uses paths that start at **`/drive-letter/`**, so **`C:\file.exe`** translates to **`/c/file.exe`**.

Vagrant tries to detect what environment is being used and translate paths appropriately. However, there are cases where this does not work.

One way to deal with this issue is to add the following lines to the per-user Vagrantfile in the **`%USERPROFILE%\.vagrant.d\Vagrantfile`** file. Note that you will most likely need to create this file.

```
# Cygwin Rsync under CMD.EXE Workaround
ENV["VAGRANT_DETECTED_OS"] = ENV["VAGRANT_DETECTED_OS"].to_s + " cygwin"
```

If you use Cygwin's **`rsync`** under Windows **`cmd.exe`**, Vagrant is not able to tell that you are using Cygwin and supplies the wrong path to **`rsync`**. The workaround is to set the **`VAGRANT_DETECTED_OS`** variable to **`cygwin`**. For example:

```
$ setx VAGRANT_DETECTED_OS cygwin
```

An alternative is to run Vagrant inside of Cygwin's Terminal Window. This will provide a Linux-like experience with the Bash shell and a window that can be sized beyond 80 columns. In this way, you can use both Linux and Windows paths.

However, to use Windows paths, you must surround thoses paths with single quotes. For example, **`c:\User\joe`** would be interpreted by the shell as **`c:Userjoe`** without single quotes. Using **`'c:\User\joe'`**, causes the path to be interpreted correctly.

To permanently set this variable for a Bash shell, add the following line to the **.bashrc** file in the user's home directory (**~/.bashrc**):

```
$ export VAGRANT_DETECTED_OS=cygwin
```

3.3. SETTING UP CONTAINER DEVELOPMENT KIT SOFTWARE COMPONENTS

1. Unzip the ZIP file you downloaded in a directory of your choice. The following commands assume you have unpacked it in your home directory, **%USERPROFILE%\cdk** (**C:\Users\<username>\cdk**).

   At this point, review the included **README** files to familiarize yourself with Red Hat Container Tools.

2. Install additional Vagrant plugins for using the Container Development Kit Vagrant boxes.

   All of the remaining steps can be performed using the Windows command-line shell, **cmd.exe**, PowerShell, or other command-line shell.

   To install additional Vagrant plugins to support several features, use the **vagrant plugin install** command. The plugins in the form of **.gem** files are included in the ZIP file.

   The installation of the first plugin may take several minutes, and Vagrant may install some additional gem files as needed.

   ```
   C:> cd %USERPROFILE%\cdk\plugins
   C:> dir *.gem
   C:> vagrant plugin install vagrant-registration-*.gem
   C:> vagrant plugin install vagrant-service-manager-*.gem
   C:> vagrant plugin install vagrant-sshfs-*.gem
   ```

Note that the **vagrant-sshfs** plugin is dependent upon **sftp-server**. On Microsoft Windows hosts only, **vagrant-sshfs** is also dependent on **Cygwin openssh**.

1. Verify the plugins are installed by running the following command:

   ```
   C:> vagrant plugin list
   ```

For information on using the plugins after Container Development Kit is installed, see Using Vagrant Container Development Kit Plugins in the *Red Hat Container Development Kit 2.4 Getting Started Guide*.

1. Add the Container Development Environment box to Vagrant. For example:

   ```
   C:> cd %USERPROFILE%\Downloads\
   C:> vagrant box add --name cdkv2 rhel-cdk-kubernetes-7.3-
   *.x86_64.vagrant-virtualbox.box
   ```

Important

The name you assign to the box using the **`--name`** parameter in this step must correspond to the name used by the Vagrantfile used to initialize the box. By default, this is **`cdkv2`** for the Vagrantfiles provided with Container Development Kit.

To use a customized name for the box:

1. Export the desired name to the **`BOX`** environment variable
   - In **cmd.exe** or **PowerShell**:

     ```
     C:> setx BOX=<box-name>
     ```

   - In a **Cygwin** shell:

     ```
     $ export BOX=<box-name>
     ```

2. Use the name in the **`vagrant box add`** command:

   ```
   vagrant box add --name <box-name> rhel-cdk-
   kubernetes-7-*.x86_64.vagrant-_<hypervisor>_.box
   ```

   Substitute ***`<hypervisor>`*** with **`virtualbox`** or **`hyperv`**, depending on which virtualization provider you use.

2. Verify that the box is installed:

   ```
   C:> vagrant box list
   ```

   The box image files will be stored in your home directory under **`%USERPROFILE%\.vagrant.d`**. You will need adequate space there, approximately 2 GB.

3.4. SETTING UP CONTAINER DEVELOPMENT KIT FOR USE BEHIND AN HTTP PROXY

To run the Container Development Environment behind a proxy server, you need to export the proxy server information to the Container Development Environment and then run **`vagrant up`** (see Starting the Container Development Kit Vagrant Box on Microsoft Windows).

Note

Currently, only HTTP and HTTPS proxy servers are supported.

3.4.1. Setting Proxy Environment Variables Manually

Run the following commands on the host system to export environment variables with information about the proxy server. This setting only remains in effect until you restart your session:

In a **Cygwin** shell:

```
$ export PROXY="<proxy-server>:<port>"
$ export PROXY_USER="<username>"
$ export PROXY_PASSWORD="<password>"
```

In **cmd.exe** or **PowerShell**:

```
setx PROXY="<proxy-server>:<port>"
setx PROXY_USER="<username>"
setx PROXY_PASSWORD="<password>"
```

3.4.2. Using vagrant-service-manager to Set Proxy Environment Variables

To automatically export information about the proxy configuration to every Vagrant box that you start, use the **`vagrant-service-manager`** plugin. The following steps need to be completed:

1. Include the following lines in your per-user Vagrantfile (**`%USERPROFILE%\.vagrant.d\Vagrantfile`** — note that you may need to create this file):

```
Vagrant.configure(2) do |config|
  config.servicemanager.proxy = '<proxy server>:<port>'
  config.servicemanager.proxy_user = '<proxy username>'
  config.servicemanager.proxy_password = '<proxy password>'
end
```

2. Remove proxy-related configuration lines from the Container Development Kit Vagrantfile you intend to use.

   - To use the **OpenShift Container Platform** configuration, remove the following lines from the **`rhel-ose`** Vagrantfile (**`cdk/component/rhel/rhel-ose/Vagrantfile`**):

```
# explicitly enable and start OpenShift
config.vm.provision "shell", run: "always", inline: <<-SHELL
  PROXY=#{PROXY} PROXY_USER=#{PROXY_USER} PROXY_PASSWORD=#
{PROXY_PASSWORD} IMAGE_TAG=#{IMAGE_TAG} /usr/bin/sccli openshift
SHELL
```

     And modify the following line to include the service your Vagrantfile needs to start (**`openshift`**):

```
# prevent the automatic start of openshift via service-manager by
just enabling Docker
config.servicemanager.services = "docker, openshift"
```

