LEARNING MADE EASY

HPE and Docker
Special Edition

Containers

for dummies®
A Wiley Brand

Learn what containers are

Uncover why enterprise teams are adopting containers

Embark on your journey to application containerization

Brought to you by:

Hewlett Packard Enterprise | docker

Scott D. Lowe

About HPE

Hewlett Packard Enterprise (HPE) is an industry-leading technology company that enables customers to go further, faster. With the industry's most comprehensive portfolio, spanning the cloud to the data center to workplace applications, our technology and services help customers around the world make IT more efficient, more productive and more secure.

About Docker

Docker is the world's most popular enterprise container platform. Built for both developer and IT operations teams, the end to end platform gives enterprises the portability, security and efficiency they require as they look to modernize their existing applications, embrace hybrid IT and go DevOps.

Containers

HPE and Docker Special Edition

by Scott D. Lowe

for dummies®
A Wiley Brand

Containers For Dummies®, HPE and Docker Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-119-41611-1 (pbk); ISBN 978-1-119-41609-8 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

Publisher's Acknowledgments

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

Development Editor: Elizabeth Kuball
Copy Editor: Elizabeth Kuball
Acquisitions Editor: Katie Mohr
Editorial Manager: Rev Mengle
Business Development Representative: Karen Hattan
Production Editor: Antony Sami

Introduction

A funny thing happened on the way to the data center. A technology with roots all the way back to the 1970s emerged as a "new" way to handle application development, one that could help address some of the resource inefficiency problems inherent with virtualization and with legacy IT.

Called *containers,* this currently reemerging technology construct is gaining traction and attention across the enterprise IT marketplace and is the subject of this book.

About This Book

This 48-page journey helps you understand how Docker containers and HPE can help your organization transform its IT operations and save time and money in the process. You discover how containers impact IT and the business and how easy — and how challenging — containers can be.

Foolish Assumptions

For this book, I assume that you have at least a basic understanding of data center computing and virtualization. The general audience for this book is anyone in IT who may want to learn more about data center architectures. You may be part of your organization's technical staff, or you may be managerial or executive staff. Either way, this book is for you!

Icons Used in This Book

Throughout this book, you find a number of icons intended to help you better understand and remember key concepts.

REMEMBER

When you see the Remember icon, put that information in your back pocket to save for later.

TIP

When I share something that may save you time or money, I mark it with the Tip icon.

TECHNICAL STUFF

This book doesn't go super deep into technical stuff, but sometimes I wade into the weeds, and when I do, I use the Technical Stuff icon. You can safely skip anything marked with this icon without losing the main point.

Beyond the Book

I can only cover so much in 48 pages. To learn even more about HPE and Docker's container partnership and how both companies can help you in your efforts, visit `www.hpe.com/partners/docker`.

IN THIS CHAPTER

- Seeing how the current container market came into existence
- Comparing and contrasting containers with virtualization and bare-metal workload deployments
- Identifying how containers support a variety of legacy and modern application needs

Chapter 1
What Are Containers?

Containers aren't new. In fact, today's container services can be traced way back to the early days of Unix's chroot operation. Until recently, the main problems with containers have been portability and management, problems that were addressed by the introduction of Docker to the market.

In this chapter, I explain how the modern container market got its start and how current market solutions have made containers a viable and increasingly popular option in today's enterprise IT market.

The Evolution of Containers

Containers have a long and storied history, but each iteration has in common the idea that there needs to be some level of abstraction in order for there to be efficiency and security in workload management. However, unlike virtualization, which provides abstraction at the hardware level, containers operate at the operating system (OS) level. As a result, some really interesting things happen, which I fill you in on later. For now, let's jump in the time machine and trek back to the groovy disco days of the 1970s and speed through to the techno era of the 2000s and beyond.

Back in 1979, before the heyday of the 1980s took hold, an aspiring abstraction rock star made the fateful decision to add a service called *chroot* to version 7 of Unix. In essence, chroot provided an isolated operating environment in which applications and services could run. Processes that operated inside these chroot subsystems were not able to access files outside their confines, making these isolated containers very useful for, for example, running test processes on production systems.

Fast-forward about 20 years. Around the same time that people were panicking about the potential collapse of society, the year 2000 also brought to the Unix-like OS world FreeBSD Jails. Jails took chroot to the next level by providing isolation for users, files, networking, and much more. In fact, individual jails were able to have their own IP addresses, so they were logically isolated at the networking level as well.

In 2004, Sun brought to the Solaris OS a service named, strangely enough, Solaris Containers. Like the similar services that came before it, Solaris Containers provided isolation into individual segments called *zones.*

In 2008, Linux Containers (LXC) came to the world of the popular open-source OS. This service formed the real foundation for the original Docker, which came on the scene in 2013.


REMEMBER

Docker is the company that brought containers to the mainstream. However, Docker itself has undergone a bit of a transformation since its inception as well. Originally released around LXC, Docker eventually replaced LXC with its own library called *libcontainer.* However, where Docker has really enjoyed success is in building an ecosystem around containers, something that had not happened with prior technologies, at least not to the same extent.

Among the services that Docker has brought to containers are a powerful application programming interface (API), a command line interface (CLI), an efficient image model, and cluster management tools that help container-based services to scale as well as they do. I explain much more about Docker's architecture later in this book.


REMEMBER

Lest you think that containers are a Linux-only affair, never fear! The world of Windows now enjoys containers, too. With the release of Windows Server 2016 and Windows 10, Microsoft and Docker have forged a partnership to bring Docker container support to Windows applications.

Virtual Machines vs. Containers vs. Bare Metal

As you learn about containers, you'll undoubtedly run across lots of blog posts and other authoritative articles that help you try to understand whether you should try to choose between containers or traditional virtualization for your data center. The fact is that this question requires far more context. In fact, it's almost like saying, "Should we take the car or should we go on vacation?" The comparison isn't really valid.

REMEMBER

The real question you should ask is: *Should I deploy workloads using a traditional method or a containerized one?* The focus here should be on the workload, not on the underlying infrastructure. And if you choose containers for your deployment efforts, your next question should be: *Should I deploy this on bare metal, into a virtualized environment, or in the cloud?* Adopting containers doesn't mean eliminating virtualization. You might even choose to go to a full-on Inception-like infrastructure and run containers inside other containers!

A quick virtualization architecture review

Before continuing, I want to give you a quick review of bare-metal virtualization environments versus environments based on hypervisors such as VMware vSphere or Microsoft Hyper-V. Figure 1-1 compares a bare-metal application deployment to a virtualized one.

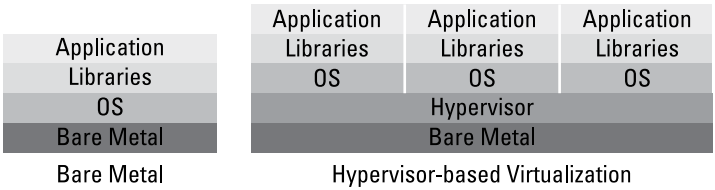

FIGURE 1-1: Virtualization versus bare-metal servers.

You can see why virtualization has become so popular: Deploying new virtual machines is dead simple, and you can share underlying hardware resources to do so. Prior to virtualization,

most servers typically ran at somewhere between 5 percent and 15 percent utilization. But with virtualization, it's possible to get closer to 100 percent, which means that you're using those available resources. In short, you get much higher levels of workload density because you're running many more workloads on far fewer servers.

In addition to the resource waste inherent in virtual machines, there are workload portability issues, too. To migrate an application from one location to another, you have to drag an entire OS along with it. Although virtualization is truly incredible in what it can do for workload density, this waste is significant. For example, each individual virtual machine requires a separate OS installation. Each OS requires access to RAM, storage, and processing resources. As you scale your environment and deploy more and more virtual machines, you end up dedicating a whole lot of resources to nothing more than operating systems. Moreover, if you peel back the covers of many virtual machines, you'll find that most don't generally use all — or even close to all — of the resources that they're assigned. There's a lot of waste.

This same issue, as well as proprietary virtual machine formats, makes it difficult to move workloads between vendors. You can't just drag a vSphere-based virtual machine over to Amazon EC2 or to Hyper-V, for instance. Workload portability beyond the hypervisor confines is not well supported.

There has to be a better way

What if you could effectively eliminate this redundancy? With containers, you can, but it doesn't necessarily mean moving away from virtualization.

I cover container system architecture later in this book, but for now, I'll focus on what's called the *engine* — the software that enables containers to work their magic.

The container engine is effectively an abstraction layer that enables libraries and application bundles to operate independently without the need for a complete OS for each bundle. The container engine handles all the hooks to the underlying single-instance OS. Figure 1-2 gives you a look at a container construct.

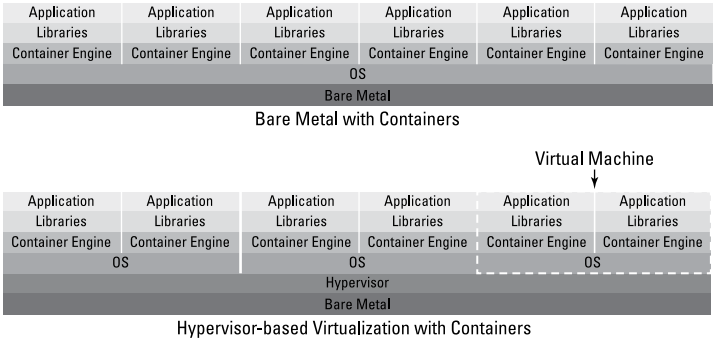

FIGURE 1-2: Improved density with the addition of containers.

TIP

Compare Figure 1-2 to Figure 1-1, and you'll notice some key differences. First, the container engine abstraction layer is attached. Second, even on bare metal, with the container engine, you can run multiple *isolated* workloads on the same hardware. Previously, this ability was the purview of a virtualization solution, which, again, also required a compete OS per instance. Now, you can do this without a full hypervisor in place and without all the overhead of a bunch of operating systems.

And even when you do run a hypervisor, you still get a lot of benefits. Virtualization has been popular for a lot more than just workload density and saving hardware money. By turning servers into software, you've gained the ability to manipulate those workloads in powerful ways. This has led to a rise in application availability capabilities, new disaster recovery opportunities, and new ways to programmatically manage workloads.

