



Red Hat
Partner Connect

CHOOSING CHOOSING AN AN AN AN AN AN AUTOMATION AU TOOL TOOL TOOL T

When to use Ansible, Helm, or Kubernetes Operators



Table of contents

Introduction	2
Chapter 1:	
Helm	4
What is Helm?	4
What is Helm used for?	4
Helm technical details	6
Chapter 2:	
Kubernetes Operators	8
What are Kubernetes Operators?	8
What are operators used for?	9
Kubernetes operator pattern technical details	10
Implementing a Kubernetes operator	10
Chapter 3:	
Ansible	12
What is Ansible?	12
What is Ansible used for?	13
Ansible technical details	15
Using Ansible with Kubernetes and Red Hat OpenShift	16
Chapter 4:	
Selecting an automation solution by use case	17
Chapter 5:	
Summary	19
Chapter 6:	
How Red Hat can help	20

Introduction

This e-book is intended to help software developers and partners choose between Helm, Kubernetes Operators, and Ansible® as an automation tool.

Why automate?

There are a number of reasons for software developers and partners to provide automation as part of the software they produce:

- ▶ **Increasing software complexity.**
The complexity of application landscapes has increased dramatically. Applications are expected to have a high level of integration with internal and external systems. Big data, artificial intelligence, and machine learning applications are often an order of magnitude larger in scale than previous generations of applications.
- ▶ **Deployment flexibility.**
Enterprises demand the flexibility to run software on site or in public and private clouds to optimize costs and performance. The deployment of an application can span multiple hybrid environments.
- ▶ **Increase software consumer satisfaction.**
Automation can improve the experience of installing and operating software. Making installations and pilot deployments smoother can shorten the sales cycle and increase sales.
- ▶ **Reduce support costs and improve productivity.**
The cost of supporting software can be reduced by using automation to avoid or detect operator errors and common configuration problems, resulting in fewer support cases. Reducing support cases for common problems can allow staff to be more focused on solving new problems and improving the quality of the software. Automation can also reduce the amount of training required for staff at the software consumer's site.
- ▶ **Higher levels of automation are expected by software consumers.**
[Containers](#), [Kubernetes](#), [cloud-native applications](#), and the prevalence of [software-as-a-service](#) solutions have raised people's expectations around responsiveness and service levels to a point where high degrees of automation are a requirement rather than a luxury. Enterprises that have been implementing [DevOps](#), [infrastructure as code](#), and [GitOps](#) practices expect their software providers to also adopt modern methodologies.

What to automate

For developers looking to add automation to their projects, automating installation is a natural starting point. The process of installation and initial configuration is often referred to as *Day 1 operations*. Automating the day-to-day tasks that are needed after the software is running, referred to as *Day 2 operations*, can provide benefits by reducing manual toil. Automating both the Day 1 and Day 2 aspects of application life cycle management can reduce the total cost of application ownership.

The Operator Framework for Kubernetes includes the [Operator Capability Model](#), a taxonomy for automation capabilities and maturity levels. Even if you aren't using operators, this taxonomy is helpful for planning which application life cycle activities to automate. The complexity of the automation and the value offered increases at each level. This taxonomy is used in this book to compare the capabilities of Helm, Operators, and Ansible. An overview of operator capability model is shown in table 1.

Table 1. Operator capability model

Operator capability level	Goal
Level 1 – Basic install (Day 1 operations)	Provide automated application provisioning and configuration management
Level 2 – Seamless upgrades (Day 2 operations)	Automate patching and upgrades between minor versions
Level 3 – Full life cycle (Day 2 operations)	Automate application life cycle events like backups and restores or failover and fallback to recover from failures
Level 4 – Detailed insights (Day 2 operations)	Automate tasks like collecting metrics, generating alerts, processing logs, and analyzing workload to support decision making and operational improvements
Level 5 – Autopilot	Provide automated: <ul style="list-style-type: none">▶ Horizontal and/or vertical scaling▶ Healing capabilities▶ Configuration tuning▶ Abnormality detection

Considerations for choosing an automation tool

There are several factors to consider when choosing an automation tool, including:

- ▶ Target environment
- ▶ Scope of the intended automation
- ▶ The skill sets needed to develop automations with that tool

An overview of the three automation tools is provided, what problems they are intended to solve, and which tool is best for specific use cases.



What is Helm?

Helm is a package manager for Kubernetes. It is designed to standardize and simplify packaging and deployment of container-based software for Kubernetes environments. Using Helm, developers can package their applications so they can be shared and easily deployed.

Helm is analogous to operating system package managers like Yum and APT or language package managers like npm and NuGet. Developers publish packages called Helm Chart to online repositories. A simple Helm install command can be used to download a Helm chart and deploy the software on Kubernetes.

Helm is an open source project that became part of the [Cloud Native Computing Foundation \(CNCF\)](#) in early 2016. Helm reached the maturity status of a [graduated CNCF project](#) in April of 2020.