   - To use the **Kubernetes** configuration, remove the following lines from the **`rhel-k8s-singlenode-setup`** Vagrantfile (**`cdk/components/rhel/misc/rhel-k8s-singlenode-setup/Vagrantfile`**):

```
# Explicitly enable and start kubernetes
config.vm.provision "shell", run: "always", inline: <<-SHELL
  PROXY=#{PROXY} PROXY_USER=#{PROXY_USER} PROXY_PASSWORD=#
```

```
{PROXY_PASSWORD} /usr/bin/sccli kubernetes
  echo "kubernetes single node cluster setup successfully"
SHELL
```

And modify the following line to include the service your Vagrantfile needs to start (**kubernetes**):

```
# prevent the automatic start of openshift via service-manager by
just enabling Docker
config.servicemanager.services = "docker, kubernetes"
```

Note

To set the IP address used by the Vagrant box, set the **PUBLIC_ADDRESS** variable in the Vagrantfile for that box:

```
# The private network IP of the VM. You will use this IP to
connect to OpenShift.
# PUBLIC_ADDRESS="10.1.2.2"

PUBLIC_ADDRESS="x.x.x.x"
```

3.5. STARTING THE CONTAINER DEVELOPMENT KIT VAGRANT BOX ON MICROSOFT WINDOWS

Container Development Kit offers two Vagrantfiles for initializing the Container Development Environment with different services:

- **OpenShift (`rhel-ose`)**: Use the OpenShift Vagrantfile to launch a Red Hat Enterprise Linux Server virtual machine (VM) with OpenShift Container Platform running in it. With OpenShift running, you can use either the web user interface from a web browser on your desktop, or **`docker`**, **`oc`**, and related commands by logging into the VM.
- **Kubernetes (`rhel-k8s-singlenode-setup`)**: Use the Kubernetes Vagrantfile to start a more generic Container Development Kit VM. Because OpenShift is not running, you can configure a more basic Kubernetes configuration or use Docker directly.

Note

You can modify the existing Vagrantfiles for your own purposes. For example, you might want to use a different IP address if it conflicts with an address on your local network.

In order to run the following steps, you need to have edited the **`%USERPROFILE%\.vagrant.d\Vagrantfile`** file as discussed in the Section 3.2.2, “Translating Windows Paths and rsync” section.

1. Initialize the Container Development Kit Vagrant box:
   - Start Container Development Kit with OpenShift Container Platform as follows:

```
$ cd %USERPROFILE%\cdk\components\rhel\rhel-ose\
$ vagrant up
```

To use a different version of OpenShift Container Platform than the default (3.4.0.40), edit the **IMAGE_TAG** parameter in the **rhel-ose** Vagrantfile before running **vagrant up**. You can find the list of supported version tags at https://registry.access.redhat.com/v1/repositories/openshift3/ose/tags.

For example:

```
IMAGE_TAG="v3.3.1.5"
```

- Start Container Development Kit with single-node Kubernetes as follows:

```
$ cd %USERPROFILE%\cdk\components\rhel\misc\rhel-k8s-
singlenode-setup\
$ vagrant up
```

2. Enter the username and password you use with the Red Hat Customer Portal to register the Red Hat Enterprise Linux system running in the Vagrant box:

```
==> default: Registering box with vagrant-registration…
    default: Would you like to register the system now (default:
yes)? [y|n] y
    default: Subscriber username: <username>
    default: Subscriber password:
```

3. Check whether your Vagrant box is running using the **vagrant status** command. Note that you must be in the same directory where your Vagrantfile is located. For example:

```
$ cd %USERPROFILE%\cdk\components\rhel\rhel-ose\
$ vagrant status
```

If the machine state shows as running, you are ready to start using your Container Development Environment.

CHAPTER 4. VAGRANT VERSIONS SUPPORTED IN CONTAINER DEVELOPMENT KIT 2.4 WITH OTHER CONTAINER TECHNOLOGIES:

Vagrant may support different container technologies, depending on the version of Vagrant used.

Table 4.1. Vagrant Compatibility Matrix

	Vagrant version	
	1.7.4	1.8.1
Microsoft Windows with VirtualBox	✓	✓
Microsoft Windows with Hyper-V	✘	✓
macOS with VirtualBox	✓	✓
Red Hat Enterprise Linux with libvirt	✓	✓

4.1. ADDITIONAL RESOURCES

- The Red Hat Container Development Kit 2.4 Getting Started Guide provides information on:
  - using installed Vagrant plugins
  - interacting with Vagrant boxes
  - starting with container development
  - using **Docker**, **Kubernetes**, and **OpenShift Container Platform**
- Cygwin Documentation
- Vagrant Documentation
- VirtualBox Documentation
- Hyper-V Getting Started Guide

CHAPTER 5. INSTALLING CONTAINER DEVELOPMENT KIT ON MACOS

To prepare your macOS development system for running Container Development Kit, the steps are:

1. Download and install VirtualBox.
2. Download and install Vagrant.
3. Download Red Hat Container Tools and the Container Development Kit Vagrant box for VirtualBox.
4. Install additional Vagrant plugins to support Red Hat Subscription Management and other features.

5.1. PREREQUISITES FOR INSTALLING CONTAINER DEVELOPMENT KIT ON MACOS

To run Container Development Kit on a macOS system, you need:

- An Intel-based Mac with a minimum of 4 GB of RAM running a recent, 64-bit version of macOS, such as 10.11 (El Capitan) or later.
- A minimum of 3 GB of free disk space for virtual machine images. Note that during the setup process, you will need to be able to store multiple copies of each of the virtual machine images.
- Adequate Internet connectivity to download 1—2 GB of software.
- An available Red Hat Enterprise Linux subscription with support or Red Hat Enterprise Linux Developer Suite. Note that Red Hat Enterprise Linux subscriptions with self-support do not have access to all of the necessary software for CDK in all environments.

5.2. INSTALLING SOFTWARE AND CONFIGURING THE HOST SYSTEM

Vagrant is used to run a Red Hat Enterprise Linux virtual machine with all necessary components for the Container Development Environment. Virtualization is provided by installing VirtualBox.

Please note that Vagrant is strictly command-line oriented. All interaction with Vagrant is through the command line from a terminal prompt. Use **Terminal.app** from the **Applications** → **Utilities** folder or a terminal emulator of your choice for running Vagrant.

Command-line instructions use path names with a tilde, such as **`~/cdk`**. This is a shorthand notation for a path relative to the current user's home directory, **`/Users/`*`username`*`/cdk`**.

5.2.1. Additional Software Requirements for macOS

In order to run Container Development Kit, some command-line development tools are needed on your Mac. Install either **Apple Command Line Development Tools** or **Apple Xcode**.

1. Download and install VirtualBox for macOS from http://www.virtualbox.org.

Important

Container Development Kit 2.4 is known to not work correctly with **VirtualBox 5.1.x**. If you already have **VirtualBox 5.1.x** installed, downgrade your installation to **VirtualBox 5.0.26**.

2. Choose a location for storing VirtualBox VM images. By default, these are stored in subdirectory in your home directory (**`~/VirtualBox VMs`**). You need several gigabytes of space wherever you choose to store these images. To change the location, start VirtualBox, go to **VirtualBox** → **Preferences** → **General** and change **Default Machine Folder** to the desired location. Documentation for VirtualBox can be found on the virtualbox.org website.