To maintain these benefits, you can run container engines on top of your existing virtualization stack, which is shown in the bottom part of Figure 1-2. As is the case with bare metal, you can deploy multiple container engines — and thus, multiple isolated workloads — on top of a single virtual machine. This way, you get those sweet virtualization benefits while, at the same time, even further increasing workload density thanks to containers.

REMEMBER

The key takeaway from this section is this: You can run containers wherever you like, whether that's on bare metal, in a virtualized environment, or in the cloud. When run in your local data center, containers provide workload density capabilities that can't be matched by virtualization alone. In fact, when HPE tested consolidation of a single host with eight MySQL VM workloads to eight

containers running on an identical bare-metal host, they measured up to 73 percent performance improvement (measured in transactions per second). That's significant! Plus, with containers, workload portability takes on new meaning because shifting workloads between hosts and between local hosts and cloud-based ones becomes trivial.

Containerized Application Support

With the infrastructure discussion out of the way, I'll turn your attention to the reason you're here: workloads. After all, without workloads, you wouldn't need containers, virtualization, or even bare-metal servers.

Traditional applications

Those new to containers are often under the misconception that they work only with brand-new cloudlike applications. Not true! In fact, one of the key themes behind Docker's recent efforts is *modernizing traditional applications.*

Moreover, deployment of applications is getting easier as more images are added to the Docker Store and as more traditional applications are containerized for easy consumption.

Take, for example, Microsoft SQL Server. With just a few simple commands, you can pull a Linux-centric version of SQL Server from the Docker Store and be up and running in minutes. SQL Server is a great example of a traditional, enterprise-class application that you can deploy in Docker in just minutes. In fact, it's easier to deploy SQL Server on Docker than it is to deploy it as a legacy application on Windows!

As you can see, this is all made easier with the Docker Store. Think of this as the app store for all things container. Need a robust graph database? Simply download the neo4j image from the Docker Store. Need Oracle Linux? There's a container for that, too. Plus, if you're worried about security and availability — and, these days, those worries are necessary — Docker Enterprise Edition can get certified containers via the Docker Store. Multiple categories of Docker Certified technology are available:

- **Certified Infrastructure:** This includes operating systems and cloud providers that the Docker platform is integrated and optimized for and tested for certification.
- **Certified Container:** Software providers can package and distribute their software as containers directly to the end user. Better yet, these containers are tested, built with Docker recommended best practices, scanned for vulnerabilities, and reviewed before posting on Docker Store.
- **Certified Plugin:** Networking and Volume plugins for Docker Enterprise Edition (EE) are now available to be packaged and distributed to end users as containers. These plugin containers, built with Docker-recommended best practices, are scanned for vulnerabilities and must pass an additional suite of API compliance testing before they are reviewed and before posting on Docker Store.

PULLS, IMAGES, AND THE DOCKER STORE

There are a lot of terms associated with Docker, and you may be unfamiliar with many of them. I explain much more about these terms throughout this book. But here's a quick intro:

- **Pulls:** A *pull* is the act of downloading an image from a registry, such as the one provided by the Docker Hub.
- **Images:** A Docker container running in your environment is the instantiation of an image. An *image* is a static file that contains all the elements that allow a container to operate. These elements include a file system and other parameters necessary for whatever workload will run in the container. Images never change and they don't carry with them any state. Images can be as simple as a single command or as complex as housing an entire application.
- **Docker Store:** The Docker Store is a free service that provides a marketplace for enterprise developers and IT operations teams to access third-party containers, plugins, and infrastructure.

Chances are, you've heard of *physical-to-virtual* (P2V), which is a process that converts physical servers to virtual images that you can then run on top of a hypervisor as a virtual machine. With containers, tools are available that can help you convert traditional applications to containerized ones, making it far easier to advance in your containerization journey. You learn more about such tools in Chapter 4, but I mention them here so that you can see that, although it used to be challenging to get to a containerized state, it's becoming far easier.

Emerging and cloud-native applications

Traditional applications, while still important, are *so* yesterday. Today's from-scratch application development is a far different beast. No longer are monolithic client/server applications being developed on a regular basis. The reasons are many and include lack of client portability, which precludes support for certain devices.

Development shops now focus their efforts on cloud-native tools that can be consumed from any web browser and from any device. The browser is the client.

Modern applications are developed using far different server-side practices, too. Whereas traditional applications are often one huge piece of software, modern applications are broken up into far smaller chunks to enable scaling of individual components. This also allows developers to implement security capabilities deeper in the code base.

REMEMBER

As the software development framework changes, so does the overall way that releases are handled. In many instances, what used to be an occasional big event has now transformed into a series of ongoing updates that take place on a very regular basis. Take, for instance, Microsoft Office 365. There is no "Office 365 2017" or "Office 365 2018." Instead, users are treated to regular ongoing updates behind the scenes. Each month, Microsoft makes updates to the service and then makes users aware of the changes. The continuous improvement approach is becoming far more common in software circles.

IN THIS CHAPTER

- Seeing how containers support modern development release processes
- Understanding why applications and infrastructure need to be more tightly coupled
- Digging into the hot topic of workload portability

Chapter 2
The Benefits of Application Containerization

Containers bring with them a bunch of benefits. In this chapter, I outline a number of these benefits and show you how they can help your IT function be more efficient and less expensive.

Application Development Processes and Capabilities

Modern data center infrastructure solutions are helping to drive changes to the way that software is developed. Between the promise of the cloud and local data center container deployments, today's developers really have it made from an infrastructure standpoint. At no time in history has infrastructure been so available and so able to be manipulated.

Continuous deployment, integration, and innovation

Increasingly, companies are adopting continuous deployment processes rather than spending years developing the "next version" of their products. This trend really got underway thanks to the plethora of Software as a Service (SaaS) tools that are on the market. With such tools, customers don't need to go through periodic upgrades, which can often be incredibly disruptive. Instead, they get far more regular and much smaller enhancements and bug fixes that are more digestible. This is good both for IT and for users who no longer have to assimilate mass changes to the products they use.

REMEMBER

Containerization — coupled with microservices — works behind the scenes to assist in this agile software development process. In a microservices-centric environment, large monolithic software products are largely eliminated in favor of a software architecture that is more distributed and focused on smaller individual units. This helps to isolate functionality, which makes iterative development efforts work like a charm.

Continuous development also calls for continuous integration (CI) and continuous deployment (CD). Traditional deployment processes typically assume that an infrastructure environment is already fully provisioned and awaiting application code to be deployed and tested. When adopting a CI/CD approach using containers, the entire deployment process — including both infrastructure and application — is deployed within the overall CI/CD pipeline.

Typically, CI/CD processes are implemented as part of the application development life cycle. You need to ensure that the whole team is involved from the outset. This enables requirements to be captured that cover both functional requirements defined by the business and nonfunctional requirements to ensure that the application can perform to meet demand and regulatory requirements

That's where DevOps comes in.

INFRASTRUCTURE AS CODE

The promised land for many organizations — even if they don't know it yet — is what is referred to as *programmable infrastructure* or *infrastructure as code.* With the right underlying physical infrastructure, such as products from HPE, developers can easily manipulate infrastructure just as they do with any code that they write. Where Docker comes into play here is that containers provide the workload runtime for the environment. With programmable infrastructure, developers can programmatically deploy more resources into the data center and, from there, spin up more containers to address, for example, increased traffic.

Closer integration between development and operations

Historically, developers and IT have sometimes had a contentious relationship. Developers blamed IT for providing shoddy infrastructure, and IT blamed developers for writing shoddy code that brought systems to their knees. Container-based solutions can help to close this gap by enabling a build → ship → run methodology. This can be accomplished by allowing far more integration between applications and infrastructure.

The end result is a far closer relationship between development and operations than has been seen in the past. In fact, there's even a name for this phenomenon: DevOps. DevOps is a mind-set in which developers and operators work closely together to ensure that software runs well on production systems.

Modernized Applications for Improved Operations

We're now living in a software-defined world in which we have application programming interfaces (APIs) for everything and anything in the data center. We're also living in an era in which computing power is plentiful and we can move previously hardware-centric capabilities into a far more flexible and

programmable software layer. While all this is happening, we have new infrastructure opportunities readily available, including containers.

You can now harness all this power and transform it into changing the relationship between hardware and software. Let's take a look at a couple of possibilities.

Imagine an application that has wildly variable infrastructure needs depending on the time of day or time of year. Perhaps it's an ecommerce application or a tax-filing application. Each of these kinds of applications places demand on infrastructure that is highly variable. In December, for example, the tax-filing system has almost no demand, but that changes really quickly in April.

With traditional infrastructure approaches, you'd need to build a data center that could handle the peak demand from these applications. In other words, you'd have to throw a whole lot of money at hardware that might simply sit idle for the majority of the time. Plus, you'd need staff to keep an eye on usage patterns to determine when they need to spin up systems to get additional servicing capacity.

Also on the development front, what if you could employ new resources only when you need them? For example, suppose that Apple is using containers to power Siri. Each time someone makes a new request of Siri, a small container-based application is launched to handle that request, after which it's shut down and removed from use. Because containers boot *very* quickly, this is possible. Imagine trying to do that with a virtual machine! A user would have to wait for minutes for a virtual machine to boot before the request could be processed.


TIP

What if your application could send the point at which current resources are becoming slim and simply launch new resources in the cloud for you? Maybe the web tier is getting overwhelmed and you need to be able to process more HTTP requests. With a fully programmable infrastructure, this kind of capability is at your fingertips.

Resource Efficiency

In Chapter 1, I provide a diagram of how containers can improve overall hardware utilization. This is an element of the technology that can't be overstated. Even with the gains that have been made with hardware efficiency under traditional virtualization methods, there's still a lot of room to improve. Typical virtual machines run at very low utilization and, worse, are often oversized by administrators. Oversizing virtual machines uses up resources that could go to other workloads that need it and can also result in the business overspending on hardware.

And, again, every virtual machine needs its own operating system. All this redundancy leads to a lot of inefficiency.


TIP

By moving abstraction from the hardware level (virtual machine) to the software level (operating system), containers make it possible to cram far more workloads onto far less hardware as compared with traditional approaches. Less hardware means less cost overall.

