Compliments of redhat

Red Hat® Special Edition

Container Storage

FOR DUMMIES®

A Wiley Brand

Learn:

- Persistent storage for modern applications
- Storage for and in containers
- Dynamic storage provisioning for developers

Ed Tittel
Sayan Saha
Steve Watt
Michael Adam
Irshad Raihan

Container Storage

FOR DUMMIES®

A Wiley Brand

Red Hat® Special Edition

by Ed Tittel, Sayan Saha, Steve Watt, Michael Adam, and Irshad Raihan

FOR DUMMIES®

A Wiley Brand

Container Storage For Dummies®, Red Hat® Special Edition
Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-41663-0 (pbk); ISBN: 978-1-119-41656-2 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Carrie A. Burchfield
Acquisitions Editor: Amy Fandrei
Editorial Manager: Rev Mengle
Business Development Representative: Ashley Barth

Table of Contents

Introduction

Linux containers are an industry phenomenon that is top of mind in development shops around the globe. Containers have quickly moved from "science project" to production status in just a few years, for many good reasons. The meteoric uptick in interest in containers is driven primarily by developers and DevOps staff looking for a way to push more code out the door, with more stable and usable code from the get-go.

Developer focus and interest has led to new containers built to permit applications to scale rapidly, be more reliable, and offer better performance than more conventional means or methods. While developers like to think about what's possible, admins tend to think in terms of what's stable. They must be able to manage infrastructures quickly and easily and want to know how containers can meet those requirements.

Although many organizations still use traditional storage appliances, they don't offer the agility needed by containerized environments. Containers are highly flexible and bring incredible scale to how apps and storage are delivered; traditional storage can be the bottleneck that stops this progress. The underlying storage should be highly elastic, easily provisioned by developers and admins, and, ideally, managed using the same orchestration framework (like Kubernetes) used for application containers.

You discover throughout this book that a key part of achieving flexibility and scalability is container-native storage, a type of storage deployed inside containers along with the apps already running there.

About This Book

This book consists of six short chapters:

- Chapter 1 introduces Linux containers and why they're useful. It also covers the inherent lack of persistent storage in containers and the necessary workaround.
- Chapter 2 explains the various flavors of storage for and in containers.
- Chapter 3 takes a deeper look at container-native storage and explores how a common management plane makes container storage easier for admins and developers.
- Chapter 4 describes the reasons why developers have embraced container-native storage.
- Chapter 5 offers ten compelling reasons why you should move toward container storage ASAP.

Chapters are designed to stand alone, so if you want to dig into SDS, jump to Chapter 2. Otherwise, start with Chapter 1 to begin noodling through all things great and wonderful about Linux containers in general.

Icons Used in This Book

Every *For Dummies* book includes small graphics called *icons* sprinkled around its margins. These icons call attention to text worth noting for some reason or another:

Remember icons highlight points to recall as you immerse yourself in the ins and outs of container storage.

As techies, we like sharing some of the inner details about how things work and why. We do understand that you might not need to know such details, so we mark tricks of the trade with this icon to let you detour around them if you like.

This icon flags on-target information you can (and should) use to make the most of container storage.

Chapter 1

What's the Big Deal with Linux Containers?

In This Chapter

- Getting to know the Linux container
- Realizing why containers need persistent storage
- Making the Red Hat and Kubernetes connection

Linux containers offer a way for an infrastructure to behave like an application. These self-contained units wrap up pretty much everything they need in a small package, making app delivery more efficient. In this chapter, you discover what containers are and why they need persistent storage.

What Is a Container, Exactly?

A *container* is a small, lightweight bundle of one or more applications and the dependencies needed for that code to run. That means a container has code, a runtime environment, system tools, and libraries — the same stuff that you traditionally install on a server. A container can also house infrastructure services such as storage, or a hybrid of apps and storage. Packaging everything together makes a container highly portable and results in fewer integration errors.

Containers are often compared to virtual machines (VMs), but a container's footprint and scope are much smaller. For example, a VM is like a heavy hammer. It assumes you're running a server and that the server can host multiple applications. A container needs a minimal amount of operating system and

system resources to run its contents, which could be one application or several. And the container can run just about anywhere, even on a bare metal machine or in a VM — the container doesn't care.