What is Helm used for?

Helm is used primarily for automating Day 1 operations tasks, specifically installation and basic configuration management of applications on Kubernetes clusters. Some basic Day 2 operations like simple upgrades and rollbacks can be performed with Helm. In terms of the operator maturity model, Helm charts are considered to cover Level 1 Basic Install and Level 2 Upgrades.

Helm's greatest strength – is its simplicity.

It is easy to understand and use for software developers as well those installing and operating software. Manually deploying software onto a Kubernetes cluster can involve editing many lines of YAML to create and configure all the Kubernetes resources like deployments, services, secrets, and configuration maps. Helm is used to make the configuration process more generic, reusable, and repeatable.

Using Helm, configuration information is turned into a set of boilerplate templates and a set of parameterized values. The templates can then be used in all the environments where the Helm chart is deployed. The parameterized values are used to provide the specific configuration details needed for each application deployment. Helm merges the templates and parameterized values to produce the configuration that is applied to the Kubernetes cluster. Using Helm, the person installing the software only needs to specify the Helm chart to use and the values to customize the installation.

Due to its simplicity, Helm is not intended to provide full life cycle management or to manage applications with complex requirements. Consider other automation solutions for applications with complex state management requirements like databases, file, object, or block stores that need to be distributed across the cluster to provide availability and scalability.

Helm can be used:

▶ **For managing simple updates.**

Like upgrading which container image is used or changing deployment parameters. However, Helm does not have the capability to manage upgrades that involve sequences of tasks like performing database schema upgrades and then deploying new container images that depend on the updated schema only after the upgrade has been completed.

▶ **For simple rollbacks.**

To revert to the previous configuration that was running before an upgrade.

▶ **For deploying multiple instances or multiple versions software on a cluster.**

Helm charts can be used for deploying multiple copies or multiple versions with unique resource names for the resources created by Helm.

▶ **As a starting point for development and deployment automation.**

The simplicity of Helm for deploying and uninstalling software on Kubernetes makes it easier to use in development and deployment automation like continuous integration and continuous deployment (CI/CD) pipelines.



See [How Red Hat can help](#) later in this ebook to learn how Red Hat software partners can get their Helm Charts certified for Red Hat Openshift and listed in the [Red Hat Marketplace](#).



Helm technical details

This summary of Helm technical details is intended to help you understand Helm and evaluate it against other automation solutions.

- ▶ **Helm charts are written in YAML.**

Helm charts are packages of related Kubernetes YAML files used as templates for creating and/or updating the objects necessary in a Kubernetes cluster to run an application. Software development skills are not necessary to produce Helm charts.

- ▶ **Helm makes configuration files reusable by separating boilerplate configuration and parameterized values.**

The *Go template language* along with some custom Helm functions are used for templating configuration files. Go templating was inspired by Jinja2 templating, which is used in Ansible and a number of other software projects.

- ▶ **Configuration parameters are typically stored in values files like `Values . YAML`.**

Helm combines the configuration templates and values files to generate YAML, which is applied to the Kubernetes cluster to deploy or modify applications. Helm commands specify the chart and the values file to use. The same chart with different values files could be used to deploy different instances of the application for development, test, and production.

- ▶ **A *Helm Release* is a running instance of an application deployed by Helm.**

In Helm, release refers to the installed object instances created on the Kubernetes cluster by the Helm Chart. When a chart is installed, the Helm library creates a release to track that information. The release information is used in subsequent Helm commands like upgrade, rollback, and uninstall.

- ▶ **Helm Charts can specify other charts as dependencies that need to be run first.**

Dependencies can be used to install other pieces of software using either local charts or charts from online repositories.

- ▶ **Helm repositories are HTTP servers.**

Similar to Yum and APT, HTTP servers are used for hosting and distributing Helm Charts. Only file serving capabilities are necessary, so HTTP services that only serve static content, like GitHub pages, can be used as Helm repositories. Helm repositories can also be local.

- ▶ **Helm consists of a client tool and a library.**

Both are implemented using the Go language. While many Helm tutorials focus on the Helm tool as a command line interface (CLI), Helm functionality is also available in Kubernetes user interfaces such as the Red Hat® OpenShift® Web Console. The Helm CLI client can be installed on a developer's or operator's machine running Linux®, macOS, and Windows. The Helm client has a number of similarities to other CLI management tools like **kubect1** and **oc**:

- ▶ It communicates with a cluster using the Kubernetes API, so it can be run outside of the Kubernetes cluster.

- ▶ Before Helm can communicate with the Kubernetes cluster, you must be authenticated. Typically the authentication is shared with other CLI tools, like **kubect1** and **oc**.

- ▶ Helm only interacts with a single Kubernetes cluster at a time, it has no multicluster capabilities. To use a Helm chart on a different cluster, change the **KUBECONFIG** environment variable, to point to the desired cluster and then run Helm again.