3. Download and install Vagrant from https://releases.hashicorp.com/vagrant/. Select the folder with the 1.8.1 release and download the Vagrant installation file in the **`.dmg`** format (**`vagrant_1.8.1.dmg`**). The 1.7.4 version of Vagrant is also supported for use with Red Hat Container Development Kit 2.4.

   The default is to install Vagrant in the **`/opt/vagrant`** directory. You can change this. The Vagrant installer creates a **`/usr/bin/vagrant`** symbolic link that points to **`/opt/vagrant/bin/vagrant`**, so no adjustments to the **`PATH`** environment variable are necessary.

5.3. SETTING UP CONTAINER DEVELOPMENT KIT SOFTWARE COMPONENTS

1. Download the Container Development Kit software components from the Red Hat Product Downloads web site. You must log in to get access to this page. If you are on the right page, you should see "Product Variant: Red Hat Container Development Kit". You need to download the following items:

   - Red Hat Container Tools
   - RHEL 7.3 Vagrant box for VirtualBox

   

   Note

   The page also offers Vagrant box downloads formatted for other virtualization platforms, such as libvirt. You only need to download the box image that matches the virtualization you are using — VirtualBox.

   The following steps assume you have saved these files in your home directory in **`~/Downloads`**. If you used a different directory, adjust the paths accordingly. You need several gigabytes of free space for the Vagrant box images.

   All of the remaining steps need to be performed using the command-line using the Terminal Application (Terminal.app). You can find Terminal.app in the **Application** → **Utilities** Folder.

2. Unzip the ZIP file you downloaded. This should create the **`~/cdk`** subdirectory:

```
$ cd
$ unzip ~/Downloads/cdk-2.4.0.zip
```

3. Install the **vagrant-registration**, **vagrant-service-manager**, and **vagrant-sshfs** plugins (the plugins in the form of **.gem** files are included in the ZIP file).

   The installation of the first plugin may take several minutes, and Vagrant may install some additional gem files as needed.

```
$ cd ~/cdk/plugins
$ ls -1 *.gem
$ vagrant plugin install \
  ./vagrant-registration-*.gem \
  ./vagrant-service-manager-*.gem \
  ./vagrant-sshfs-*.gem
```

Note that the **vagrant-sshfs** plugin is dependent upon sftp-server.

1. Verify the plugins are installed by running the following command:

```
$ vagrant plugin list
```

2. Add the Container Development Kit box to Vagrant. This is the configured virtual machine image that you downloaded in one of the previous steps.

```
$ vagrant box add --name cdkv2 ~/Downloads/rhel-cdk-kubernetes-
7.2*.x86_64.vagrant-virtualbox.box
```

Important

The name you assign to the box using the **--name** parameter in this step must correspond to the name used by the Vagrantfile used to initialize the box. By default, this is **cdkv2** for the Vagrantfiles provided with Container Development Kit.

To use a customized name for the box:

1. Export the desired name to the **BOX** environment variable

```
$ export BOX=<box-name>
```

2. Use the name in the **vagrant box add** command:

```
$ vagrant box add --name <box-name> rhel-cdk-
kubernetes-7-*.x86_64.vagrant-virtualbox.box
```

3. Verify that the box is installed:

```
$ vagrant box list
cdkv2 (virtualbox, 0)
```

The box image file will be stored in your home directory under **~/.vagrant.d**. You need adequate space there, approximately 2 GB.

5.4. SETTING UP CONTAINER DEVELOPMENT KIT FOR USE BEHIND AN HTTP PROXY

To run the Container Development Environment behind a proxy server, you need to export the proxy server information to the Container Development Environment and then run **vagrant up** (see Starting the Container Development Kit Vagrant Box on macOS).

Note

Currently, only HTTP and HTTPS proxy servers are supported.

5.4.1. Setting Proxy Environment Variables Manually

Run the following commands on the host system to export environment variables with information about the proxy server. This setting only remains in effect until you restart your session:

```
$ export PROXY="<proxy-server>:<port>"
$ export PROXY_USER="<username>"
$ export PROXY_PASSWORD="<password>"
```

5.4.2. Using vagrant-service-manager to Set Proxy Environment Variables

To automatically export information about the proxy configuration to every Vagrant box that you start, use the **vagrant-service-manager** plugin. The following steps need to be completed:

1. Include the following lines in your per-user Vagrantfile (**~/.vagrant.d/Vagrantfile** — note that you may need to create this file):

```
Vagrant.configure(2) do |config|
  config.servicemanager.proxy = '<proxy server>:<port>'
  config.servicemanager.proxy_user = '<proxy username>'
  config.servicemanager.proxy_password = '<proxy password>'
end
```

2. Remove proxy-related configuration lines from the Container Development Kit Vagrantfile you intend to use.

   - To use the **OpenShift Container Platform** configuration, remove the following lines from the **rhel-ose** Vagrantfile (**cdk/component/rhel/rhel-ose/Vagrantfile**):

```
# explicitly enable and start OpenShift
config.vm.provision "shell", run: "always", inline: <<-SHELL
  PROXY=#{PROXY} PROXY_USER=#{PROXY_USER} PROXY_PASSWORD=#
{PROXY_PASSWORD} IMAGE_TAG=#{IMAGE_TAG} /usr/bin/sccli openshift
SHELL
```

And modify the following line to include the service your Vagrantfile needs to start (**openshift**):

```
# prevent the automatic start of openshift via service-manager by
just enabling Docker
config.servicemanager.services = "docker, openshift"
```

- To use the **Kubernetes** configuration, remove the following lines from the **rhel-k8s-singlenode-setup** Vagrantfile (**cdk/components/rhel/misc/rhel-k8s-singlenode-setup/Vagrantfile**):

```
# Explicitly enable and start kubernetes
config.vm.provision "shell", run: "always", inline: <<-SHELL
  PROXY=#{PROXY} PROXY_USER=#{PROXY_USER} PROXY_PASSWORD=#
{PROXY_PASSWORD} /usr/bin/sccli kubernetes
  echo "kubernetes single node cluster setup successfully"
SHELL
```

And modify the following line to include the service your Vagrantfile needs to start (**kubernetes**):

```
# prevent the automatic start of openshift via service-manager by
just enabling Docker
config.servicemanager.services = "docker, kubernetes"
```

Note

To set the IP address used by the Vagrant box, set the **PUBLIC_ADDRESS** variable in the Vagrantfile for that box:

```
# The private network IP of the VM. You will use this IP to
connect to OpenShift.
# PUBLIC_ADDRESS="10.1.2.2"

PUBLIC_ADDRESS="x.x.x.x"
```

5.5. STARTING THE CONTAINER DEVELOPMENT KIT VAGRANT BOX ON MACOS

Container Development Kit offers two Vagrantfiles for initializing the Container Development Environment with different services:

- **OpenShift (rhel-ose)**: Use the OpenShift Vagrantfile to launch a Red Hat Enterprise Linux Server virtual machine (VM) with OpenShift Container Platform running in it. With OpenShift running, you can use either the web user interface from a web browser on your desktop, or **docker**, **oc**, and related commands by logging into the VM.
- **Kubernetes (rhel-k8s-singlenode-setup)**: Use the Kubernetes Vagrantfile to start a more generic Container Development Kit VM. Because OpenShift is not running, you can configure a more basic Kubernetes configuration or use Docker directly.