Implementing containers is quick and easy. Where it might take minutes or (gulp) days to spin up a fully functional VM, the time to get a container moving is typically seconds.

Increased agility, scalability, and aggregation of services helps IT be more efficient and cost-effective, resulting in lower total cost of ownership (TCO).

Everything about containers sounds pretty great, but there's at least one sticky issue to consider — persistent storage for stateful applications deployed in containers.

The container ecosystem

Although containers are independent entities, they work with a lot of other technologies. You see a lot of terms covered throughout the book, but here, you look at the big picture, from the Red Hat perspective:

- **Container platform:** The goal of a container platform is to automate the packing and loading of containers, for greater efficiency, in addition to providing governance for the overall app development process. Red Hat OpenShift Container Platform enables developers to build, host, deploy, manage, and scale multi-container applications.
- **Orchestrator:** An orchestrator, such as Kubernetes, is a piece of software that manages application containers across a cluster, ensuring that each container keeps running regardless of conditions. (*Kubernetes* is Greek for the "helmsman" of a ship, as in a ship holding many containers.) But Kubernetes can manage more than just apps. It can help orchestrate storage in containers, too. Kubernetes is included in Red Hat OpenShift Container Platform.
- **Storage cluster:** This is a network of storage nodes that provides high availability and scale to applications that need to persist data. The storage cluster providing storage services could reside outside or within the container.

Why Containers Need Persistent Storage

Few applications are useful without some way to store data. Back in the day, Linux containers had no features for persisting data, and container engines and orchestrators couldn't support or manage storage, either. But all that's changed in today's brave new world.

Containers are ephemeral

A container, by nature, is a transient object. It might live on one server for a period of time and then head over to another if the orchestrator tells it to. While a container keeps its bundle of software and dependencies wherever it goes, it doesn't store data so it can maintain a light footprint.

Virtual machines (VMs) don't have this problem. With VMs, you start with an image, modify it, and save the altered state as a new VM. It's the same thing with containers, but the container isn't designed to persist any data that an application generates. If a process stops or the container is rebooted, all the data associated with any applications within is lost. Heck, hypervisors have always allowed for persistent storage in one form or the other. What's up with containers?

Local storage isn't enough

Like with VMs, some applications may need to persist their state, data, and configuration. For example, a database container needs persistent storage for its data store (where the actual database lives). In addition, given the ephemeral nature of containers, applications may need to persist their state beyond the life of the container. Local storage isn't sufficient because if the container moves to another host, it loses access to the data.

It's common in an on-premise data center to find servers and local disks, with a lot of storage. (A good deal of that storage is wasted, but that's beside the point right now.) Then consider a public cloud scenario. You have some local storage

capacity, but it's limited and often ephemeral. What if you have no access to network storage or the service is down for no fault of yours? Until you solve this issue, you can't host stateful applications effectively.

Stateful applications also require an underlying storage layer to provide enterprise features just like those available to apps deployed in virtualized (and other runtime) environments.

Storage adapters to the rescue

When it comes to developers building containerized applications, they have two primary concerns. First, they need to provision the storage their application will consume, and second, they need to ensure that their containerized application can mount and use the storage they provisioned in order to get the persistence they need.

In order to address the provisioning concern, OpenShift's storage framework allows provisioners to deliver volumes from a wide array of on-premise and cloud storage platforms. Similarly, OpenShift provides volume plugins that ensure that when a container is scheduled on a node, it can mount the storage volume, start the container, and bind the volume mount to a directory within the container.

Persistent volumes are storage connections that point to storage resources.

No need to worry about the security of your data, either. If you're using the right storage platform, you can restrict access to it using SELinux, which protects the data from unauthorized changes or corruption.

Don't lose track of metadata

Metadata is as important as containers themselves. It describes the contents within each container, without which management across a cluster becomes a nightmare.

Metadata is essentially arbitrary key-value pairs in a container image. There's all kinds of metadata, such as name, release, vendor, architecture, and so on.

Developers can store metadata inside containers, along with other contents, but what if the container is damaged and no longer functions?