- ▶ **Software does not have to be installed on the Kubernetes cluster to use Helm.**

The Helm client interacts directly with the Kubernetes API. Versions of Helm prior to version 3 required a component called tiller to be installed on the cluster. This is no longer necessary.

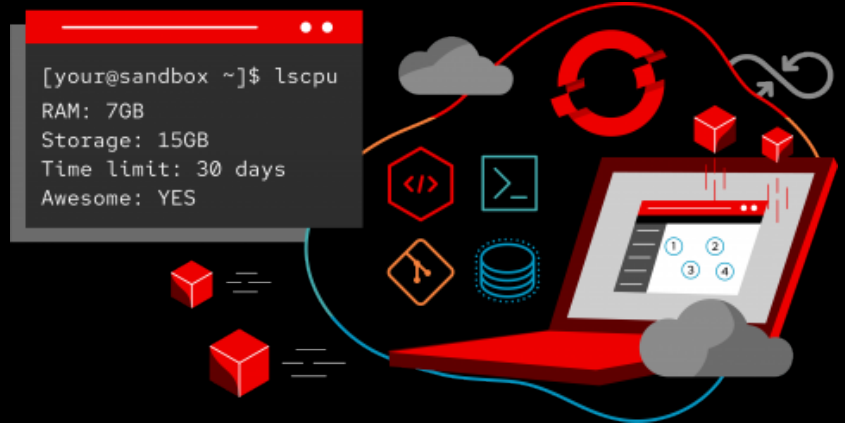
For more information on Helm

- ▶ The Helm project site, [Helm.sh](https://helm.sh)
- ▶ [What is Helm?](#)
- ▶ [Simplify application management in Kubernetes environments](#) (e-book)
- ▶ [Using Helm with Red Hat OpenShift](#)
- ▶ [Working with Helm](#) (Red Hat Developer)

Try out Kubernetes and Red Hat OpenShift

Learn and experiment with Kubernetes and Red Hat OpenShift in your browser for free now. Provision a Red Hat OpenShift development cluster and deploy a sample application in our no-cost sandbox.

[Explore](#) the OpenShift sandbox.



Kubernetes Operators

What are Kubernetes Operators?

[Kubernetes Operators](#), also known as the Kubernetes [operator pattern](#), are intended to automate the deployment and management of Kubernetes-native applications. Human operators who manage an application, like site reliability engineers (SREs), develop deep knowledge of how an application is deployed, how it is supposed to behave, and how to react if there are problems. The goal for operators is to package this operational knowledge in the form of software that can be shared with an application's consumers. While the goal for all automation is to reduce or eliminate repetitive tasks, operators are intended to make application-specific operational expertise more widely available, particularly for complex, dynamic applications that react to changes in load and can handle failures gracefully.

As can be seen from the operator maturity model described earlier in this ebook, operators are intended to be able to provide full life cycle automation for applications. An application that is managed with a sufficiently capable and mature operator should be able to provide a service level through automation that is similar to what is available from managed service providers.

Not every operator will implement all five levels of the operator maturity model. Initially, an operator implementation might provide only Level 1 or 2 capabilities covering basic installation and minor version upgrades. However, the operator pattern and tooling for producing operators, like the Kubernetes Operator Framework and Kubernetes Operator Software Development Kit (SDK), support the development of advanced capabilities through Level 5.

The concept of the Kubernetes operator pattern was developed at CoreOS in 2016. Red Hat acquired CoreOS in 2018. Work on operators continued in open source communities, leading to a number of active open source projects that provide tooling for operator development and management.

What are operators used for?

Operators can be used for:

- ▶ Deploying and managing an application, including any dependencies and resources needed to run the application.
- ▶ Making periodic backups or checkpoints of the application's state and restoring from a backup if necessary.
- ▶ Performing multi-step application upgrades that include tasks like database schema upgrades in conjunction with rolling out new container image deployments and other related configuration changes.
- ▶ Collecting metrics, generating alerts, and performing performance analysis independently or in conjunction with other tools installed on the cluster.
- ▶ Scaling an application's deployment to match current loads or anticipated changes in load based on schedule or season.
- ▶ Publishing service-discovery information to applications outside the cluster that cannot use Kubernetes APIs to locate the services.

Operators are ideal for automating deployment and management of complex applications and services like highly available distributed storage and messaging systems running on Kubernetes.

Red Hat has been driving development and adoption of operators in its products and in open source communities:

- ▶ Red Hat products like Red Hat AMQ streams, a high-performance data streaming system based on Apache Kafka, provide operators for installation and management on Red Hat OpenShift and Kubernetes clusters. The Red Hat Ansible Automation Platform Operator simplified the deployment of Red Hat Ansible Automation Platform on Red Hat OpenShift, giving organizations a streamlined path to implementing an enterprise-wide automation solution.
- ▶ Starting with version 4 of Red Hat OpenShift, Red Hat uses operators to install and manage the OpenShift cluster itself. The benefits is that the full life cycle management capabilities of operators are available for managing the OpenShift platform.
- ▶ Red Hat is leading the development of operators in the upstream open source communities for software defined storage and data management projects like [Rook-Ceph](#) and [NooBaa](#). These operators are available on the repository [OperatorHub.io](#). This public repository for sharing operators was launched by Red Hat in conjunction with Amazon, Microsoft, and Google Cloud.