Note

You can modify the existing Vagrantfiles for your own purposes. For example, you might want to use a different IP address if it conflicts with an address on your local network.

1. Initialize the Container Development Kit Vagrant box:
   - Start Container Development Kit with OpenShift Container Platform as follows:

     ```
     $ cd ~/cdk/components/rhel/rhel-ose/
     $ vagrant up
     ```

     To use a different version of OpenShift Container Platform than the default (3.4.0.40), edit the **`IMAGE_TAG`** parameter in the **`rhel-ose`** Vagrantfile before running **`vagrant up`**. You can find the list of supported version tags at https://registry.access.redhat.com/v1/repositories/openshift3/ose/tags.

     For example:

     ```
     IMAGE_TAG="v3.3.1.5"
     ```

   - Start Container Development Kit with single-node Kubernetes as follows:

     ```
     $ cd ~/cdk/components/rhel/misc/rhel-k8s-singlenode-setup/
     $ vagrant up
     ```

2. Enter the username and password you use with the Red Hat Customer Portal to register the Red Hat Enterprise Linux system running in the Vagrant box:

   ```
   ==> default: Registering box with vagrant-registration…
       default: Would you like to register the system now (default:
   yes)? [y|n] y
       default: Subscriber username: <username>
       default: Subscriber password:
   ```

3. Check whether your Vagrant box is running using the **`vagrant status`** command. Note that you must be in the same directory where your Vagrantfile is located. For example:

   ```
   $ cd ~/cdk/components/rhel/rhel-ose/
   $ vagrant status
   ```

If the machine state shows as running, you are ready to start using your Container Development Environment.

CHAPTER 6. VAGRANT VERSIONS SUPPORTED IN CONTAINER DEVELOPMENT KIT 2.4 WITH OTHER CONTAINER TECHNOLOGIES:

Vagrant may support different container technologies, depending on the version of Vagrant used.

Table 6.1. Vagrant Compatibility Matrix

	Vagrant version	
	1.7.4	1.8.1
Microsoft Windows with VirtualBox	✓	✓
Microsoft Windows with Hyper-V	✘	✓
macOS with VirtualBox	✓	✓
Red Hat Enterprise Linux with libvirt	✓	✓

6.1. ADDITIONAL RESOURCES

- The Red Hat Container Development Kit 2.4 Getting Started Guide provides information on:
  - using installed Vagrant plugins
  - interacting with Vagrant boxes
  - starting with container development
  - using **Docker**, **Kubernetes**, and **OpenShift Container Platform**
- Vagrant Documentation
- VirtualBox Documentation

CHAPTER 7. INSTALLING CONTAINER DEVELOPMENT KIT ON RED HAT ENTERPRISE LINUX

This chapter provides instructions on how to install and configure Red Hat Container Development Kit components and dependencies on Red Hat Enterprise Linux.

7.1. PREREQUISITES FOR INSTALLING CONTAINER DEVELOPMENT KIT ON RED HAT ENTERPRISE LINUX

To run Container Development Kit, you need:

- A 64-bit machine with a minimum of 4 GB of RAM. At least 2 GB should be reserved for Container Development Kit, with 3—4GB reserved for Container Development Kit being more reasonable if you plan to run multiple virtual machines. The Kubernetes Vagrantfile defines 1GB of disk space while the OSE Vagrantfile asks for up to 3GB of disk space.
- A minimum of 3 GB of free disk space for virtual machine images. Note that during the setup process, you will need to be able to store multiple copies of each of the virtual machine images.
- Adequate Internet connectivity to download 1—2 GB of software.
- An available Red Hat Enterprise Linux subscription. Note that Red Hat Enterprise Linux subscriptions with self-support do not have access to all of the necessary software for Container Development Kit in all environments.

The following steps outline the installation procedure:

1. Enable virtualization support in your computer's BIOS.
2. Install the host operating system.
3. Install the KVM and libvirt virtualization software.
4. Install Vagrant.
5. Download Red Hat Container Tools and the Container Development Kit Vagrant box.
6. Install additional Vagrant plugins to support Red Hat Subscription Management and other features.
7. Start the Vagrant box using a Vagrantfile.

If any of the steps have already been performed (such the installation of the host operating system), they do not need to be repeated.

7.2. INSTALLING SOFTWARE AND CONFIGURING THE HOST SYSTEM

Vagrant is used to run a Red Hat Enterprise Linux virtual machine with all necessary components of Container Development Kit included in it. Virtualization is provided by using the native Linux kernel-based virtual machine (KVM) hypervisor and libvirt.

A Vagrant plugin is installed to enable using libvirt as one of Vagrant virtualization providers. If you are not familiar with KVM and libvirt, see the Red Hat Enterprise Linux Virtualization Getting Started Guide.

You need root privileges to install the necessary software and perform configuration on your development host. Once Vagrant and libvirt are installed and correctly configured, you complete the preparation as a regular, non-root user. You will be able to start, stop, and configure Vagrant boxes using your regular user account.

Take care to use the same user ID for running Vagrant. Because Vagrant stores configuration components, such as plugins and downloaded box images, in the user's home directory (**`~/.vagrant.d`**), if you change to a different user ID, you will need to repeat the steps that install Vagrant plugins and boxes.

Do not run Vagrant when you are logged in as the root use, to avoid creating problems with file permissions.

7.2.1. Installing Virtualization and Container Development Kit Components

The following steps need to be completed as root:

1. If you have not already done so, install the host system directly on hardware or on a virtual machine that is set to act as a hypervisor.
2. Make sure your host Red Hat Enterprise Linux system is registered using your Red Hat username and password.
3. Enable required software repositories.
4. Install and configure the libvirt virtualization environment.
5. Install Vagrant and configure the development host for running Vagrant boxes.

7.2.1.1. Registering a Red Hat Enterprise Linux System and Enabling Repositories

1. Register Red Hat Enterprise Linux and enable required system repositories:

   You need access to the Red Hat Software Collections (**`rhel-<variant>-rhscl-7-rpms`**) repository. See Getting Access to Red Hat Software Collections.

   ```
   # subscription-manager register --auto-attach --
   username=<username> --password=<password>
   # subscription-manager repos --enable rhel-<variant>-rhscl-7-rpms
   # subscription-manager repos --enable rhel-7-<variant>-optional-
   rpms
   ```

   In the above examples, replace **`<variant>`** with **`server`** or **`workstation`**, depending on which variant of Red Hat Enterprise Linux you are using.

2. Download and unpack the **`centos-release-scl`** package from CentOS.

   The contents of the package will be used to enable the Software Collections repository and add a GPG key to enable the checking of the validity of installed RPM packages:

Note

Because Vagrant is not officially packaged for Red Hat Enterprise Linux, it needs to be installed using a Software Collection packaged for CentOS.

```
~]$ mkdir Downloads
~/Downloads]$ wget
http://mirror.centos.org/centos/7/extras/x86_64/Packages/centos-
release-scl-2-2.el7.centos.noarch.rpm
~/Downloads]$ rpm2cpio ./centos-release-scl-2-
2.el7.centos.noarch.rpm | cpio -idmv
```

3. As the **root** user, enable the Software Collections repository:

```
/home/joe/Downloads]# cp etc/yum.repos.d/CentOS-SCLo-scl.repo
/etc/yum.repos.d/
```

4. As the **root** user, copy the Software Collections GPG key to the directory where **yum** will be able to use it:

```
/home/joe/Downloads]# cp etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-
SCLo /etc/pki/rpm-gpg/
```

5. Update your system using **yum update**. If a new kernel is installed during the update, reboot your system before proceeding with the remaining steps.