Persistent storage maps to information about containers in a cluster, which is needed to ensure secure and distributed data storage for applications in containers, and proper delivery of each package. Bundling such critical information as metadata within a container (for example, local disk storage) isn't a great solution. Metadata is too important to the whole process to risk it disappearing due to some failure or disaster.

The best approach is to distribute metadata across multiple containers. The information could be stored in a container of its own — something called container-native storage — or in container-ready storage. Linux containers offer flexibility and agility, plus packaging and distribution for applications, data, and metadata.

The Red Hat and Kubernetes Connection

Red Hat has been a solid contributor to the Kubernetes community, with the most contributors and contributions after Google. (A bunch of Star Trek-loving Google engineers designed Kubernetes and released the first version in 2015.) Red Hat also has the second largest contributor base in the Docker community after Docker itself.

Red Hat standardized on Kubernetes for its container platforms and has contributed open-source volume plugins and provisioners, such as iSCSI, FibreChannel, Amazon Elastic Block Store, Azure File Storage, Google Compute Engine (GCE) Persistent Disks, and NFS along with a number of Red Hat technologies, to Kubernetes.

During the technology's fledgling days, Red Hat quickly doubled down on its efforts to build storage capabilities for Kubernetes, resulting in breakthrough innovations that not

only included multiple adapters but also critical features such as the following:

- Container scheduling
- Persistent volume framework and claims
- The persistent volume selector
- Dynamic provisioning
- Storage classes
- Volume security

All of these features are now rolled into the open-source Kubernetes.

Red Hat is all in on containers, with skin in the game. It has invested significant engineering resources behind containers and container storage initiatives. Enabling container storage goes a long way toward removing barriers to container adoption, and we're committed to keeping the momentum moving.

To build a truly enterprise container environment, you need elastic, durable, secure and enterprise class storage that can be integrated into the container stack. Learn more about container-ready and container-native storage in Chapter 2.

Chapter 2

Looking at Storage for and in Containers

In This Chapter

- Understanding software-defined storage as an integral prerequisite
- Considering types of container storage

Just as there isn't one kind of beer (which would be a tragedy, for some), container storage comes in a few different forms. Storage deployed in containers gives developers and admins almost unlimited flexibility.

What's Wrong with Traditional Storage?

Storage appliances weren't designed for the agility, speed, and scalability that enterprises demand today. You could simply plug a traditional storage appliance into a container platform; however, an antiquated storage platform can hold you back from reaping the benefits of containerizing your applications.

Software-defined storage (SDS) separates storage hardware from storage controller software, enabling seamless portability across multiple forms of storage hardware. You can't slice and dice storage using appliances or typical SAN/NAS as easily, readily, or quickly as you can with SDS.

There's More Than One Type of Container Storage

Broadly speaking, container storage comes into play in two ways:

- **Storage *for* containers:** Also known as *container-ready storage,* this is essentially a setup where storage is exposed to a container or a group of containers from an external mount point over the network. Most storage solutions, including SDS, storage area networks (SANs), or network-attached storage (NAS) can be set up this way using standard interfaces. However, this may not offer any additional value to a container environment from a storage perspective. For example, few traditional storage platforms have external application programming interfaces (APIs), which can be leveraged by Kubernetes for Dynamic Provisioning.
- **Storage *in* containers:** Storage deployed inside containers, alongside applications running in containers, is an important innovation that benefits both developers and administrators. By containerizing storage services and managing them under a single management plane such as Kubernetes, administrators have fewer housekeeping tasks to deal with, allowing them to focus on more value-added tasks. In addition, they can run their applications and their storage platform on the same set of infrastructure, which reduces infrastructure expenditure. Developers benefit by being able to provision application storage that's both highly elastic and developer-friendly.

Naturally, given the inflexible nature of traditional storage appliances, they're unable to co-locate storage services inside containers as can be done with SDS. Organizations can use SDS, such as Red Hat Gluster Storage, to guard their existing storage investments. If you use a SAN, for example, you can use Red Hat Gluster Storage as a bridge to interact with Red Hat OpenShift Container Platform.