The development of the Operator Framework, which includes the Operator SDK, the Operator Lifecycle Manager (OLM), and the OperatorHub software has been driven by Red Hat in conjunction with the open source community since 2018. The Operator Framework became an incubating project under the CNCF in July 2020.

Kubernetes operator pattern technical details

The key aspect of Kubernetes Operators that differentiate them from other automation approaches is that operators themselves are actually Kubernetes applications. An operator is a piece of software installed on Kubernetes that reacts to events and manages application-specific resources on the Kubernetes cluster. The operator pattern can be used to extend and enhance Kubernetes without modifying Kubernetes itself.

Operators are a special type of Kubernetes application that defines [custom resources \(CRDs\)](#) and controllers to manage them. CRDs are higher-level abstractions for applications and/or their components. A CRD defines high-level objects like **MyApp-WebServer**. The operator is responsible for managing the existing lower-level Kubernetes resources like deployments, services, secrets, and configuration maps when an instance of **MyApp-WebServer** is required to run on the Kubernetes cluster.

Kubernetes Operators extend the core Kubernetes concept of the control loop. The control loop continually observes the state of the Kubernetes cluster, compares it to the state that is desired as expressed by the sum of the configuration, and takes action to resolve those differences by calling on controllers that manage specific Kubernetes resources.

When an operator is installed it is defined as the controller for the operator's CRDs. As an extension of the control loop, operators continually check that the application's current state matches what is defined and desired. The controllers implemented by the operator will then be called to make any necessary changes to the current state.


Implementing a Kubernetes operator

Implementing an operator requires interaction with Kubernetes APIs at a detailed level. Initially, like Kubernetes itself, operators were implemented using the Go language. Fortunately, tooling like the operator SDK is available to simplify the task of implementing operators. The operator SDK provides high-level APIs and abstractions to make it easier to implement operators without in-depth knowledge of Kubernetes internals. The operator SDK provides tools that generate scaffolding and boilerplate code to make the process of starting a new project faster.


The operator SDK provides three approaches for implementing operators:

- ▶ Using Helm charts as the starting point for building Helm-based operators that provide Level 1 and 2 capabilities
- ▶ Using Ansible to implement operators that leverage Ansible's automation capabilities
- ▶ Using the Go programming language

Due to the nature of Helm, Helm-based operators are limited to Level 1 and 2 operator capabilities. The operator SDK allows for Go and Ansible-based operators to implement all five levels of the operator maturity model, as illustrated in Figure 2.



For in-depth information on building operators, using the Operator-SDK, and best practices, read the O'Reilly book, [Kubernetes Operators – Automating the container orchestration platform](#).



See [How Red Hat can help](#) later in this ebook to learn how Red Hat software partners can get their Operators certified for Red Hat Openshift and listed in the [Red Hat Marketplace](#).

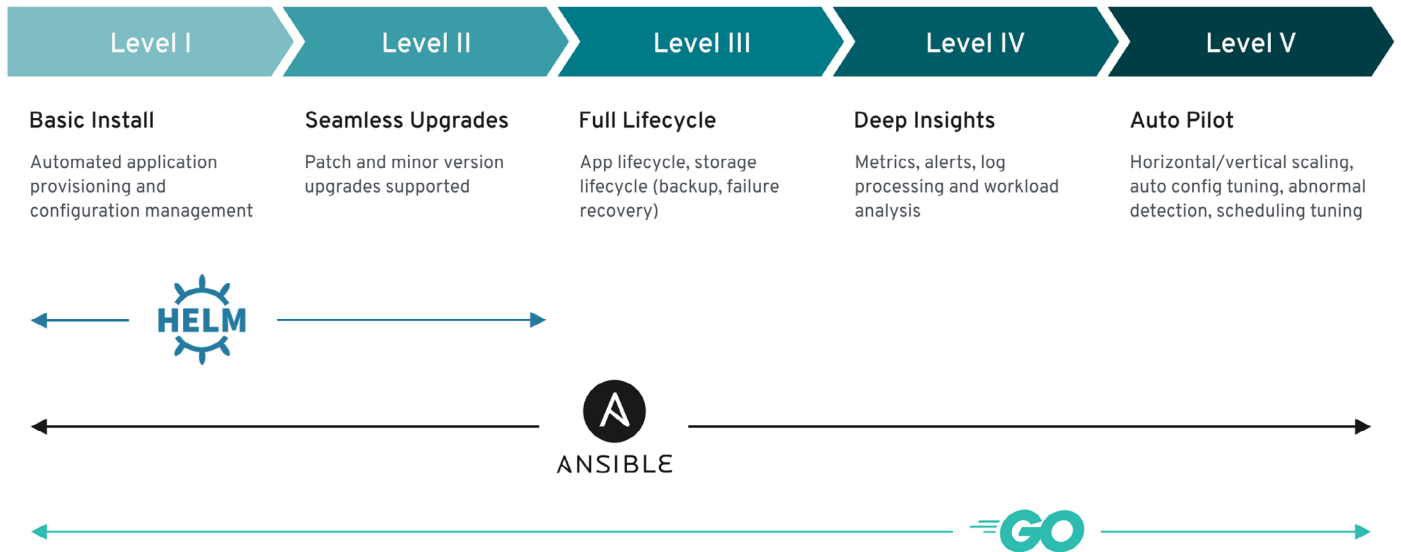
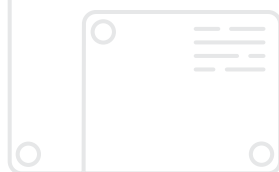