7.2.1.2. Installing and Initializing Virtualization Software

1. Install virtualization software.
   - On Red Hat Enterprise Linux Server, install the **Virtualization Host** package group:

     ```
     # yum groupinstall "Virtualization Host"
     ```

   - On Red Hat Enterprise Linux Workstation, install the following package groups:

     ```
     # yum groupinstall base, core, virtualization-hypervisor,
     virtualization-tools
     # yum install libvirt-devel rpm-build zlib-devel ruby-devel
     rh-ruby22-ruby-devel gcc-c++
     ```

2. Launch the **libvirtd** daemon and configure it to start at boot, run:

```
# systemctl start libvirtd
# systemctl enable libvirtd
```

3. Install Vagrant and other required packages, including the **vagrant-libvirt** plugin:

```
# yum install sclo-vagrant1 sclo-vagrant1-vagrant-libvirt \
  sclo-vagrant1-vagrant-libvirt-doc
```

Note

Before the packages are installed, you will be asked to confirm the importing of the Software Collections GPG key that was installed previously.

4. Allow your regular user ID to start and stop Vagrant boxes. A PolicyKit rule will be added that allows users in the **vagrant** group to control VMs through libvirt. The necessary rule is included in the **`vagrant-libvirt`** package you just installed. Run the following command to add the rule on your system:

```
# cp /opt/rh/sclo-
vagrant1/root/usr/share/vagrant/gems/doc/vagrant-libvirt-
*/polkit/10-vagrant-libvirt.rules \
  /etc/polkit-1/rules.d
```

The following is the contents of the newly created **`/etc/polkit-1/rules.d/10-vagrant-libvirt.rules`** rule for reference, or if you prefer to add it by hand:

The `/etc/polkit-1/rules.d/10-vagrant-libvirt.rules` file

```
/*
 * Allow users in vagrant group to manage libvirt without
authentication.
 * Copy this file to /usr/share/polkit-1/rules.d/ to activate.
 */
polkit.addRule(function(action, subject) {
  if ((action.id == "org.libvirt.unix.manage"
    || action.id == "org.libvirt.unix.monitor")
    && subject.isInGroup("vagrant")) {
    return polkit.Result.YES;
  }
});
```

5. Restart the libvirt and PolicyKit services for the changes to take effect:

```
# systemctl restart libvirtd
# systemctl restart polkit
```

6. Check your user name, then become root to add your user name to the **vagrant** group. (Note that you need to log out and back in for the change to your group membership to take affect.)

```
$ echo $USER
joe
$ su -
Password:
# usermod -a -G vagrant joe
```

That completes the list of steps that need to be performed as root.

7.3. SETTING UP CONTAINER DEVELOPMENT KIT SOFTWARE COMPONENTS

The remaining steps should be performed under your regular non-root user ID. This should be the same user ID you added to the **vagrant** group.

1. Verify you are a member of the **vagrant** group (you may need to log in again to have the new group appear).

```
$ grep vagrant /etc/group
vagrant:x:978:joe
$ id
uid=1001(joe) gid=1001(joe) groups=1001(joe),978(vagrant)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

2. Enable the Vagrant Software Collection:

```
$ scl enable sclo-vagrant1 bash
```

3. Verify that the system is set up to run Vagrant as a regular, non-root user:

```
$ vagrant global-status
id       name   provider state  directory
-----------------------------------------------------------
-------
There are no active Vagrant environments on this computer! Or,
you haven't destroyed and recreated Vagrant environments that
were
started with an older version of Vagrant.
```

4. Download the Container Development Kit software components from the Red Hat Product Downloads web site. You must log in to get access to this page. If you are on the right page, you should see **Product Variant: Red Hat Container Development Kit**. You need to download the following items:

   - Red Hat Container Tools
   - RHEL 7.3 Vagrant box for libvirt

Note

The download page also offers Vagrant **`.box`** files formatted for other virtualization platforms, such as VirtualBox. You only need to download the **`.box`** image that matches the virtualization you are using.

5. Unzip the downloaded ZIP file:

Note

The following steps assume you have saved the downloaded files in your home directory in **`~/Downloads`**. If you used a different directory, adjust the paths accordingly.

```
~/Downloads]$ unzip ~/Downloads/cdk-2.4.0.zip
```

At this point, review the included **README** files to familiarize yourself with Red Hat Container Tools.

6. Install the plugins contained in the unpacked **cdk/plugins/** directory. Each plugin is in the form of a **.gem** file:

```
~/Downloads]$ cd ~/cdk/plugins/
~/cdk/plugins]$ ls -1 *.gem
~/cdk/plugins]$ vagrant plugin install \
  ./vagrant-registration-*.gem \
  ./vagrant-service-manager-*.gem \
  ./vagrant-sshfs-*.gem
```

Note that the **vagrant-sshfs** plugin is dependent upon **sftp-server**.

1. Verify that the **vagrant-libvirt**, **vagrant-registration**, **vagrant-service-manager**, and **vagrant-sshfs** plugins are properly installed by running the following command:

```
$ vagrant plugin list
```

2. Add the Container Development Kit Vagrant box to Vagrant:

```
$ vagrant box add --name cdkv2 \
  ~/Downloads/rhel-cdk-kubernetes-7.*.x86_64.vagrant-libvirt.box
```

Important

The name you assign to the box using the **--name** parameter in this step must correspond to the name used by the Vagrantfile used to initialize the box. By default, this is **cdkv2** for the Vagrantfiles provided with Container Development Kit.

To use a customized name for the box:

1. Export the desired name to the **BOX** environment variable

```
$ export BOX=<box-name>
```

2. Use the name in the **vagrant box add** command:

```
$ vagrant box add --name <box-name> rhel-cdk-
kubernetes-7-*.x86_64.vagrant-libvirt.box
```

3. Verify that the box is installed:

```
$ vagrant box list
cdkv2 (libvirt, 0)
```

The box image file will be stored in your home directory under **~/.vagrant.d**. You need adequate space there, approximately 3 GB.

7.4. SETTING UP CONTAINER DEVELOPMENT KIT FOR USE BEHIND AN HTTP PROXY

To run the Container Development Environment behind a proxy server, you need to export the proxy server information to the Container Development Environment and then run **vagrant up** (see Starting the Container Development Kit Vagrant Box on Red Hat Enterprise Linux).

Note

Currently, only HTTP and HTTPS proxy servers are supported.

7.4.1. Setting Proxy Environment Variables Manually

Run the following commands on the host system to export environment variables with information about the proxy server. This setting only remains in effect until you restart your session:

```
$ export PROXY="<proxy-server>:<port>"
$ export PROXY_USER="<username>"
$ export PROXY_PASSWORD="<password>"
```

7.4.2. Using vagrant-service-manager to Set Proxy Environment Variables

To automatically export information about the proxy configuration to every Vagrant box that you start, use the **vagrant-service-manager** plugin. The following steps need to be completed:

1. Include the following lines in your per-user Vagrantfile (**~/.vagrant.d/Vagrantfile** — note that you may need to create this file):

```
Vagrant.configure(2) do |config|
  config.servicemanager.proxy = '<proxy server>:<port>'
  config.servicemanager.proxy_user = '<proxy username>'
  config.servicemanager.proxy_password = '<proxy password>'
end
```

2. Remove proxy-related configuration lines from the Container Development Kit Vagrantfile you intend to use.

   - To use the **OpenShift Container Platform** configuration, remove the following lines from the **rhel-ose** Vagrantfile (**cdk/component/rhel/rhel-ose/Vagrantfile**):

```
# explicitly enable and start OpenShift
config.vm.provision "shell", run: "always", inline: <<-SHELL
  PROXY=#{PROXY} PROXY_USER=#{PROXY_USER} PROXY_PASSWORD=#
{PROXY_PASSWORD} IMAGE_TAG=#{IMAGE_TAG} /usr/bin/sccli openshift
SHELL
```

     And modify the following line to include the service your Vagrantfile needs to start

(**openshift**):

```
# prevent the automatic start of openshift via service-manager by
just enabling Docker
config.servicemanager.services = "docker, openshift"
```

- To use the **Kubernetes** configuration, remove the following lines from the **rhel-k8s-singlenode-setup** Vagrantfile (**cdk/components/rhel/misc/rhel-k8s-singlenode-setup/Vagrantfile**):

```
# Explicitly enable and start kubernetes
config.vm.provision "shell", run: "always", inline: <<-SHELL
  PROXY=#{PROXY} PROXY_USER=#{PROXY_USER} PROXY_PASSWORD=#
{PROXY_PASSWORD} /usr/bin/sccli kubernetes
  echo "kubernetes single node cluster setup successfully"
SHELL
```

And modify the following line to include the service your Vagrantfile needs to start (**kubernetes**):

```
# prevent the automatic start of openshift via service-manager by
just enabling Docker
config.servicemanager.services = "docker, kubernetes"
```

Note

To set the IP address used by the Vagrant box, set the **PUBLIC_ADDRESS** variable in the Vagrantfile for that box:

```
# The private network IP of the VM. You will use this IP to
connect to OpenShift.
# PUBLIC_ADDRESS="10.1.2.2"

PUBLIC_ADDRESS="x.x.x.x"
```

7.5. STARTING THE CONTAINER DEVELOPMENT KIT VAGRANT BOX ON RED HAT ENTERPRISE LINUX

Container Development Kit offers two Vagrantfiles for initializing the Container Development Environment with different services:

- **OpenShift (rhel-ose)**: Use the OpenShift Vagrantfile to launch a Red Hat Enterprise Linux Server virtual machine (VM) with OpenShift Container Platform running in it. With OpenShift running, you can use either the web user interface from a web browser on your desktop, or **docker**, **oc**, and related commands by logging into the VM.
- **Kubernetes (rhel-k8s-singlenode-setup)**: Use the Kubernetes Vagrantfile to start a more generic Container Development Kit VM. Because OpenShift is not running, you can configure a more basic Kubernetes configuration or use Docker directly.

Note

You can modify the existing Vagrantfiles for your own purposes. For example, you might want to use a different IP address if it conflicts with an address on your local network.

1. Initialize the Container Development Kit Vagrant box:
   - Start Container Development Kit with OpenShift Container Platform as follows:

     ```
     $ cd ~/cdk/components/rhel/rhel-ose/
     $ vagrant up
     ```

     To use a different version of OpenShift Container Platform than the default (3.4.0.40), edit the **`IMAGE_TAG`** parameter in the **`rhel-ose`** Vagrantfile before running **`vagrant up`**. You can find the list of supported version tags at https://registry.access.redhat.com/v1/repositories/openshift3/ose/tags.

     For example:

     ```
     IMAGE_TAG="v3.3.1.5"
     ```

   - Start Container Development Kit with single-node Kubernetes as follows:

     ```
     $ cd ~/cdk/components/rhel/misc/rhel-k8s-singlenode-setup/
     $ vagrant up
     ```

2. Enter the username and password you use with the Red Hat Customer Portal to register the Red Hat Enterprise Linux system running in the Vagrant box:

   ```
   ==> default: Registering box with vagrant-registration…
       default: Would you like to register the system now (default:
   yes)? [y|n] y
       default: Subscriber username: <username>
       default: Subscriber password:
   ```

3. Check whether your Vagrant box is running using the **`vagrant status`** command. Note that you must be in the same directory where your Vagrantfile is located. For example:

   ```
   $ cd ~/cdk/components/rhel/rhel-ose/
   $ vagrant status
   ```

If the machine state shows as running, you are ready to start using your Container Development Environment.

CHAPTER 8. VAGRANT VERSIONS SUPPORTED IN CONTAINER DEVELOPMENT KIT 2.4 WITH OTHER CONTAINER TECHNOLOGIES:

Vagrant may support different container technologies, depending on the version of Vagrant used.

Table 8.1. Vagrant Compatibility Matrix

	Vagrant version	
	1.7.4	1.8.1
Microsoft Windows with VirtualBox	✓	✓
Microsoft Windows with Hyper-V	✘	✓
macOS with VirtualBox	✓	✓
Red Hat Enterprise Linux with libvirt	✓	✓

8.1. ADDITIONAL RESOURCES

- The Red Hat Container Development Kit 2.4 Getting Started Guide provides information on:
  - using installed Vagrant plugins
  - interacting with Vagrant boxes
  - starting with container development
  - using **Docker**, **Kubernetes**, and **OpenShift Container Platform**
- libvirt Documentation
- Vagrant Documentation

CHAPTER 9. UNINSTALLING OR REINSTALLING CONTAINER DEVELOPMENT KIT

Container Development Kit does not offer any automated uninstallation mechanism. To unistall or reinstall the software, the following steps need to be performed:

1. Destroy the Container Development Environment virtual machine, and remove its Vagrant box.
2. Remove the **cdk/** directory created by unpacking the Red Hat Container Tools ZIP file.
3. Only on Linux: remove the Container Development Environment storage image from **libvirt**.

9.1. REMOVING VAGRANT BOXES

1. Check for existing Vagrant virtual machines.

   Use the **global-status** command to display all virtual machines administered by Vagrant for your account:

   ```
   $ vagrant global-status
   id       name    provider state   directory
   ---------------------------------------------------------------
   ----------
   e781364  default libvirt running
   /home/test/cdk/components/rhel/rhel-ose