Workload Portability

On a virtual machine, moving workloads from place to place requires some time because you're also carrying a full operating system (OS) along with it. Plus, unless you're using some kind of a translation tool, you need to have the same hypervisor running on both sides of a migration process. However, by abstracting inside the OS, as you shift workloads from place to place, you're only taking exactly what that workload needs to operate, and the hypervisor doesn't really matter. (Of course, the container engine, which drives the container environment and which I discuss in more detail shortly, *does* matter in this case.)

This smaller chunk of data means that you can move workloads far more quickly than before. Plus, thanks to the way that abstraction happens in a container world, you effectively get a "package once, run anywhere" deployment process. This portability enables developers to move from dev, test, staging, and into production and avoid the dreaded "Well, it runs on *my* machine" issues that can plague troubleshooting efforts. IT operations teams can move workloads across environments with ease.

A PORTABILITY PRIMER

The size of the workload is important, but portability of container-based workloads isn't a guarantee. It depends on how you've deployed the technology and is dependent on the engine. Today, a number of container formats are available to developers, and they're not all compatible with one another. I recommend that you standardize on the Docker Container format, which is widely adopted. By doing so, you ensure the widest workload portability options between on-premises deployments and cloud-based ones.

Infrastructure Agnosticism

The best part about containers is that they take hardware agnosticism to the next level. Virtualization already does this to a point. As long as you're running an x86-based system, you can run a virtual machine on it and, in general, move workloads around. However, there are some limitations, such as processor family incompatibilities, that can limit virtual machine portability.

REMEMBER

As you move to Docker-based containers, these limits generally go away. In addition, you can still choose to run Docker on your hypervisor of choice, so you don't even really have to give up virtualization to go the container route. Even better, you can shift Docker workloads from bare metal to a virtual machine and then to the cloud.

And, best of all, you don't need a developer to do all this. You can begin to containerize some of your existing web applications immediately and start getting the benefits of the technology. Most important, this gives enterprises the ability to choose the best infrastructure that works for their business needs. If you're stumped and don't know where to start, you're in luck, too! There are companies out there — such as HPE — that have gone down the container road, and they have the hardware, software, and services that you need to get a jump start on your own container initiatives.

IN THIS CHAPTER

- Paying attention to the people side of containers
- Understanding how containers may impact your processes
- Seeing how containers affects your underlying technology

Chapter 3
The Impact of Containers

Every new technology has some kind of impact on IT and the organization. Sometimes that impact is good and sometimes it's, well, not so good. In the case of containers, however, things are pretty positive. In this chapter, I show you what you and the other people in your organization can expect as a part of your containers deployment.


TIP

Understanding the combination of simplicity and complexity that comes with containers is important. You need to plan upfront and begin to ensure that your staff have all the skills and knowledge they need around containers. In the meantime, and if you're not able to get the expertise in-house, consider working with a partner such as HPE to accelerate your container deployment.

The Impact on IT Staff

Fact: Container adoption at the product level won't directly impact IT staff all that much. Such adoption simply changes the tools they're using, but in many organizations, that happens frequently anyway. After all, the one constant in the world of technology is that there is no such thing as a constant. Everything changes.

That's not to say that IT folks will simply forge ahead with containers. There is a natural tendency to stick with what we know, a cultural paradigm driven by the desire for IT pros to keep systems operational 24/7/365. There is also a control issue. In many current architectural options, IT is the focal point.

Let's take a quick trip back in time. There was a day when virtualization did not exist in its current form. During those dark days, intrepid IT visionaries charted a path through the virtualization wilderness, ultimately taming it so that organizations around the globe could enjoy the outcomes. After some time, virtualization became a truly disruptive force and, today, it's hard to imagine ever going back to what we had before. No longer do IT pros need to rack and stack dozens or hundreds of servers for a single application. Now, they just tap on the mouse a few times and — *boom!* — instant servers.

Virtualization truly improved the lives of admins, but at the beginning, IT pros were wary. It seemed too good to be true. It took time for risk-averse IT pros to warm to the technology, but they did. And now, virtualization is standard in the data center.


TIP

A similar story will be written about containers. For container adoption to become widespread and for organizations to modernize their application infrastructures, applications will need to be able to support the technology and skeptical IT pros will have to be won over. With containers, you can run your application on any infrastructure — virtualized or bare metal — which allows both development and operations to manage their portion of the application life cycle with the tools they require, without having to change the application code. The end result is a streamlined, more collaborative environment.

Operations can quickly and easily provision development environments with containers, or they can allow development to self-manage their own environments, freeing time from the operations workload. Similarly, development can share knowledge about, for example, the applications they're producing, and the debugging tools and techniques available to them. This greatly reduces the time spent by operations when supporting production systems, because they're able to solve problems faster.

There's even a word for this culture change; it's one that describes the merging of development and operations: *DevOps.*

The Impact on Processes

As was the case with virtualization, containers bring with them a number of positive impacts on processes. The first revolves around management. Under traditional processes, IT operations has a lot of responsibility, which is often not shared with development or others in the organization. Some traditional processes include ongoing virtual machine deployment, regular operating system patching, new physical server deployments, synchronizing test/dev and production environments, managing security, and a lot more. For small installations, this may all be doable with relatively few people, but at scale, new tools are needed.

On the management front, tools such as Docker Enterprise Edition running on HPE hardware can provide agility, portability, control, and security needed to run container environments at large scale. Moreover, there are other indirect, but really important internal IT process improvements that can accompany container deployments. Here are two big ones:

- **"Patch Tuesday" impact is reduced.** Every month, Microsoft releases a swath of updates intended to close holes in its operating systems and application products. With fewer underlying operating systems to manage thanks to increased workload density, Patch Tuesday's impact is reduced.
- **The server refresh cycle is eased.** More workload density equates to fewer servers, which results in overall reduction in workload related to hardware refresh cycles. Also, the abstraction provided by the Docker Enterprise Edition makes shifting workloads to new hardware a breeze.

The Impact on Technologies

It's not just people and processes that are impacted by the deployment of containers. The very technology that you're using might be impacted, too. With great flexibility comes great opportunity.

Want to change hypervisors? No problem!

A hypervisor change is a huge pain in the neck! It's one of the reasons they don't happen often. But, when they do, you have to convert each and every virtual machine to the new virtual machine format and migrate to the new platform. Some downtime is more than likely as a result. Most organizations can handle a little bit of downtime, but many can't, and the kind of disruption imposed by essentially ripping and replacing underlying virtual infrastructure can be significant.

That said, there are often really desirable cost incentives to consider a hypervisor change, particularly because the leading hypervisors at this point have practically the same features and general capabilities.

HPE IT: OR MAYBE NO HYPERVISOR AT ALL!

Docker containers can run on bare metal just as easily as they can on a hypervisor. Data center architectures can help bring to these bare-metal deployments the kind of flexibility that is often considered the sole purview of virtualization.

Sean Sargent and Elliot Berg, Solutions Architects at HPE Pointnext, HPE's consulting arm, have deep experience with emerging data center architectures. Interviewed for this book, Sargent and James indicated that HPE has "invested in things like the composable infrastructure API that was exposed to HPE's OneView management tool. We found that if we use OneView, we can remove things such as the hypervisor from our configuration, and run Docker native platforms on our bare-metal infrastructure. We still get a virtual machine–like experience for provisioning and managing. And for the parts that OneView doesn't have, like object store and DNS as a Service, or load balancer service, we can leverage the components from those from other APIs, such as Helion OpenStack."

A full Docker container deployment fully negates the risk of downtime that accompanies a hypervisor change and may allow organizations to adopt less expensive hypervisor solutions. The reason is the Docker Engine. By ensuring that there is a Docker Engine running on both the old hypervisor system and the new one, you can simply shift those containers across to their new home.

The management and monitoring plane

Overall application management and monitoring remains a challenge even today. All kinds of companies are working really hard to provide complete solutions, but the sheer vastness of the application landscape makes a one-size-fits-all approach incredibly challenging.

REMEMBER

Containers brings with them some new options for ongoing management, but you may be required to make a couple of upfront (but pretty easy!) choices:

- **Choose an engine.** Again, Docker is recommended thanks to the fact that it's widely available and supported.
- **Choose a tool for ongoing management.** I discuss these tools in a bit more depth later in this book, but the big ones are Docker Swarm and Universal Control Plane (UCP), Kubernetes, or Mesosphere DC/OS.

TIP

By making these choices early and sticking to them, you can drastically reduce tool sprawl. Doing so allows developers and operations to work from the same toolkit and promotes collaborative working styles.

On the monitoring front, there is another key element to bear in mind: Container-based applications are typically distributed applications. In some cases, containers are big and almost permanent members of the data center, but in many cases, they're also short-lived and run for just a few seconds, carry out their tasks, and are then recycled. In cases like this, using traditional approaches for monitoring is not appropriate because many traditional approaches would see the "death" of a container as a system-down issue, when, in fact, it's perfectly normal. A mind shift needs to take place. You're not necessarily monitoring availability. Instead, you're using centralized logs so that key performance indicators (KPIs) focusing on end-user application performance or speed of transactions within the system are meeting or exceeding established requirements.

HPE POINTNEXT: IMMUTABILITY

Sean Sargent and Eliot James, Solutions Architects at HPE Pointnext, HPE's consulting arm, have deep experience with emerging data center architectures. When HPE Pointnext architects were asked about this topic, they talked about the concept of *immutability*.

What does this mean? In short, immutability is a technical term that means that they can't be changed. A huge part of the reason that IT staff need to deploy so many monitoring tools is because things in the environment change. Change is the leading cause of performance problems and downtime, so eliminating change can reduce a whole lot of issues.

HPE Pointnext architects said:

> Once container-based applications are deployed, the application configuration cannot drift. When applying configuration changes or patches, the current container instance running is not updated. The base image is updated, tested, and then deployed and replaces the currently running container instance.
>
> The time it takes to troubleshoot application issues [is] also greatly reduced. As issues occur, instead of checking the configuration of the application to see what might have changed to cause the issue, you simply deploy the standard, tested and known working version of the container to reverse any changes that might have occurred.

Control and security in a container world

New data center and application architectural opportunities bring about the potential for changing how an IT organization operates. These changes may not be attributes of the new architecture, but they may be brought to the forefront based on what that architecture can do.