Red Hat takes storage in containers to a new level by integrating Red Hat Gluster Storage into Red Hat OpenShift Container Platform — a solution known as container-native storage.

Read on to Chapter 3 to find out how container-native storage delivers added value to developers and administrators.

Chapter 3

What's So Cool about Container-Native Storage?

In This Chapter

- Container-native storage is here, and it's a game-changer
- Exploring the common management plane for container-native storage
- Understanding the secret sauce behind container-native storage

Containers have become a matter of when, not if. But what are the benefits of container-native storage (CNS) that offset the pain and effort of climbing the learning curve, transitioning from current methods and approaches, and incurring the costs of switching over? This chapter should clear those things up and, hopefully, also buoy your interest in CNS.

Container-Native Storage Is the Next Sliced Bread

Storage, in general, is moving toward software-defined storage (SDS), which is expected to overtake conventional storage deployed on x86 boxes in the near future. SDS is a prerequisite of sorts to container-native storage.

Gartner predicts that by 2019, 70 percent of existing storage array solutions will be available as a software-only version. The research firm also predicts that by 2020, "70 to 80 percent of unstructured data will be stored in less-expensive storage hardware managed by SDS systems."

That means organizations should look to SDS, and container-native storage, as their future go-to storage options.

Common Management Plane Makes Life Easier

The Kubernetes orchestrator, which also goes by "k8s," is the container cluster manager within the OpenShift Container Platform, used for container-native storage. This common management plane greatly eases many potential burdens associated with container storage, much as the control plane and data plane make software-defined networking easier (for those familiar with this emerging paradigm).

Even though container-native storage is a fairly new technology, you don't have to live with some stripped-down version of the storage you're used to. You get the same types of features as in traditional storage, such as high availability (HA), data integrity, security, geo-replication, and snapshotting — they're either automatically included in the Kubernetes management plane or can be leveraged independently as is offered by the underlying storage platform.

Dynamic volume provisioning is unique to Kubernetes and a feature that really resonates with developers and administrators. It allows anyone with access to the management console to create storage volumes on demand.

Issues with static provisioning

Before dynamic provisioning came to light, allocating and requesting storage space had two major drawbacks: time and wasted storage space. The task of storage provisioning often involved an administrator getting on the phone with a storage provider to request more volumes. The same principle applied to developers, who had to guesstimate the amount of storage they might require and request it from the administrator.

The whole notion of guessing at storage capacity resulted in wasted space. Say an administrator had 1 terabyte (TB) of

space, which had ten 10 gigabyte (GB) volumes. If an application needed 50 GB, that left 50 GB unused in a 100 GB volume. If an application needed 150 GB, the administrator couldn't fulfill the request with the current configuration — not the kind of problem anybody wants!

The efficiency of dynamic provisioning

In fact, dynamic provisioning solves a lot of problems:

- Developers can provision storage on their own. They use a single console pane to select the amount of storage they want, the kind of storage, and access modes (such as read-write). Queuing up persistent volumes (PVs) is just that simple.
- Dynamic provisioning lets a developer or admin choose whatever amount of space is needed at the time, regardless of how a drive's space is configured, and the system handles allocation behind the scenes. Figure 3-1 shows how the platform automatically creates the volume and attaches it.
- Developers don't have to submit a storage request to an administrator, wait a week, and perhaps not even get the storage they want. Asking is getting, in this case.
- Admins can intervene to assist with dynamic storage, if needed, but they don't have to insert themselves in the middle of development projects just to provision storage.

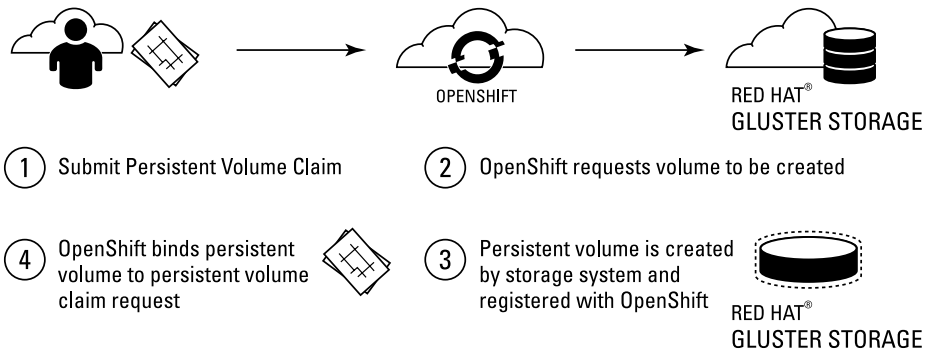

Figure 3-1: Dynamic provisioning of storage volumes is handled on demand, using the storage subsystem's APIs.