Figure 2. Operator capabilities levels by implementation approach

Even with the operator SDK tooling, the process of developing operators should be viewed as a software development project. The generated operator needs to be built into an image that can be deployed and become a running service on the Kubernetes cluster. Installing operators requires cluster administration privileges. Aspects like operator permissioning need to be addressed even with simpler approaches like Helm-based operators. A software engineering skill set is recommended.

For more information on Kubernetes Operators

- ▶ [What is a Kubernetes operator?](#)
- ▶ [What are Red Hat OpenShift Operators?](#)
- ▶ [Simplify application management in Kubernetes environments](#)
- ▶ [Kubernetes Operators: Automating the Container Orchestration Platform](#)
- ▶ [Operator framework](#) project, which includes [Operator SDK](#), [Operator Lifecycle Manager](#), and [OperatorHub](#)
- ▶ [Operator SDK with Helm, Memcached](#): A tutorial to create a Memcached operator from an existing Memcached Helm Chart
- ▶ [Operator SDK: Helm Operator Tutorial](#)
- ▶ [Operator SDK: Ansible Operator Tutorial](#)
- ▶ [5 tips for developing Kubernetes Operators with the new Operator SDK](#)
- ▶ [Kube by Example Learning path: Building operators with Helm, Go, and Ansible](#)



What is Ansible?

Ansible is an open source automation tool that can be used for configuring systems, deploying software, and managing complex processes like continuous deployments or zero downtime rolling updates. Ansible was initially released in 2012 and is the oldest and most mature of the automation tools described in this e-book. Red Hat acquired Ansible in 2015.

It can be used to automate tasks on systems running Linux, macOS, and Microsoft Windows, network infrastructure, cloud environments, and orchestrate across environments. The term Ansible has been used to refer to a number of things:

- ▶ A set of foundational open source command-line binary applications (also known and packaged as Ansible Core), as well as dozens of additional open source projects and integrated applications in the [Ansible GitHub organization](#).
- ▶ An aggregated [community package](#) that includes all command-line binary applications (Ansible Core), and many popular Ansible content collections that are built, packaged, and released on a [regular cadence](#).
- ▶ A way to describe automation – specifically, a human-readable data serialization language abstraction ([YAML](#)) that defines how automation tasks are written, built, and run.
- ▶ [Red Hat Ansible Automation Platform](#), a paid product that, when combined with a subscription, provides full enterprise life cycle support for organizations looking to standardize, operationalize, and scale automation. It includes numerous upstream components, Red Hat Ansible Certified Content Collections from 60+ partners to assist in installing, configuring and supporting automation in your organization.

While Ansible is a powerful automation tool, it was designed to be radically simple and very extensible. Ansible is flexible, and lightweight. Systems managed by Ansible do not need to have an agent or any Ansible software installed—usually they only need SSH connectivity or access to an appropriate API. To be able to automate a wide range of devices, Ansible has multiple connection plugins that can connect to many different types of managed nodes. This makes it easier to start using Ansible and improves the security posture by minimizing the attack surface.

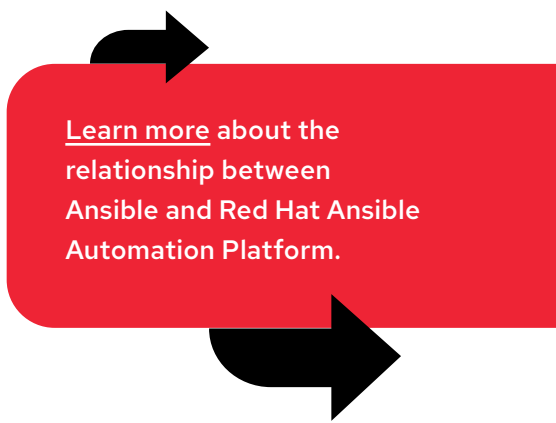
The **ansible-core** package includes the Ansible engine. Once that package is installed on a system, it provides the ability to configure and manage other systems and devices. It is primarily implemented in Python and can run on Linux systems and other Linux/Unix derived systems such as macOS. It is also available to run on Microsoft Windows using Linux or POSIX compatibility tools such as Windows Subsystem for Linux (WSL). Ansible core does not require any daemons, agents, or databases. It is lightweight enough that it can be packaged with and installed by the software you produce to manage deployments.