Perhaps one of the biggest organizational changes resulting from deploying containers has to do with the fact that developers can now have direct infrastructure control in ways that were exceedingly difficult with more traditional approaches. Of course, this is handled in concert with operations people, but the fact is that developers are far closer than they ever were before.

With the right tools, developers have true infrastructure power right at their fingertips, alongside their operations counterparts. In this software-driven data center model, all the servers and storage and networking equipment that drives the environment are simply software elements that can be controlled by code. But not every organization will require the exact same balance of shared infrastructure management between operations and development. With tools such as Docker Enterprise Edition, operations will still be able to maintain the control they require, while granting developers the access they need to get their work done more quickly.

TIP

In fact, there is a real-world example that best serves to illustrate this, and it's found in the nearby sidebar, "HPE IT: OneView for centralized control."

HPE IT: ONEVIEW FOR CENTRALIZED CONTROL

HPE internal IT has contributed a ton of work to the open-source community and continues to do so. In addition, HPE has extended its OneView product with a Docker plug-in than enables automated deployment and management of Docker containers.

According to HPE Pointnext architects:

> Prior to completing our work, the only thing we were really missing [in order] to integrate it with native Docker tools was a software development kit or SDK. So we invested in creating that SDK for OneView and we first coded for OneView 2.0 and we implemented a plug-in for Docker Machine, which we open-sourced. That's available at GitHub.com. That gave us the first building blocks of creating an SDK that we then applied to more scaled-out solutions for provisioning.
>
> The impact has been very good. For example, one of the teams of developers that manage[s] our infrastructure wanted to learn how to use our playbooks. We did some training with them on inventory management, after which we made the decision to deploy a series of secondary production nodes with the latest inventory files. They spent basically 35 minutes, and were able to deploy a full production cluster within that time. Prior to Docker, this would have been a weeks- or months-long project.

Security is a key concern for just about every company on the planet, especially as we see more and more leaks of confidential information hitting the black market. No matter how you run your workloads, you need to keep security top of mind.

REMEMBER

Here's the current scenario: In the context of containers, people often talk about security as one of the challenges with the technology. However, the devil is always in the details. Keep in mind that containers remain relatively new compared to other technologies, and there will be gigantic leaps forward in all areas.

When people talk about container security, they're often talking about availability. It's true that knocking a container host offline will probably impact more individual workloads than if you take down a virtualization host. That's just the laws of physics at play. You have the potential for far more workloads operating on a container environment, so more workloads are impacted when something happens.

There are a couple of key security items that you should keep in mind with regard to containers:

- **Operating system (OS) compromise:** In a traditional application deployment, an OS compromise via an application would compromise the entire server. With containers, however, while some container exploits have exposed the host operating system, Docker is constantly working on improving security and the container construct itself can provide some abstraction that can thwart certain attacks.
- **Compromised images:** If you download images from untrusted sources, you might be at risk of installing malware into your organization. However, there is a way to avoid this. You have the ability to sign or to check vendor signatures on all the images prior to running them, thereby ensuring the validity of the container image and minimizing the risk of infection from malware. In addition, you can add metadata to the container to indicate whether the image can be run in a production environment.

A seamless hybrid environment: Physical, virtual, cloud

In the world of containers, infrastructure becomes an application platform rather than just data center hardware. To understand

what this means, let's compare a container deployment with a traditional deployment. In a traditional system, you'd have your storage, operations, networking, and database teams creating and configuring virtual machines, databases, and firewalling for you, and then deploying the application onto it. If it doesn't quite work (because the developer's environment was subtly different), they would reconfigure the infrastructure until it does.

Now, consider the myriad options you have with regard to infrastructure — physical, virtual, and cloud. Ensuring 100 percent consistency across all of this is nearly impossible. This consistency comes in the form of the abstraction provided by the container engine.

IN THIS CHAPTER

- Identifying the major components of the Docker architecture
- Seeing how images are constructed
- Discovering the operating system prerequisites for both Docker editions

Chapter 4
Preparing for and Testing Docker Containers

You've learned a lot about why companies want to adopt containers, so let's switch gears a bit into something a bit more technical. In this chapter, I introduce you to all the components that make Docker work.

REMEMBER

There is far more detail inherent in Docker than I'm able to cover in this book, so I'm just covering the big stuff here.

Understanding Docker Container Constructs

REMEMBER

There are a bunch of different components to Docker that you need to understand in order to really put the software through its paces. I'll start with a look at the high-level constructs and tell you what each one is and what it does. In Chapter 5, you have an opportunity to see Docker's installation requirements and processes and see how easy it is to deploy a new container from a registry.

The Docker Engine

The Docker Engine is the workhorse of the solution and consists of three main elements:

- **Docker host:** This is the server that runs the Docker engine as an operating system (OS) daemon or process that lives forever.
- **Application programming interface (API):** The Docker engine's REST API dictates how the program can interact with the daemon.
- **Command line interface (CLI):** The Docker client's CLI is exactly what it sounds like. It's a command line that allows an administrator to interact with the API, which interacts with the server.

Dockerfile

Everything starts with a Dockerfile. This is a list of the exact OS commands that ultimately generate a Docker image.

Images

A Docker image is the result of running the `docker build` command with the Dockerfile as the input. An image is just a static (read-only), nonrunning representation of all the various components that define the workload. Think of an image as a workload template that sits on storage, awaiting the day when it's put into service.

An image can consist of multiple layers, as shown in Figure 4-1. Each layer is a set of file system changes imposed by the Dockerfile. I explain a bit more about how images are assembled later in this chapter.

Containers

When you start up an image on a Docker host, you have a running container. You can have a lot of running containers that are based on a single image. But a running container is more than just a running image. It's an image to which a read/write layer has been attached atop all the read-only information that's present in the container.

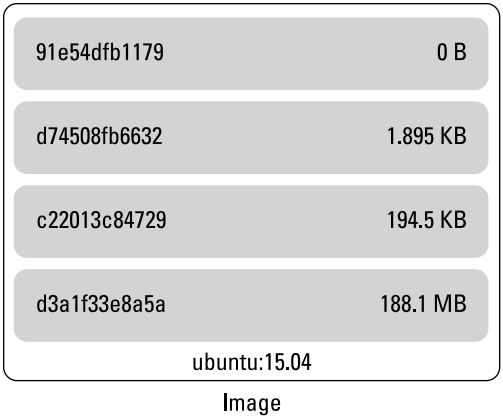

FIGURE 4-1: A Docker image with four layers.

You don't want changes made to an image, but you do want to be able to interact with what's running inside, and that often means that you'll need a read/write file system to make that happen. When a container is started up, Docker adds a thin read/write layer on top of the image to handle this interaction.

Figure 4-2 gives you a look at a running container with this additional layer.

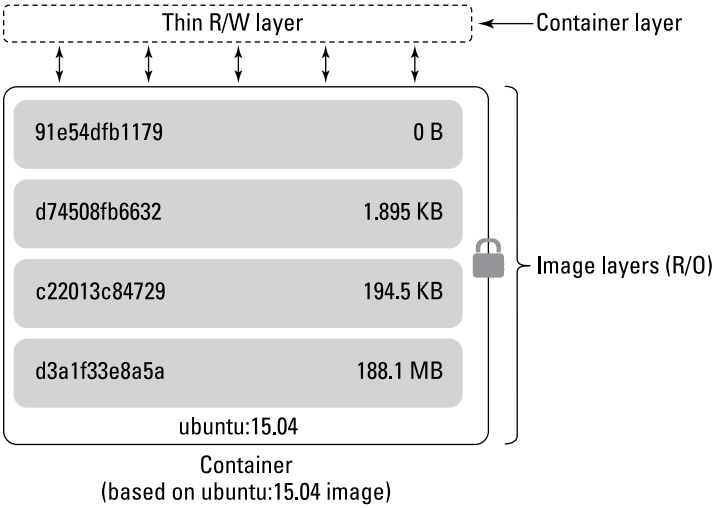

FIGURE 4-2: A running Docker container.

The Registry

A registry is a centralized repository that holds Docker images. Administrators can deploy their own private registry service, which operates in a highly scalable way.


TIP

If you want to fully control where your images are stored or tightly integrate images into your local development processes, consider deploying a private registry, such as Docker Trusted Registry. Docker Trusted Registry is an enterprise-class containerized application that provides fine-grained access control, and security and compliance capabilities.

Under the hood

Let's lift the hood a bit and take a peek at the main items that comprise a Docker deployment.

Namespaces

The Docker Engine leverages multiple kernel namespaces, with each namespace controlling specific functions inside that image or container. A namespace is used to isolate certain OS functionality to limit the scope of access. The end result is that a namespace imbues processes with their very own view of underlying resources.

Docker uses namespaces to separately manage process trees and isolation, file system mounts, interprocess communication (IPC) resources, networking, hostnames, and user access. Namespaces can also limit what a container can see.

Control groups

Whereas namespaces limit what a container can see, control groups (cgroups, for short) are used to allocate resources, including RAM, CPU, storage I/O, network I/O, and more. For example using a cgroup, you can limit a process to just a single CPU core, which would prevent it from consuming additional processing resources that could impact other workloads. By using cgroups, the Docker Engine can dictate to a container what resources a container can use and how much of that resource it can use.

Union file systems

A *union file system* allows the piecing together of files and directories from separate file systems to be overlaid on one another to create a single, coherent file system.

The union file system plays a huge role in the lightweight nature of Docker containers. Let's take the case for updating an application. When you modify an image, you simply swap out a changed layer for a new one, leaving all the other layers in place.

When an image is started up, the union file system attaches a read/write layer to the container.

Docker on Windows

If you prefer or require Windows Server in the data center, you're in luck! Docker support has come to the Windows-based data center. With the release of Windows Server 2016, Microsoft and Docker have provided an additional platform on which to run container-based workloads.