If you're not yet convinced of the dreaminess of CNS, consider these points:

- It's already configured. You need only ensure that the configuration is right for your environment.
- It's attached to an application and stays with the app throughout its development and deployment life cycles.

The Secret Sauce behind Container-Native Storage

It's not just Kubernetes and Red Hat Gluster Storage that are important for dynamic storage volume management — Heketi and gluster-kubernetes are both open-source projects initiated by contributors from Red Hat that help enable this solution.

The Heketi project provides both a RESTful API and a CLI for dynamically provisioning volumes out of GlusterFS. Heketi supports any number of Red Hat Gluster Storage clusters, which provide great flexibility in selecting the GlusterFS deployments from which you want to provision.

The gluster-kubernetes project enables administrators to manage the deployment and configuration of GlusterFS in Kubernetes and automatically manages hardware on behalf of administrators.

If you're ready to keep reading, head to Chapter 4 for reasons why developers are flocking to CNS.

Chapter 4

Container-Native Storage Is Blowing Up the Dev World

In This Chapter

- Examining the benefits of container-native storage to developers
- Understanding how container-native storage lets developers focus on code

Like the kid who says "It's not that I don't like tomatoes; I just don't like to eat them," developers who say they don't care about storage should think again. Container-native storage truly changes the face of storage for developers, offering them a lot more control and letting them spend more time on their first priority: code.

Developers Have More Control

Developers are certainly aware of the need for storage, but dealing with it can be a real pain. They simply want storage to be agile, reliable, and persistent. The mindset of developers regarding storage, or infrastructure in general, is that "it's best when it's out of my way."

Container-native storage puts developers in control. It lets them deal with storage directly from an easy-to-use interface, independent of an administrator. Developers can manage storage with a few clicks, within a few screens, without the know-how and technical details expected of a storage expert.

While traditional storage solutions have always been viewed as too rigid by developers, container-native storage allows them to get the storage they need immediately, with the ability to size up or down through dynamic provisioning. Really, what's not to like about that?

Developers Focus On What They Do Best: Code

Code, test, fix, retest, rip, retest, repeat. And do it as quickly as possible to meet the schedule. With heightened expectations at a breakneck pace, there's little wiggle room for developers these days.

From a development perspective, one of the coolest things about containers is that the environment in which code is developed closely resembles the environment in which the code is tested and deployed, whether that's a physical, virtual, or cloud environment. Sounds like a great time-saver.

Container-native storage, in particular, takes the guesswork out of storage capacity requirements, letting developers resume the more important task at hand — generating usable, stable code. It all adds up to a richer development experience.

Chapter 5

Ten Reasons to Get on the Container Storage Bandwagon

In This Chapter

- Preparing for the container storage transition
- Welcoming persistent storage to containers
- Understanding why developers and admins are turning to container storage
- Recognizing Red Hat as the de facto container platform from storage to application development

Each *For Dummies* book, including this one, ends with a Part of Tens. This one gives you a list of reasons to join the container storage revolution.

The Future Is Here; Go With It

Industry analysts see a major shift to software-defined storage (SDS), with SDS overtaking conventional storage by 2020. Container-based or container-native storage (CNS) is a big part of that move. Isn't your organization better off accepting and adopting the "storage of the future" rather than continuing to invest in traditional SAN and NAS technologies? The benefits of transitioning into the modern (container) era will pay big dividends down the road.

The Persistence Payoff for Container Storage

Stateful applications need to access data, but containers don't keep it for very long, by design. That's a problem. Ephemeral (for example, local) storage is usually too limited to be useful. With containers perceived as the next step in the evolution of server virtualization, it's critical to provide persistent storage options to administrators. Enter storage provisioners and volume plugins that create and mount storage requested by a container.