While Ansible core does not include a graphical user interface (GUI), Red Hat Ansible Automation Platform does. One of the main components of Red Hat Ansible Automation Platform is the automation controller (formerly Ansible Tower). Along with a user interface (UI), the automation controller includes role-based access control (RBAC), workflows, and continuous integration and continuous delivery (CI/CD) to help organizations standardize how automation is deployed, initiated, delegated, audited and scaled.

The focus in this e-book is on the Ansible core functionality that is available from the [aggregated community package \(Ansible\)](#). That package includes all the command-line binary applications (Ansible Core) and many popular Ansible content collections. The Ansible community package is not supported by Red Hat though help is available from the Ansible community. Support is available from Red Hat for those with a subscription for Red Hat Ansible Automation Platform.

What is Ansible used for?

As a very flexible, simple, and general automation tool, Ansible has been used for an incredibly wide range of tasks. There is a vast library of Ansible content available from Red Hat, software and hardware providers, and the Ansible open source community. The breadth of the certified and community supported Ansible content available covers installing and operating a wide range of software, systems, and devices including network routers, switches, SDN controllers, firewalls and load balancers. A selection of key [Ansible integrations](#) are shown here – click on a company logo to see their integrations.



[Learn more](#) about the relationship between Ansible and Red Hat Ansible Automation Platform.

While Ansible is used in enterprises to manage large fleets of systems, Ansible is also used by operators and developers to automate tasks that only run on a few systems, but need to be repeatable.


Ansible automations are written using primarily YAML. [Ansible Playbooks](#) offer a repeatable, reusable method to declare configurations, orchestrate steps of any manual ordered process and launch tasks synchronously or asynchronously. Ansible modules provide high-level functions for a vast range of operations that can be declared from a task and run from an Ansible playbook. For example, to install a specific package such as `ntpdate` on a set of systems, I could use the package module within an automation task as follows:

```
- name: Install ntpdate
  ansible.builtin.package:
    name: ntpdate
    state: present
```



Ansible is easy for operations staff to learn, a software engineering skill set is not required. A large community of people already have Ansible skills or have relevant experience from other similar configuration management and automation tools.

Ansible can be used to automate the full life cycle of applications. A few of the things developers might use Ansible for include:

- ▶ Software build and deploy automation including implementing the actions that need to be performed by CI/CD systems.
- ▶ Automating the process of building and testing container, virtual machine, and operating system images.
- ▶ Automating the installation, configuration, and management of application software dependencies, such as web, database, and cache servers.
- ▶ Orchestrating complex multi-step application roll outs or upgrades where each step needs to be completed successfully before the next step is performed.
- ▶ Orchestrating tasks across multiple environments, including hybrid deployments across on-premise systems, cloud, and edge environments.



For more information about Red Hat Ansible Automation Platform download the e-book, [Red Hat Ansible Automation Platform: A beginner's guide](#).



Develop an organizational strategy for automation

Read [An IT executive's guide to automation](#) to learn the benefits of a developing long-term automation strategy and the best practices to promote adoption within your organization.



Ansible technical details

This brief summary of Ansible technical details is intended to help you understand Ansible and evaluate it against other automation solutions.

- ▶ **Ansible is primarily implemented in Python.**

Ansible core can be installed in Python development environments by running `pip install ansible`.

- ▶ **Ansible has a number of mechanisms like roles and collections for organizing Ansible content, making it reusable, and shareable.**

Collections are a packaging format for bundling and distributing Ansible content, including plugins, roles, modules, and more. A select group of collections are included when Ansible is installed. Other collections can be found online and installed through [Ansible Galaxy](#). Ansible Galaxy is an online resource for finding and distributing Ansible community content. The command line tool for managing Ansible collections is `ansible-galaxy`.

- ▶ **Ansible is easily extensible through modules and plugins.**

- ▶ Ansible modules are the units of work that Ansible ships out to remote machines. Modules are often implemented in Python where there is a useful communal library of code to use. However, modules can be implemented in any language that is available in the target environment. The primary requirement is that modules return JavaScript Object Notation (JSON).

- ▶ Ansible plugins are used to extend the core functionality of the Ansible engine. Plugins run on the system running Ansible that is used to manage other targets.

- ▶ **[Idempotency and desired state](#) are key Ansible concepts that make Ansible well suited for infrastructure as code.**

Most Ansible modules check whether the desired final state has already been achieved, and exit without performing any actions if that state has been achieved, so repeating the task does not change the final state. Modules that behave this way are often called *idempotent*. Whether a playbook is run once, or multiple times, the outcome should be the same.