There are, however, some key considerations to keep in mind as you go down the Windows-based container journey:

- **OS version:** Whereas Docker can run on almost all modern Linux variants, only certain versions of Windows 10 and Windows Server 2016 can run Docker. For example, since Hyper-V is required, Windows 10 Home Edition is not an eligible platform, even for testing. Check Docker's compatibility list to make sure your operating system version is supported.
- **Orchestration:** Container orchestration systems that are common to Linux-based Docker implementations aren't always available on Windows. Docker's own Swarm mode, however, is available. Over time, as Docker on Windows becomes more popular, expect to see more orchestration systems become available.
- **Command line:** PowerShell is one of the unsung heroes of the Windows world. In addition to the same CLI used with Linux-based Docker containers, Docker on Windows can also be managed using the PowerShell CLI, easing the transition for Windows users. Of course, if you're a Linux Docker admin making the jump to Docker for Windows, you'll be able to bring over all of your command line knowledge and it will work for you.

Linux and Windows take fundamentally different approaches to kernel design. As such, Docker on Windows doesn't have direct support for things like Linux namespaces and cgroups, which are integral to Docker and which are built into the Linux kernel. To address this, Windows Server 2016 adds what Microsoft calls the *host compute service.* The host compute service replicates this functionality and acts as an API that can be used by the Docker Engine.

Let's turn attention to workload portability between Linux and Windows. There is a lot of confusion around whether you can take any Linux-based Docker container and run it on Windows. Is this supported? The answer, unfortunately, is: It depends.

Prior to Docker support being added to Windows Server 2016, Docker Linux containers ran on Windows thanks to the VirtualBox hypervisor, although this functionality wasn't ever intended to run containers in production. All the containers that appeared to be running on the Windows host were, in fact, running in a virtual machine that used Linux as the OS. As such, Linux containers worked spectacularly. However, it was really aimed at developer desktops, not production. And with Windows Server 2016, VirtualBox is out of the picture, as is the use of a virtual machine running Linux to support Docker.

In fact, with Windows Server 2016, you don't need virtual machines at all to make Docker work its magic. As of Windows Server 2016, Microsoft has added native container support. But there are a few things you should understand.

Most important: *Your Linux-based containers won't run on Windows.* (You know that we're serious about that since we put it in italics.) You can't just pick up your Linux containers and drag them over to Windows. You need to re-create new Windows-based images starting with new Dockerfiles.

Second, even if you were testing containers on older versions of Windows, you can't bring those to your native Windows Server 2016 environment. The reason is Linux. Under the hood, a pre–Windows Server 2016 container deployment was actually running in a Linux VM on VirtualBox. And because you can't simply bring Linux containers to native Docker on Windows Server 2016, you may need to restart your container journey.

A LINUX WORKAROUND FOR WINDOWS

If you've got a lot invested in your Linux-based Docker environment, but need to move to Windows anyway, you can still choose to run those Linux containers under a Linux virtual machine in Hyper-V rather than using the native Docker for Windows tools.

That said, unlike the older methodology, the latest Docker for Windows *can* be used for production workloads, so it's far and away better than what came before.

Containerized Workload Deployment Options

As was the case with virtualization, containerization projects ultimately take one of three forms:

- **Greenfield:** You start from scratch and don't bring over anything from the legacy environment.
- **Full migration:** You have to migrate all your legacy workloads from the old environment into the container environment.
- **Hybrid:** You have some new applications and some legacy applications and need to develop a strategy for deploying both.

REMEMBER

As you might imagine, the hybrid scenario is the most common deployment method for pretty much any new kind of infrastructure.

Migrating Windows-based web applications from virtual machines

Deploying new workloads is pretty obvious, but a whole lot of people will need to migrate legacy applications to containers if they're interested in going all-in on the technology.

Migrating those workloads manually would be a huge chore — about as huge as manually migrating physical servers to virtual machines was just a few years ago. A manual process would mean basically going from scratch. You'd have to install a new OS, reinstall applications, and then migrate all the files and users from the old to the new. It would be highly time consuming and highly risky.

Back then, though, a number of companies came to the rescue by creating what became known as physical-to-virtual (P2V) tools. By running these tools, in just a short amount of time, administrators could create snapshots of a physical server that they converted to a virtual machine and then simply deploy without having to reinstall all the OS and application underpinnings. It transformed a process that would have taken weeks into one that took hours and was critical to the overall success of early virtualization efforts.

For container users, the beginning to such an ecosystem is developing. The first entrant into this space is an open-source tool called Image2Docker. It won't work with every application on your server. For example, you're not going to containerize your Exchange Server with it. However, if you're a development shop and have a lot of web-based applications running across a multitude of Windows Servers with IIS, Image2Docker may be a goldmine for you.

TECHNICAL STUFF

Image2Docker can scan your WIM, VHD, and VHDX files — all which are standard Windows Server deployment template or virtual machine formats — and extract from them a Docker image for each website in each location. In other words, if you have ten such files, each with ten ASP.NET web applications running in separate IIS instances, Image2Docker can automate the creation of a Dockerfile for each of these 100 applications for you.

Again, you're not bringing Exchange to Docker (yet), but there is nothing stopping you from bringing containers to your in-house applications.

REMEMBER

Although container deployments aren't quite as straightforward and "no brainer" as virtualization deployments were, companies like HPE and Docker have significant resources that can be brought to bear for you, so you don't have to go it alone.

Discovering Operating System Prerequisites

As you now know, Docker runs on top of a myriad of operating systems, with both Linux and Windows Server options. And, thanks to the Unix underpinnings of macOS, you can run Docker on that OS as well. Figure 4-3 provides an overview of Docker OS support as of this writing.

Platform	Docker EE	Docker CE
Ubuntu	●	●
Debian		●
Red Hat Enterprise Linux	●	
CentOS	●	●
Fedora		●
Oracle Linux	●	
SUSE Linux Enterprise Server	●	
Microsoft Windows Server 2016	●	
Microsoft Windows 10		●
macOS		●
Microsoft Azure	●	●
Amazon Web Services	●	●

FIGURE 4-3: Docker OS support.

You might be wondering what is meant by Docker EE and Docker CE. Docker is broken up into two products:

- **Docker Enterprise Edition (EE)** is the enterprise container platform designed for enterprise development and IT teams who build, ship, and run business-critical applications in production at scale. Docker EE comes in three tiers: Basic, Standard, and Advanced. Basic is what ships by default on HPE Proliant servers. Docker EE Standard adds image management and Docker Datacenter. Docker EE Advanced wraps it all up by providing the additional Docker Security Scanning technology.

- **Docker Community Edition (CE)** is the freely available version of Docker. Docker CE is often used for development purposes, as it doesn't offer the support options available with Docker EE. In this way, Docker CE is like many other open-source projects. There is a free tier, but as you need support, you can either turn to the community or choose from among vendor-provided options.

Planning Docker Resource Requirements

Each OS carries with it separate requirements that you need to consider. Plus, just like any other application environment, Docker containers need RAM, compute, and storage resources in order to operate.

On the RAM and CPU front, there's not a lot of difference in how you plan resources, but there may be a big difference in how those resources are used. With virtual machines, resources are started up and they stay up for long periods of time. With containers, they may live for a long time, but more often than not, they survive for a short period of time to carry out their singular function and then they're killed. They consume and release resources very rapidly. The single-service, short-lived, lightweight nature of Docker containers means that you can probably dial in your resource requirements in a more exact way than you could before, but this will take a bit of trial and error on your part as you work out what your containers need.


TIP

There is also the question of failure domain to consider. Underlying hardware will fail. So, as you architect your environment, be careful about putting too much on one server. You may be better off buying a few more servers that are a bit less powerful than buying just one or two monster servers. This will help ensure high availability.

Although it's getting better, storage is not as straightforward as it is in the world of the virtual machine. As you plan your storage requirements, make sure that you understand how your existing or planned storage will connect to your Docker cluster. We'll talk more about Docker storage in Chapter 5.

Regardless of how you connect to storage, the often short-lived nature of a container means that latency is an enemy. The longer it takes to instantiate a container or for a container to release its resources once its work is done, the slower the underlying service will operate. Speed is king in the world of containers. Especially today, with the ubiquity and cost-effectiveness of flash media, buying storage that is either all flash or hybrid — a combination of flash and spinning disk — is highly recommended, regardless of application.

As Docker continues to mature, expect to see improvements in storage services. More on that topic shortly.

Before I move on, though, I want to talk storage capacity. In a virtual machine world, let's assume that you have 100 virtual machines, each with a 10GB virtual disk for the OS. That equates to a terabyte of storage used just for the OS.


TIP

In the container environment, you have a single OS instance, so that takes up 10GB. Each container in then just a fraction of the size of a virtual machine and could be just a few megabytes. You can quickly gain capacity savings by moving to containers. In fact, in a study carried out jointly by Docker and HPE, containers saw a greater than 30% savings in storage capacity when compared to certain virtual machine configurations.

IN THIS CHAPTER

- Installing the Docker Engine
- Deploying the "Hello, World" container
- Seeing what makes a Docker enterprise deployment tick

Chapter 5
Docker Installation and Deployment

In this chapter, you actually get Docker up and running and install the "Hello, World" container — a test container that you can use to verify that you've successfully deployed Docker.

Installing Docker

TIP

For this exercise, I'm installing Docker for Mac. To get started, I need to download and install the engine. Each operating system (OS) has a different download. To get the right one for your system, visit `https://docs.docker.com/engine/installation/#platform-support-matrix`, click on your OS, and follow the installation instructions. Make sure you pay attention to the general system requirements on the instruction page and ensure that your system meets the stated requirements. Also, make sure you follow Docker's installation instructions to the letter to make sure you don't miss something.

I'll wait for you to finish your installation before continuing.

Done yet? Good! Now, let's move on to the next step. Again, I'll be on my trusty Mac, so you'll see Mac screen shots and output and I'll be using the built-in Terminal application to issue Docker commands. If you're a Windows diehard, you can choose to use PowerShell, but you don't have to. The Docker CLI is also directly available on that platform and using it has the added benefit that you'll become an instant cross platform Docker management guru!

My first test is to make sure Docker is really running, so I'm issuing the command `docker version`, which results in the output shown in Figure 5-1.

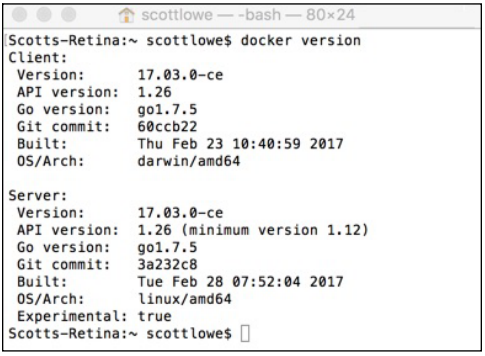

FIGURE 5-1: The Docker installation was successful.