Orchestrators Simplify Container Management

Orchestrators keep tabs on all containers within a cluster, handling run schedules and environment monitoring. A container is guaranteed to be available when needed, even if the server on which it's running goes down, when the orchestrator is running the show.

Kubernetes is the most popular orchestrator around, capable of managing apps *and* storage in containers. Red Hat made Kubernetes the standard framework in Red Hat OpenShift Container Platform for good reason. Administrators can manage containerized storage services under a single management plane, and developers get self-service dynamic provisioning. Now that's putting the *uber* in Kubernetes.

More Automation with Dynamic Provisioning

Dynamic provisioning lets developers use SDS container-native storage, via Kubernetes, to obtain storage with known characteristics, without having to wait for the storage guys to get involved to configure allocations and set things up on their behalf. Go developers!

Scalability

Container storage scalability lets you run many more applications on the same box compared to using VMs, by an order of ten. If you run hundreds of VMs in a cluster, you can up that to thousands of containers. That's a lot more oomph from existing hardware. We like oomph, don't you?

Developers Love It

Container storage lets developers shave time from development and testing cycles, and gives them control over how storage is delivered. Developers can provision their own storage, autonomously, through a simple interface. Docking and loading containers adds another layer of efficiency to development, testing, and deployment scenarios. The bottom line is, developers can focus more on code and less on housekeeping.

Admins Love It

Container-based solutions reduce administrative burdens while giving admins more control over their environments. Because developers can request and provision storage themselves, admins no longer need to plant themselves in the middle of development projects. And orchestrators go to town keeping containers in check and running. Although administrators and developers seem to be on two different planets much of the time, CNS unites!

Lower TCO

What manager isn't concerned about costs? We understand that organizations want to maximize their current storage investments and expect to spend even more money to adopt container storage technologies. But given the increased agility, scalability, and aggregation of services that container storage brings to the table, as well as more efficient IT practices, you can realize a lower total cost of ownership (TCO) over time. The sooner you start, the more you save!

Red Hat containerized the storage platform itself to enable the vision of storage as a microservice, delivered via a container running alongside application containers, all managed by a single framework. And it lowers the cost of storage.

It's Open Source

With Red Hat's open-source orientation, everyone knows its road map. Customers can look at upstream, open-source developments and know what to expect, and developers see the code. Proprietary efforts are neither transparent nor predictable.

Red Hat Knows Its Stuff

Adoption of containers is tough in certain scenarios, but Red Hat can help because it's been at this for a long time and is one of the top contributors to open-source projects for Kubernetes and Docker. As a trusted advisor to industry, research, academia, and government, Red Hat offers a single point of contact for solutions across the container stack to help you succeed.

Container storage technology is changing rapidly. Red Hat isn't done with containers or container-native storage, either, so be assured that there's plenty more features and technology enhancements in the works. Keep an eye on `redhat.com/containerstorage` for announcements as things roll out. Also, don't forget to bookmark the blog with product announcement, industry news, and thought leadership articles: Visit `redhatstorage.redhat.com`.

Discover storage for and in containers

Container Storage For Dummies, Red Hat Special Edition, helps you understand why containers are all the rage, and how persistent storage for containers is an important challenge as enterprises transition from science project to production. Red Hat Gluster Storage is an enterprise grade, highly scalable software-defined storage solution that can be integrated with Red Hat OpenShift Container Platform to help developers manage storage for modern applications.

- ***Discover Linux containers* — *understand their importance***
- ***Understand container-native storage* — *see why it's blowing up the dev world***
- ***Get on the container storage bandwagon* — *find out reasons why***

Open the book and find:

- **Persistent storage for stateful apps**
- **Storage for container registries**
- **Single point of support**
- **Seamless scale and elasticity**
- **Unified management plane**
- **Dynamic provisioning**
- **Advanced storage services**

Go to Dummies.com®
for videos, step-by-step examples, how-to articles, or to shop!

FOR DUMMIES®
A Wiley Brand

ISBN: 978-1-119-41663-0
Not for resale

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.