Idempotency allows Ansible playbooks to describe how things should be configured. If the configuration needs to be changed, the playbook that describes the desired configuration gets updated and can then be used to update the configuration on existing deployments or used to correctly configure new deployments.

- ▶ **Ansible supports both a declarative and imperative style for module implementation.**

The declarative approach is preferred for infrastructure as code and is the approach used by Kubernetes.



Certify your automation content

Red Hat partners can build and publish [certified Ansible Content Collections](#) through the Automation Hub on Red Hat Ansible Automation Platform. For consumers, the certification is a statement that the Ansible content is supported by Red Hat and Red Hat ecosystem partners.

Using Ansible with Kubernetes and Red Hat OpenShift

Given Ansible's flexibility, there are many ways it can be used for automation in and around Kubernetes environments. However, there are three broad categories to consider:

- ▶ **Using Ansible in the traditional way using either the command line or the Red Hat Ansible Automation Controller**

Ansible playbooks and other content can be run outside the Kubernetes cluster to automate tasks both inside and outside the cluster. This approach is very flexible and could be used for deploying and managing Kubernetes clusters or the infrastructure they run on, as well as, the applications and other objects running inside the cluster.

- ▶ **Using the Operator SDK to build a Kubernetes operator that includes logic from Ansible playbooks, roles, and collections**

The Operator SDK uses Ansible content along with Go language scaffolding and libraries to build and compile an operator executable. When building an Ansible-based operator with the Operator SDK, you define the CRDs your operator will manage. As with any other operator, the generated operator is installed on a Kubernetes cluster and becomes the controller for the CRDs managed by the operator.

The advantage of this approach is that it leverages Ansible skills and content to create an operator that runs inside the cluster and can react to changes in state. This approach does not require an installation of Ansible inside the cluster.

- ▶ **Running Red Hat Ansible Automation Platform on Red Hat OpenShift as an application managed by OpenShift**

The Red Hat Ansible Automation Platform operator, which can be found on OperatorHub.io, uses the power of operators to install and manage the Red Hat Ansible Automation Platform on OpenShift.

This makes it easier to install and operate all the tools included in the automation platform. Ansible playbooks, roles, and collections can then be run on the cluster to manage objects inside or outside the cluster.

All three of these categories have the benefit of letting you use the breadth and depth of Ansible content, which has been produced for over a decade, in your automations. One of the core Ansible collections is the [collection for managing Kubernetes](#). The collection provides access to Kubernetes APIs and objects from your Ansible automations.

For more information on Ansible

- ▶ Learning Ansible

- ▶ [Overview: How Ansible Works](#)

- ▶ [Learning Ansible basics](#)

- ▶ [What is an Ansible playbook?](#)

- ▶ [Ansible self-paced labs](#),
begin with [Writing your first playbook](#)

- ▶ [Red Hat Developer: Getting started with Ansible](#)

- ▶ [Download Ansible e-books](#)

- ▶ Download [Red Hat Ansible Automation Platform](#)
at no cost by joining the Red Hat Developer program.



Learn Ansible online with self-paced labs like:

- ▶ [Writing your first playbook](#)

- ▶ [Getting started with Red Hat Ansible
Automation Platform on Microsoft Azure](#)

- ▶ [Installing Red Hat Ansible Automation Platform
on Red Hat OpenShift using the Ansible Operator](#)



Selecting an automation solution by use case



This section covers common use cases and the automation solution that is recommended in that situation.

Use cases that lean towards Helm

My application runs on Kubernetes and I:

- ▶ Want to provide a simple method for installation and basic upgrades.
- ▶ Want to make it easy for people to install and evaluate my software.
- ▶ Lack the time or resources right now to develop more full-featured automation.

Helm charts provide a straightforward way to quickly get basic applications running on Kubernetes. Helm is relatively easy to learn and use for people who package applications and those who install them. While the functionality available in Helm is limited compared to other automation solutions, it is a good way to get started automating deployment of applications on Kubernetes.

Use cases that lean towards Kubernetes Operators

My application runs on Kubernetes and:

- ▶ Has multiple components, like microservices, or has a fairly involved installation and upgrade process.
- ▶ Has complex requirements for managing state and/or availability.
- ▶ I want automations that can respond to events to help manage scalability and availability.
- ▶ I want to make the operational expertise my team has available to all the consumers of my application through automation.

Operators are actually Kubernetes applications that monitor the state of the applications they manage and react to changes to provide full life cycle management. As an application, operators can encode operational expertise to manage complex multi-component applications and those that need to be distributed across the Kubernetes cluster for scalability and availability.



Use cases that lean towards Ansible

I have other needs like:

- ▶ My application or some of its component do not run on Kubernetes.
- ▶ My application is deployed across multiple, possibly hybrid environments and I want automations that can manage across those environments.
- ▶ As part of my application's deployment, I also need to manage infrastructure like systems, security, network, storage, or edge devices.
- ▶ I want a general-purpose automation solution that reduces the number of tools my team needs to learn and support.