Now, are you ready for the absolutely arduous process of downloading an image, installing it, and then executing it and hoping it works?

At your command line, type **docker run hello-world** and press Return.

Yep, that's it. Figure 5-2 shows you what just happened.

You didn't actually have to go anywhere to do anything. After you issued the Run command, Docker looked for a local image named hello-world. Not finding it, Docker pulled the hello-world image from the Docker Store. It then created a local container from that image and executed it.

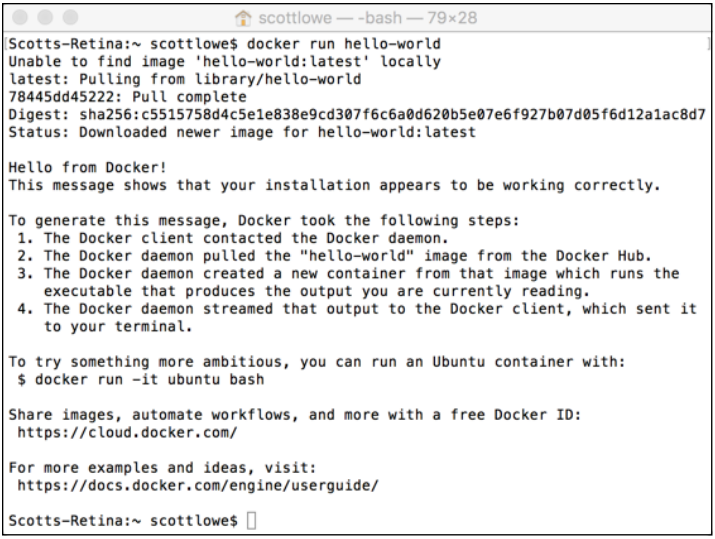

FIGURE 5-2: The "Hello, World" container is running.

Understanding Docker Enterprise Deployment Requirements

So Docker is super easy to get going for brand-new containers, eh? Well, in a lab or test environment, you can get away with typing a few commands and getting your first container launched. In an enterprise environment, you need to consider far more.

Security

One of the main benefits of containers is unleashing your developers to become more productive and create value. On the other hand, you still need to have in place controls if you want to deploy workloads into production, especially in regulated industries. You need to give consideration to compliance and separation of duties between developers and operations staff.

REMEMBER

Governance needs to be put in place to standardize on images and updates used within your environment.

That's all handled outside Docker before you even type your very first run command.

TIP

To maintain agility, you should consider introducing a security element into any DevOps you undertake. Security processes can be automated using tools that have published application programming interfaces (APIs), which can be integrated as part of your continuous integration (CI)/continuous deployment (CD) pipeline.

Docker containers are quite secure by default, and you can increase their security by following a few best practices.

- **Manage your container-based processes using non-privileged user accounts.** In other words, don't use *root* for everything.
- **Make sure to use trusted images.** A compromised image can get past even your best security measures. Make sure that the source from which you obtain images is trusted and secure. It's often recommend that enterprises undertaking Docker projects use their own private registries. If you're using a public repository, make sure downloaded images have a valid signature. Docker makes it super simple for you to sign your own images, too. Whereas a security authority in traditional infrastructure can be a beast to manage, Docker Datacenter includes a built-in notary server and already has the infrastructure set up to do this for you. All you need is Docker EE Standard or Advanced, which includes Docker Datacenter, and you'll have all the infrastructure you need to sign your own images and keep your environment safe.
- **Consider additional security layers.** You can use tools such as Security-Enhanced Linux (SELinux) to harden your Docker hosts. You can also use Docker Bench, which is a script that checks for dozens of common security best practices around deploying Docker containers in production.

Storage

If you're undertaking a Docker enterprise deployment, you need to consider storage. Just as you did for physical servers and virtual machines, you need a place for workloads to reside and store their data.

Here's how storage works in Docker: In a container, if you make changes during runtime, those changes are saved in the read/write layer that is attached to that container when it's initialized.

Any changes to that specific container are retained even between starts and stops of that container.

REMEMBER

If you ever remove the container, however, say goodbye to your data. This behavior is where the myth that "containers are stateless" originated. Although this is true to a point, containers are far from stateless while they're running and, often, the service running inside the container needs to be able to store data that will persist even upon the deletion of that container.

This is where *data volumes* come into play. A *volume* is a storage target that resides outside a container. Data saved to a volume persists long after a container has left this world. You can use these data volumes across containers.

You can also mount directories from the host's file system, but this storage method is not recommended because you can't use this technique in Docker files. Instead, you must connect to the directory at runtime.

If you're coming back to Docker after a hiatus, you may be used to building a container and then adding a volume, which then stays around even if the container goes away. Today, that's all a bit different. In modern Docker, volumes are their own standalone atomic units, and you can always attach containers to it. Volumes now remain persistent without the need for administrator intervention.

But you probably already have a storage area network (SAN) that you like, and you might want to be able to leverage some of its built-in capabilities, such as data reduction and replication. Plus, standalone SANs provide far more scalability than other storage models.

What if you'd like persistent storage that can span hosts? Although you can use OS storage protocols such as NFS, having the ability to more deeply integrate with your storage vendor is increasingly important and desirable.

TECHNICAL STUFF

Docker and the open-source and vendor communities are making available a growing number of *volume plug-ins.* Plug-ins are tools that extend the capabilities of the Docker Engine. Volume plug-ins can enable you to leverage external storage that you can easily share among all your Docker hosts.

Plug-ins are available for major storage providers, such as HPE 3PAR (`https://github.com/hpe-storage/python-hpedockerplugin`).


TIP

The real point is that you can continue to use your existing direct-attached, iSCSI, and Fibre Channel, or NFS storage system with ease. You can also unlock a lot of new options. For example, you can create a persistent volume at Amazon and use that for your persistent storage.

Scale

Eventually, you need to be able to scale your workloads. In this context, this means being able to horizontally scale stateless workloads within seconds using containers. The speed by which you can scale containers is a great advantage over scaling virtual machines horizontally, which takes multiple minutes and consumes a larger infrastructure footprint and increased management overhead.

Greater density and rapid scale-out capability provide huge advantages over traditional methods of deploying applications, but you need to consider the infrastructure and application architectural design, including workload placement, to ensure availability of the applications being hosted in the containers.

Imagine a situation where you achieve high levels of density of containers per container host and then the container host suffers a failure. A large number of applications would be impacted, unless you took appropriate planning steps ahead of time.

As you scale, you also need to make sure that you have tools that can help you manage multihost environments.

Swarm mode

In Docker parlance, a *Swarm* is a cluster of Docker engines (nodes) that are running across one or more Docker hosts. You can run multiple containers in a single host, but larger implementations will almost certainly also have multiple physical, virtual, or cloud-based host servers.


REMEMBER

Swarm mode (or a tool that can replace Swarm, such as Kubernetes) is particularly important as you scale because it provides, among other services, key load balancing capabilities that ensure that incoming requests are fairly distributed to all the workers.

In a Swarm, a *worker node* executes a task assigned to it by a *manager node.*

In Swarm mode, your Docker cluster also provides multihost networking capabilities, which automatically assigns network addresses to containers when they initialize.

On the security front, nodes that participate in a Swarm use TLS encryption between them in order to secure communication and keep your data safe from prying eyes.

Universal Control Plane

Docker's Universal Control Plane (UCP), part of Docker EE Standard and Advanced, is an enterprise-grade cluster management solution that adds a graphical user interface (GUI) and security features atop Swarm and also includes some monitoring capabilities.

Modernizing Traditional Applications

The Hello, World container is incredibly simplistic, and, although it does showcase just how easy it is to get Docker running, it doesn't do justice to just how large a container can really get. The goal for most companies is to get to market faster with their products and services, a task that can be assisted by modernizing what they already have. In fact, according to Docker and HPE, three out of every four companies want to modernize their existing diverse application portfolios, which may consist of commercial applications, monolithic custom applications, and microservices. Docker's Modernization of Traditional Applications framework empowers the development team to focus on a framework that allows for continuous development and integration. This results in faster time to market to value.

Organizations' efforts to containerize and modernize traditional applications revolve around addressing these core needs:

- Agility
- Portability
- Security

At the same time, companies are working hard to transform monolithic applications to microservices-oriented architecture. Why? When you think about something large and static, there is also a feeling of lack of inertia. That is, getting that thing underway is going to take a ton of effort. If you could just break it up into smaller, nimbler components, getting those parts underway individually would be much easier.

This is what's happening today. Large applications are difficult to support and update and are being broken apart. Whereas these monolithic applications required large virtual machines or containers to operate, their new microservices-based versions are spread across many smaller containers. This makes the application far easier to maintain because there isn't a single massive construct to work with anymore and it makes the individual components far easier to scale as workload demands increase.

These are just some of the reasons that companies are working hard to modernize what they have and migrate their business-critical applications to Docker.

IN THIS CHAPTER

- Getting tips straight from HPE IT about the tools you should deploy
- Adapting your IT and development culture to best support containers
- Seeing what a post-container world might look like

Chapter 6
Ongoing Management, Advanced Deployment, and Tips for the Future

Much of the information in this chapter comes right from HPE IT. Throughout this book, you've seen snippets of information provided by these experts. As pioneers in the use of containers, HPE has gained deep experience into the use of Docker Enterprise Edition, contributed a great deal of its work product to the open-source community, and applied what it has learned to its products, including integrations with HPE OneView, 3PAR storage, Synergy composable infrastructure series, and C-class blades, among other products. The advice in this chapter is from the practitioners who have paved the way for you.

Ongoing Management

TIP

Before you do anything, make sure you get the tools right. Too many IT departments have deployed way too many tools because they didn't plan things out well in advance of deployment. If you standardize on formats for containers and tools for ongoing management (such as Kubernetes, Mesosphere, or Docker Swarm and Universal Control Plane), you can drastically reduce tool sprawl. Doing so allows developers and operations to work from the same toolchain and promotes collaborative working styles.

Knowing What Is Needed

Before delving into container technology, you need to understand what you're trying to achieve. Are you looking for greater agility? If so, map out your existing processes to understand what part of the process is slowing you down. Next, assess the types of workloads that are part of this process. Are they suitable for containerization? For example, are they mainly stateless?