   The above shows information about all known Vagrant environments
   on this machine. This data is cached and may not be completely
   up-to-date. To interact with any of the machines, you can go to
   that directory and run Vagrant, or you can use the ID directly
   with Vagrant commands from any directory. For example:
   "vagrant destroy 1a2b3c4d"
   ```

2. Destroy existing Container Development Environment virtual machines.

   Destroy the virtual machine identified in the previous step. Specify the **-f** or **--force** option to destroy the machine without an interactive confirmation:

   ```
   $ vagrant destroy -f e781364
   ```

3. List installed Vagrant boxes:

   ```
   $ vagrant box list
   cdkv2 (virtualbox, 0)
   ```

4. Remove the Container Development Environment Vagrant box.

Important

Before you proceed with uninstallation, make sure to back up any work you did within the Container Development Environment. This may include, for example, pushing the container images you prepared to a remote registry, copying files to the home directory that is synchronized to your host system, or exporting OpenShift projects.

Remove the Container Development Environment Vagrant box (named **cdkv2** by default):

```
$ vagrant box remove cdkv2
Removing box 'cdkv2' (v0) with provider 'libvirt'...
Vagrant-libvirt plugin removed box only from you LOCAL
~/.vagrant/boxes directory
From libvirt storage pool you have to delete image manually
(virsh, virt-manager or by any other tool)
```

Note

With Red Hat Container Development Kit 2.4, the **vagrant box remove cdkv2** command fails to remove **Hyper-V** Vagrant boxes. To work around this issue when you wish to use a different **Hyper-V** box, you need to forcefully overwrite the existing box:

```
$ vagrant box add cdkv2 new-hyper-v-box.img
```

9.2. REMOVING RED HAT CONTAINER TOOLS

Red Hat Container Tools, which is the contents of the **cdk-2.4.0.zip** file, can be removed by deleting the **cdk/** directory, including its subdirectories.

Note

When reinstalling the Container Development Environment without updating Red Hat Container Tools, skip this step and only install a different Vagrant box.

1. Remove the Red Hat Container Tools ZIP file (**cdk-2.4.0.zip**) and the **cdk/** directory. This example assumes that the Red Hat Container Tools ZIP file was downloaded to the **Downloads/** directory and unpacked directly to the home directory:

   - Run the following commands on Linux or macOS:

     ```
     $ rm -rf ~/cdk/
     $ rm ~/Downloads/cdk-2.4.0.zip
     ```

   - On Microsoft Windows, run the following commands:

     ```
     $ rmdir %USERPROFILE%\cdk\ /s /q
     $ del %USERPROFILE%\Downloads\cdk-2.4.0.zip
     ```

2. Uninstall the Vagrant plugins supplied with Container Development Kit:

```
$ vagrant plugin uninstall vagrant-registration vagrant-service-
manager vagrant-sshfs
Uninstalling the 'vagrant-registration' plugin...
Uninstalling the 'vagrant-service-manager' plugin...
Uninstalling the 'vagrant-sshfs' plugin...
```

9.3. DELETING LIBVIRT STORAGE IMAGES

On Linux, when the **libvirt** hypervisor is used, it is necessary to manually delete the storage image that was created for the Container Development Environment virtual machine.

In the following procedure, the **`rhel-ose_default.img`** file name is used. Substitute this file name with the **libvirt** image file name used on your system in all steps of this procedure.

1. List all storage images to identify the one you need to delete:

```
# virsh vol-list default
 Name                 Path
------------------------------------------------------------
---------
 rhel-ose_default.img /var/lib/libvirt/images/rhel-
ose_default.img
```

2. Change the ownership of the image file:

```
# chown root:root /var/lib/libvirt/images/rhel-ose_default.img
```

3. Output volume information:

```
# virsh vol-dumpxml rhel-ose_default.img --pool default
```

4. Delete the image:

   This can be done in two alternative ways. Use the second one if the first one does not work.

   - Use the **`vol-delete`** command with **`virsh`**:

```
# virsh vol-delete rhel-ose_default.img --pool default
```

   - Alternatively, delete the file directly and refresh the volume pool:

```
# rm /var/lib/libvirt/images/rhel-ose_default.img
# virsh pool-refresh default
```

5. Restart the **libvirtd** daemon:

```
# systemctl restart libvirtd
```

9.4. REINSTALLING CONTAINER DEVELOPMENT KIT

After completing the uninstallation of Container Development Kit, start a new installation by unpacking the Red Hat Container Tools ZIP file and adding a new Container Development Environment Vagrant box as described in:

- Chapter 3, *Installing Container Development Kit on Microsoft Windows*
- Chapter 5, *Installing Container Development Kit on macOS*
- Chapter 7, *Installing Container Development Kit on Red Hat Enterprise Linux*

9.5. ADDITIONAL RESOURCES

- See Chapter 10, *Troubleshooting Container Development Kit Problems* of this Installation Guide for additional information about uninstallation issues.

CHAPTER 10. TROUBLESHOOTING CONTAINER DEVELOPMENT KIT PROBLEMS

Use the information in this chapter to troubleshoot Vagrant Container Development Kit in general. See the Red Hat Container Development Kit 2.4 Release Notes and Known Issues to learn about the main features of the product and about problems that you may encounter when using this version.

10.1. TROUBLESHOOTING GENERAL PROBLEMS

How do I start Vagrant?

Vagrant is strictly command-line oriented. There are no menu entries. To run Vagrant, launch a command prompt or a terminal session on your machine. Vagrant should be automatically added to your path, so you only need to type **`vagrant`**.

Something failed during `vagrant up` or it complained that an SSH command failed. If I try `vagrant up` again, it says the VM is already running.

There are a number of provisioning steps performed when Vagrant launches a virtual machine. These may come from plugins, such as the Vagrant Registration plugin for Red Hat, or from steps that are included in the Vagrantfile for provisioning the box on startup. Vagrant runs these commands by using SSH.

In many cases, errors during the provisioning step are not fatal, so the Vagrant box will still be running. Use the **`vagrant status`** command to see what the state of your box is. You can stop the Vagrant box with the **`vagrant halt`** command. If the machine is running, you should be able to log into it using the **`vagrant ssh`** command to examine the VM and see what went wrong.

While the box is running, you can rerun the provisioning steps by running the **`vagrant provision`** command.

Note that you need to be in the same directory where your Vagrantfile is. If you have lost track of where your Vagrantfile is, use the **`vagrant global-status`** command to list the boxes that you have started and the directory where the Vagrantfile and state is stored.

The `vagrant up` command fails with error message `could not find capabilities for domaintype=kvm`.

Any system running the CDK (Windows, Mac, or RHEL), requires not only that the computer it is running on have *Virtualization Support*, but also that the support be enabled in the BIOS. On a Red Hat Enterprise Linux system, if *Virtualization Support* is not enabled when you run **`vagrant up`**, you will see a message similar to the following:

```
Error while creating domain: Error saving the server: Call to
virDomainDefineXML failed: invalid argument: could not find
capabilities for domaintype=kvm
```

Enabling *Virtualization Support* in the computer's BIOS should fix the problem.

On Red Hat Enterprise Linux, run the **`virt-host-validate`** command to check whether or not virtualization has been enabled on the computer.

The output of `vagrant up` indicates that OpenShift failed to start when using the `rhel-ose`

Vagrantfile.

Due to internal configuration problems of the Container Development Kit box, OpenShift start may time out. To repeat the provisioning steps, and thus start OpenShift in the Container Development Environment, run:

```
$ vagrant provision
```

Red Hat registration failed, or I entered the wrong username and password. How can I manually register the box or check my subscription status?

You can run the **`subscription-manager`** tool to log in, register or unregister the box, check your subscription status, or enable available software repositories as you would on any other Red Hat Enterprise Linux installation. However, you need to use the **`sudo -i`** command to be root because the root password for the Red Hat Enterprise Linux Vagrant boxes is not distributed. The **sudo** utility has been setup for the vagrant user to run any commands.

For more information, see Red Hat Subscription Management.

How do I free up my Red Hat subscription used by the Vagrant box?

The Vagrant Registration plugin automatically detaches the box from the Red Hat subscription when you shut down the box using the **`vagrant halt`** or **`vagrant destroy`** commands. If the box is not shut down by Vagrant, this does not happen. If you still have the box set up, you should be able to bring the box up again using the **`vagrant up`** command and then shut it down correctly with **`vagrant halt`** or **`vagrant destroy`**, which will unregister the box from Red Hat Subscription Management.

Alternatively, you can use the subscription management on the Red Hat Customer Portal to find and delete the virtual system that is no longer being used.

I need to make changes as root on the Vagrant boxes, what is the root password?

The **sudo** utility has been set up for the **vagrant** user to run any commands as root. You can use the **sudo** command to run a single command as root, or use **`sudo -i`** to start an interactive root shell.

Running `vagrant global-status` gives me a `permission denied` error.

Run the **`id`** command to check whether your regular user ID is a member of the**`vagrant`** group. If you skipped this step before, log out and log back in for the changes to take effect. Adding the user to the **`vagrant`** group works in conjuction with the PolicyKit rule that was added during the steps run under root to allow non-root users to work with libvirt.

10.2. TROUBLESHOOTING PROBLEMS WITH LIBVIRT

When I try to run Vagrant, I get an error message about connecting to libvirt. Do I need to be root to run Vagrant?

One of the steps in section Section 7.2.1.2, “Installing and Initializing Virtualization Software” installs a PolicyKit rule in the **`/etc/polkit-1/rules.d/`** directory that allows non-root users to run Vagrant and perform libvirt operations that are normally restricted to root users as long as the user is a member of the **vagrant** group. The error message is:

```
Error while connecting to libvirt: Error making a connection to
libvirt URI qemu:///system:
Call to virConnectOpen failed: authentication failed: no agent is
available to authenticate
```

To resolve this, make sure your user ID is a member of the **vagrant** group. You can do this by running the **id** command. Review the installation steps listed above to verify that the PolicyKit rule is installed in **/etc/polkit-1/rules.d/**. Restart the PolicyKit and libvirtd services for the changes to take effect:

```
# systemctl restart libvirtd
# systemctl restart polkit
```

The `vagrant` command is failing because libvirt says the resource it is trying to create already exists.

Normally, Vagrant takes care of all interaction with libvirt, creating and destroying resources as necessary. Running the **vagrant destroy** command should free up any resources that were allocated to an environment. However, if the Vagrant state directory, **.vagrant** is deleted, or if a Vagrant operation is interrupted, before it can clean up, it is possible that libvirt resources are left around that cannot be removed using **vagrant destroy**. If that happens, you need to use the **virsh** command-line utility or **virt-manager**, a graphical tool, to clean up.

Note that these tools must be run as root (or with the sudo command) to see all of the libvirt resources on the system. If you do not run them as root, it appears that there are no allocated resources.

The `vagrant up` command fails with `Name rhel-ose_default of domain about to create is already taken`.

If there is no volume of such name (check by running **virsh vol-list default**), you can undefine this name in libvirt:

```
# virsh undefine rhel-ose_default
```

I want to install an updated Vagrant box, but when I run it, I keep seeing the older box still being used.

For Container Development Kit with libvirt, old Vagrant boxes are not automatically removed when the new one is installed. As a result, even after adding a new box, the old one is used in place of the new box.

To add a new Vagrant box for an updated Container Development Kit, you need to destroy the running boxes, delete the **.vagrant/** directory in each directory where you ran **vagrant up** and remove the image created by each box.

If you ran both the OpenShift and Kubernetes Vagrantfiles, delete the respective virtual machines completely before adding the updated box. First go to each directory to stop the running box and remove the **.vagrant/** directory.

Warning

The **`rm -rf`** command is destructive. Be absolutely sure you identify the proper directory to delete before running this command.

```
$ cd ~/cdk/components/rhel/misc/rhel-k8s-singlenode-setup
$ vagrant destroy && rm -rf .vagrant/
$ cd ~/cdk/components/rhel/rhel-ose
$ vagrant destroy && rm -rf .vagrant/
```

Next remove the images representing the Vagrant boxes in the libvirt images directory and restart libvirt.

```
# rm -rf /var/lib/libvirt/images/*_0.img
# systemctl restart libvirtd
```

You can now proceed to adding the new Vagrant box.

10.3. TROUBLESHOOTING PROBLEMS ON MICROSOFT WINDOWS

The `vagrant ssh` command is not working, it cannot find `ssh.exe`.

In order to use **`vagrant ssh`** on a Microsoft Windows system, you need to have **`ssh.exe`** in your path. It is recommended to install **Cygwin** and ensure that **`ssh.exe`** is installed and in your path.

The `vagrant up` command refused to run because `rsync` is not in the path.

If **`rsync`** is being used for Vagrant synchronized folders on a Microsoft Windows system because it was explicitly selected, Vagrant does not start a Vagrant box unless **`rsync.exe`** is available in the path. It is recommended to install **Cygwin** and ensure that **`rsync.exe`** is in your path.

Something failed during `vagrant up` or it complained that an SSH command failed. If I try `vagrant up` again, it says the VM is already running.

There are a number of provisioning steps performed when Vagrant launches a virtual machine. These may come from plugins, such as the **`vagrant-registration`** plugin, or from steps that are included in the Vagrantfile for provisioning the box on startup. Vagrant runs these commands by using SSH.

In many cases, errors during the provisioning step are not fatal, so the Vagrant box is still running. Use the **`vagrant status`** command to see what the state of your box is. You can stop the Vagrant box with the **`vagrant halt`** command. If the machine is running, you should be able to log into it using the **`vagrant ssh`** command to examine the VM and see what went wrong.

While the box is running, you can rerun the provisioning steps by running the **`vagrant provision`** command.

Note that you need to be in the same directory where your Vagrant file is. If you have lost track of where your Vagrantfile is, use the **`vagrant global-status`** command to list all the boxes that you have started and the directory where the Vagrantfile and state is stored.

Errors upon upgrading Vagrant from 1.7.4 to 1.8.1.

The **`C:/Users/cdk/.vagrant.d/`** folder needs to be removed when upgrading from Vagrant 1.7.4 to Vagrant 1.8.1. After uninstalling the 1.7.4 version, remove the folder and the install the 1.8.1 version.

10.4. ADDITIONAL RESOURCES

- The Red Hat Container Development Kit 2.4 Release Notes and Known Issues contains information about the current release of the product as well as a list of known problems that users may encounter when using it.
- Report issues with Red Hat Container Development Kit or request new features using the **CDK** project at https://issues.jboss.org/projects/CDK.