Ansible can be used for automation across the IT landscape including systems running Linux, macOS, and windows, network, storage, security, and edge devices. Automations can be run from a central system that manages systems and devices across multiple environments without requiring any agents or daemons to be installed on the target systems or devices.

- ▶ As part of my application's installation or upgrade procedures, I need to automate processes that involve workflows.

Ansible automations can orchestrate multiple-step processes where each step must be completed successfully before subsequent steps can be allowed to run, even if those steps involve systems and services that span environments. For workflows that might need to be restarted after correcting problems, the design principle of idempotency in Ansible content is helpful to avoid rerunning steps that should not be executed more than once.

- ▶ I want to provide automations for my application that can integrate with an enterprise automation system.

Ansible is an automation solution that scales from managing tasks on a single system to an enterprise-wide automation platform. Developers can produce Ansible content that can run standalone or can be managed by an enterprise deployment of the Red Hat Ansible automation controller.



Download [Red Hat Ansible Automation Platform](#)
at no cost by joining the Red Hat Developer program.

Summary

This table provides a high-level comparison of Helm, Kubernetes Operators, and Ansible:

Table 2. Comparison of Helm, Operators, and Ansible

	Helm	Kubernetes Operators	Ansible
Target environment	Kubernetes	Kubernetes	Full IT landscape
Intended purpose	Basic application installation and upgrades	Full life cycle management for applications	Broad scope IT automation platform
Effort to implement automation	Low	High	Varies based on task
When does automation run	Once	Automatically in response to changes in state	When invoked from CLI or automation controller
Support for conditional logic	Limited to templating	Yes	Yes
Automate complex workflows	No	Yes, but single cluster only	Yes
System, operating system, and network automation	No	No	Yes
Automate multicluster deployments	No	No	Yes
Automate across deployment environments	No	No	Yes

Learn with Red Hat Developer

Go to developers.redhat.com/learn for articles, interactive learning labs, e-books, and more.

Chapter 6:

How Red Hat can help

Red Hat can help you build your products, introduce them to a wider audience of prospective customers, and grow the services your organization offers. Training, certification, and consulting services are available from Red Hat. Organizations that sell products and services can grow their business by joining one of the [Red Hat Partner Connect programs](#).

To improve your reach, Red Hat offers a number of ways to make it easier to find and install your software:

- ▶ Red Hat OpenShift includes a catalog of installable operators using an embedded version of the OperatorHub site. This makes it easy for Red Hat OpenShift users to discover and deploy Kubernetes-native services from Red Hat, Red Hat's ISV partners, and the open source community.

Prospective customers can have higher confidence choosing software with Red Hat OpenShift Certified Operators, knowing that they are validated and well-integrated offerings from Red Hat and Red Hat partners. [Red Hat OpenShift Operator Certification](#) is available through the partner programs that are part of [Red Hat Partner Connect](#).
- ▶ [Red Hat Marketplace](#) is an open cloud marketplace to discover, try, purchase, deploy, and manage certified software for Red Hat OpenShift clusters whether they are running in public or private clouds or on premise. Red Hat Partners that certify their products can promote their products in the Marketplace.
- ▶ [Red Hat Ecosystem Catalog](#) makes it easy to find certified enterprise-grade products that run on any of the Red Hat platforms including Red Hat Enterprise Linux, Red Hat OpenStack platform, and Red Hat OpenShift. In addition, the Red Hat Ecosystem catalog also lists certified hardware and service providers.
- ▶ [Red Hat Ansible Automation Certification](#) is a shared statement of support for certified Ansible collections between Red Hat and ecosystem partners. Partners can build, certify, and publish content to deliver Ansible Content Collections through Automation Hub on Ansible Automation Platform.

Training and accreditation for Red Hat partners

In 2022, Red Hat began offering Red Hat training – previously only available to customers – to partners at no cost. [Red Hat Partner Training](#) offers courses to strengthen your open hybrid cloud expertise by building technical skills that boosts developer and sales productivity and customer support. Courses are updated regularly and cover a variety of topics.



About Red Hat

Red Hat is the world's leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high-performing Linux, hybrid cloud, container, and Kubernetes technologies. Red Hat helps customers develop cloud-native applications, integrate existing and new IT applications, and automate and manage complex environments. A trusted adviser to the Fortune 500, Red Hat provides award-winning support, training, and consulting services that bring the benefits of open innovation to any industry. Red Hat is a connective hub in a global network of enterprises, partners, and communities, helping organizations grow, transform, and prepare for the digital future.

Copyright © 2022 Red Hat, Inc. Red Hat, the Red Hat logo, Ansible, and OpenShift are trademarks or registered trademarks of Red Hat, Inc., in the U.S. and other countries. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.