TIP

When you understand the areas for improvement and know that the workloads are appropriate for containerization, look to introduce automation in your nonproduction environments and use containers to implement and automate a continuous integration (CI) process. CI is an ideal use case for containers, because the build/test/package steps are generally short-lived and transient — exactly like containers.

HPE IT: RETHINKING YOUR APPLICATION ARCHITECTURE

Containers require a new general mind-set about how applications function in the data center, particularly because the walls of the traditional data center are often shattered due to the way in which containers can operate. Architect for resiliency at the *application layer,* as opposed to the infrastructure layer. This is essential when deploying containers at density.

Aiming for Reasonable Rollout

Go for a crawl, walk, run approach to container adoption. Start small (crawl) with simple, well-known or proof-of-concept applications or processes. Then expand (walk) by ensuring all operations procedures work with containers. And finally (run), look at implementing modern application architecture design using containers.

Getting the Culture Right

TIP

Containers are often adopted to increase agility. In a traditional environment, technology teams often operate in silos. This means there are many handoffs between teams, which causes delays and often misconfiguration. When using containers, adopt cross-functional teams to minimize handoffs and to increase the overall awareness of the environment that the applications are being designed, developed, deployed, and operated in. When creating cross-functional teams, make sure that the team understands — and is measured on — the outcomes that you're trying to achieve.

HPE POINTNEXT: CULTURE CAN'T BE OVERSTATED

The entire team needs to be involved from the outset when adopting containers. Bringing team members in halfway through the adoption leads to confusion, because the wider teams (Dev/Projects and Ops) requirements need to be considered throughout. Consider using communication techniques used by Management of Change practices to share information regularly. It has been widely recognized that as teams (especially cross-functional teams) grow, communication becomes harder. A technique used by some high-performing organizations is the *two-pizza team* concept. This approach states that no team should be larger than what two pizzas can feed (typically, eight people). This approach ensures that communication between team members is effective. However, it also requires that team members have a broad skill set and learn rapidly from each other in order to consider every level of an application and infrastructure stack within a cross-functional team.

REMEMBER

Key performance indicators (KPIs) must be defined and measured to ensure that the cross-functional team is fully aligned.

Recognizing That Containers 1.0 Is Just the Beginning

We often look at new technologies and wonder how we ever got by without them and what could possibly be better. We look at containers today and see so much more workload density, scaling, and portability potential that it can be hard to realize that it can get even better.

Hot on the heels of containers are *unikernels*, or what some refer to as Containers 2.0. They're the next step in the evolution (revolution?) of the data center development and application landscape, and with good reason.

Containers go light years beyond virtual machines when it comes to a number of factors, but there is still a complete host operating system (OS) to deal with. Many "on by default" services are inefficient and expand the attack surface of the system, making things more vulnerable than many people would like.

Unikernels go to the next level. They're fully self-contained alternatives to traditional containers that are tailored to the needs of modern microservices-centric development. They contain just the low-level features that are needed by whatever microservice is operating. Because there is no host operating system, if a unikernel-based service is breached, there's no easy pathway to a host operating system and the other containers running on that host.

IN THIS CHAPTER

- Looking at real-world examples of containers in action
- Recognizing the benefits of Docker and HPE

Chapter 7
Containers in Action

If you've been head down putting out fires and struggling to help keep IT aligned with the business, you can be forgiven if you haven't seen much happening around containers. After all, it's human nature to focus on what's in front of us and on things we're familiar with. However, it's really important for you to see what some others are doing — *today* — with containers. This isn't the future folks. It's the present.

The most powerful way to understand the potential of a technology is to see it in action. You probably can't just jump in your car and drive to some other company and beg to see its container deployment. So, the next best thing is to look from afar at what others have done. Here are just a few examples from across five wildly different industries. This chapter illustrates that Docker and HPE are providing a transformational opportunity, regardless of industry.

Nonprofit: HudsonAlpha

HudsonAlpha is a nonprofit institute that translates the power of genomics into real-world results. HudsonAlpha is a joint Docker Enterprise Edition (EE) and HPE Synergy customer.

HudsonAlpha was facing a myriad of challenges, from scaling woes to security worries:

- Needed to accommodate thousands of diverse research-centric web applications across 38 facilities
- Wanted portability of applications across a hybrid cloud operating environment that was based on HPE Synergy and Google Cloud
- Required compliance with security mandates for security identity and applications
- Wanted to bring ballooning hypervisor licensing costs under control
- Wanted to remove dependency challenges being experienced by developers when building applications

HudsonAlpha turned to HPE and Docker for a solution and chose to deploy the Docker Enterprise Edition container platform running on HPE Synergy hardware. The benefits were immediate:

- **Portability:** HudsonAlpha is able to migrate workloads across its hardware and cloud environments.
- **Security:** HudsonAlpha can far more easily comply with security mandates.
- **Cost savings:** The organization now delivers new service far more quickly at a lower cost.
- **Standardization:** HudsonAlpha now has a standard platform for all apps and across hybrid infrastructure, eradicating all dependency issues.

Finance: ADP

Managing over 55 million Social Security numbers moving about $1.8 trillion per year, ADP needs technology that is secure and just works.

The ability to use "hardened containers. is critical for ADP," because, inside those containers moves very sensitive information. Docker enables ADP to know what's running inside the

container and where that container originated so the ops team can keep tabs on what's running in production.

The ADP infrastructure has different environments with different security levels, enabled by a progressive trust workflow with multiple Docker Trusted Registries (DTR). The first DTR is the "whatever goes" registry that allows for complete developer creativity. The second DTR is the "we think we want to run this" registry that is separate and allows for the team to vet and validate new applications. The last DTR manages containers that will be deployed to production. ADP was able to deploy Docker in a flexible way to allow for both developer freedom and high security in the right context.

Higher Education: Cornell University

Cornell University is one of the world's preeminent research institutions. Its online library, ArXiv.org, serves 30,000 people and receives almost 1.5 million hits a day from all around the world, with 14 million papers downloaded per month.

But ArXiv.org, a monolithic legacy application, needed some attention. Enhancements were taking too long to develop and move to production. Cornell took a look at what a microservices-based solution could do and was transfixed by how transforming the ArXiv application to microservices would accelerate time to new value.

Cornell made it a priority to modernize this educational and research tool and undertook a project to migrate ArXiv to Docker Datacenter, with incredible results:

- **Portability of applications across the application life cycle:** Cornell benefits from the streamlined workflow and the ability to track changes.
- **A central location for hosting Docker images and enables multiple organizations secure access to the images:** This increases productivity for developers.
- **High availability set up with secure registry replicas to ensure continuous availability:** The secure image storage allows the university to comply with industry standards and reduces risk.

Cornell is now deploying applications 13 times faster by leveraging the reusable architecture patterns of Docker Datacenter and simplified build and deployment processes. Equally important is the manageability of microservices. Docker Datacenter provides a platform that is resilient, reliable, and available.

Media: BBC

The BBC, delivers the latest news by posting more than 80,000 items online daily, and that's just in English! The technology enabling the distribution of news was made up of roughly 150 servers focusing on JVM and PHP languages that 50 developers use to push updates every couple of weeks.

It was contentious with ten different continuous integration (CI) environments running over 26,000 jobs that the more than 500 developers across all of the BBC had to share. The locked-down nature of the CI environments was making it difficult to make changes. Jobs could take up to 30 minutes just to schedule and could then require *another* 30 minutes to run. If the job failed, then they were back to the drawing board with another hour of lead time. The jobs were run sequentially, so scheduling issues and failures were common, leading to a lot of wasted developer time.

To circumvent another issue, lack of certain tools in the CI environment, the developers at the BBC News started a process called "Sideloading" to create packages with languages and versions not in the CI environment. Although this allowed them to build and test with the right language stack, it could add a few days to the process. In addition, other teams were now using these Sideloading packages putting a maintenance burden on the BBC News team.

The BBC's technology vision aimed to empower its developers with more control over their environments while eliminating restrictions on the applications. This transformation included a migration to the cloud, building an internal continuous development (CD) tooling solution, consolidating CI environments, and adopting Docker for distributed applications, allowing businesses to build, ship, and run applications anywhere.

As part of this transformation, BBC News started over with a new Docker-based CI environment consisting of a single master

solution that could have as many support nodes as needed. BBC News eliminated the 30-minute wait time for scheduling and can now run multiple jobs in parallel. Having official container images eliminated the time-consuming Sideloading process as well, and made maintaining language stacks a nonissue.

The overall results have been impressive. Here are just a few outcomes:

- **Reduction in job time:** Job time was slashed from one hour to ten minutes. No more waiting to schedule jobs. Jobs could run in parallel, and each job ran over 60 percent faster.
- **Language flexibility:** Containers isolate language stacks and eliminate the need for them while making it easy to use any language or version for a CI job.
- **Elimination of the multiday Sideloading process:** Containers helped to open the CI environment without adding risk.
- **Empowering developers:** They were able to control their application architecture and use the right language and version for their application.
- **Standardization:** The new CI environment is open, reproducible, scalable, highly available, and is the best practice that is being adopted across the rest of the BBC.

Mortgage: SA Home Loans

Prior to using Docker, SA Home Loans faced challenges in development, as well as in production. SA Home Loans currently has four scrum teams, each with a development and a system test lab. The team faced slow deployment times and was only able to build and deploy two apps in the dev labs, causing long deployment cycles and sometimes taking up to two weeks to get applications over to the testing environment.

These issues extended to production as well. The company's main home loan servicing software is built on C# and .NET and is a mixture of monolithic Windows services and IIS applications. In the past, when SA Home Loans deployed new features or fixes, it didn't have an easy or reliable rollback plan if something were to go wrong.

The company made the decision to adopt a microservices architecture and needed a solution that would enable its move to microservices while also enabling the agility, portability, and control it needed to build, test, and then deploy applications.

When it saw the benefits of Docker, SA Home Loans realized that it could move all its main application services over to containers. The company decided on Docker as its preferred platform, because containerizing its .NET services would allow them to be immutable and easily transferable across development and the deployment pipeline. For a production-ready orchestration service that could give the company a single point from which to manage and distribute containers, the company selected Docker Swarm because of its ease of use and the fact that it is Docker native.

Enterprise IT Partner: HPE IT

As a part of this book, HPE IT was interviewed about its experience deploying Docker Enterprise Edition and impacting the company's development efforts to extend its support for Docker. Edward Raigosa, HPE's Docker "Brew Master," says,

> As a part of our container journey on composable infrastructure, we invested in things like the composable infrastructure API that was exposed to OneView. So we learned about OneView and we found that if we use OneView, we can remove things such as the hypervisor from our configuration, and run Docker native platforms on our bare-metal infrastructure. We can get virtual machine–like experiences for provisioning, just like we do in our other environments.

The clear take-away here is that HPE is taking its own medicine and learning along the way. And, as the company learns, it's integrating that knowledge into its products to make it more consumable by its customers. That's the kind of behavior you often see from startups, not multi-billion-dollar multinational tech firms, and it's a very welcome trait. Via HPE Pointnext, HPE is taking all these learned lessons to its customers as well, and helping to enable them on their own Docker journeys.

IN THIS CHAPTER

- Discovering how HPE's product lineup can meet the needs of any size Docker initiative
- Finding out how composable infrastructure and Synergy work
- Getting insight into Docker deployment on HPE with three planning best practices

Chapter 8
HPE's End-to-End Docker Solution

Your workload operating environment is too important to leave to chance. As you undertake your container's journey and as you continue to support your full portfolio of applications, you need a partner that can help keep you on the right path and that has your back. HPE has you covered, no matter what you need. The HPE solution portfolio is composed of hardware, software, and services intended to help you to accelerate your Docker journey.

Hardware Options Abound

HPE provides a full range of hardware to support your container journey, from single ProLiant servers to HPE SimpliVity–based hyperconverged solutions to the Synergy platform. But your data center is more than just servers. You also need storage and networking equipment to make a complete solution.

Traditional servers

There will always be a need for traditional infrastructure approaches in which organizations buy servers, storage, and networking solutions and build their own data center environments. HPE sells the world's most popular line of servers. The ProLiant server lineup can easily support your Docker needs.

Hyperconverged solutions

Earlier in this book, you learned that containers can run on bare-metal servers, in virtual machines, or even in other containers. With a hyperconverged infrastructure as a container foundation, you'll be running containers in a virtualized environment. In hyperconvergence, all workloads run virtualized. That is simply the nature of the platform.

There are three characteristics that define hyperconverged infrastructure:

- Fluid pools/flexible ratio of compute and storage
- Native software defined storage/virtual storage appliance
- High-density internal storage

Hyperconverged infrastructure can be an ideal target for your container initiative if you need something a bit easier to manage than traditional infrastructure but don't want to make the jump to composable infrastructure. In the hyperconverged infrastructure area, HPE makes available a built-for-enterprise hyperconverged infrastructure solution based on SimpliVity OmniStack software. The HPE SimpliVity 380 with OmniStack provides customers with a container host that provides excellent data efficiency via deduplication and compression, built-in backup and disaster recovery capabilities, and simple, centralized management.

HPE Synergy Composable Infrastructure solutions

HPE Synergy is the first solution to support composable infrastructure. The HPE Synergy product line delivers on a series of key architectural principles:

- **Fluid resource pools:** With fluid pools of compute, storage, and fabric, resources can be instantly turned on and flexed to meet the needs of any business application. You aren't forced to add resources that aren't required in order to get resources that you do need.
- **Software-defined intelligence:** HPE Synergy bakes software-defined intelligence right into the infrastructure and uses workload templates to tell the infrastructure how to compose, recompose, and update on the fly in a very repeatable, frictionless manner. The various infrastructure elements simply become an extension of the software layer.
- **Unified application programming interface:** With these tools, the infrastructure can be programmed like code, bringing powerful Infrastructure as a Service (IaaS) capabilities to the private data center.

Figure 8-1 shows what makes composable infrastructure tick.

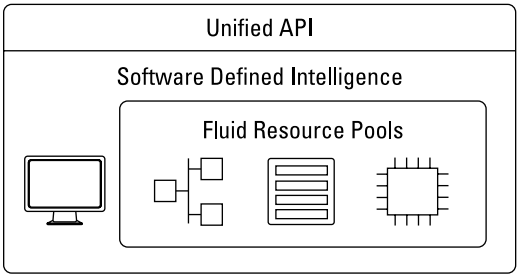

FIGURE 8-1: Composable infrastructure architecture.

Synergy

As the first full-range composable infrastructure solution on the market, HPE Synergy is a line of hardware and software that can become the single infrastructure for all your container needs. With powerful availability, expansion, and management tools, Synergy brings cloud-like infrastructure to the private data center. I talk a bit more about Synergy later in this chapter.

Storage and networking

On the storage front, unlike their stateless counterparts, certain application containers require data to persist beyond the life cycle of that specific container. Through partnerships with Docker and

Mesosphere, HPE has developed the Docker volume plugin, which allows containerized applications to store data on HPE Storage platforms such as 3PAR StoreServ and StoreVirtual.

The plugin enables dynamically provisioned, scheduled, and orchestrated storage volumes from Docker Swarm and Mesosphere Marathon. The storage volumes on 3PAR and StoreVirtual simply follow the containers wherever they're scheduled to run by the Orchestrator, allowing stateful containers to have the same mobility benefits as their stateless brethren. In addition, you can apply granular volume features and attributes such as size of the storage volume, volume type (thin or thick), and Adaptive Flash Cache, providing ultimate flexibility to customize storage options with finer granularity at the container level.

Through the integration of this plugin, you can have a single storage platform for traditional as well as modern container cloud-native applications.

And lest you think we forgot about the network, think again! Inside every software-defined data center lies software-defined networking. For Docker solutions, HPE has partnered with Aruba Networks as a preferred solution for building high-performance networks that can support even the largest container environments. HPE and Arista offer a range of easily managed networking solutions designed to address the growth of big data, storage, cloud, and increasingly dense virtualized and containerized environments.

To take full advantage of containers, the networking environment must be optimized to connect container-based applications. HPE's server and network hardware agnostic Distributed Cloud Networking (DCN) can be implemented as a non-disruptive overlay for all virtualized and physical server resources. With high flexibility, allowing it to be deployed in any Docker container, hypervisor, or bare-metal environment, HPE DCN has been field-tested to integrate fully with Docker.

Software: HPE OneView

Hardware is just one part of the story. For a complete experience, you also need software. HPE OneView is a management tool that automates the delivery and operation of server, storage,

and network resources in physical, virtual, and container-based environments and that supports the full HPE product portfolio. By converging management of HPE server, storage, and networking resources, HPE OneView improves IT administrator efficiency. Through automation, HPE OneView also helps to prevent downtime caused by human error. It interoperates closely with HPE CloudSystem, HPE Business Service Management, VMware vCenter, and Microsoft System Center, as well as Chef, Docker, and OpenStack.

Services: HPE Pointnext

Sometimes, DIY becomes "DI why?" as in, "Why did I do this?" There are a whole lot of moving parts in a workload operating environment, and Docker is no exception, particularly given the kind of cultural and technical changes that it can introduce. Sometimes, you need a trusted advisor to help guide you through the pitfalls so you know how to avoid the traps and end with the best possible results.

HPE Pointnext is the HPE consulting arm, and they're there to help you get the most from your HPE and Docker deployment and exist to provide counsel and services to that end.

Some Best Practices Advice

Now that you have a complete picture of what HPE offers and how Docker fits into the mix, how can you turn this knowledge into action? Well, let's start with three big items that may change how you think about infrastructure deployment.

Plan for higher density

Docker containers, from a resource and performance perspective, are essentially just standard Linux processes that have been wrapped by lightweight Linux kernel features that have little to no effect on performance overhead. When you plan for workloads in containers, plan for higher density of these workloads on the host OS (virtual or bare-metal) than you may have with virtual machines. You'll gain utilization benefits from higher density of

resources like storage and memory due to no longer needing the OS overhead of multiple virtual machines.

Of course, Synergy can support you nicely here. You can deploy as many bare-metal systems as you like in your Synergy environment.

Tune to optimize


TIP

Depending on the application, some tuning may be required to achieve optimal performance with virtual machines or containers. For example, HPE's tests show that for CPU-bound workloads, performance can vary widely based on VM and container CPU affinity settings. If you're using VMs (with or without containers) and have workloads that are largely CPU bound, you should perform some benchmarking with and without CPU affinity set in your environment to gain a clearer understanding of its potential benefits. Managing CPU affinity settings in virtualization can add complexity to infrastructure management, but rarely does it negatively affect performance — often, it improves it.

If this sounds complicated to you, never fear! HPE has your back and can help you with this analysis and help you to deploy a perfectly tuned Docker infrastructure.

Go bare-metal when you're ready or use big virtual machines

Either way, HPE Synergy has you covered because it supports both bare metal and virtual deployment models, and containers can run on either. In testing, HPE's analysis shows that the bare-metal Docker deployments yielded performance and capacity improvements beyond all other scenarios. If it fits your infrastructure management model to choose bare-metal hosts for your containers, you'll get higher density of workloads with improved overall performance than you would with virtual machines. For example, HPE was able to consolidate a single host with eight MySQL virtual machine workloads to eight containers running on an identical bare-metal host and achieve a 73 percent performance improvement.

TIP

The combination of Docker-ready HPE servers and infrastructure provisioning and management tools, such as HPE OneView, provides the option for a reliable, fast, and repeatable solution for deploying bare-metal Docker hosts that are supported jointly by Docker and HPE. Consider utilizing a hybrid approach toward Docker hosts as described in the HPE Reference Configuration for Docker Datacenter on Bare Metal with Persistent Docker Volumes, because this offers the flexibility to only scale up the bare-metal HPE ProLiant DL360 Docker UCP work nodes when you have workloads ready and suited for bare-metal deployments. Even if moving to bare-metal containers is not something you're able to take advantage of today, you can still gain performance and resource utilization improvements by consolidating your workloads to fewer, larger virtual machines with a higher count of container workloads in each. This option is typically better than replacing the "one workload per VM" with a "one container per VM" model.

Chapter 9
Ten Reasons to Go with Docker and HPE

There are ten key reasons that you should be considering HPE and Docker as your trusted partners in your container journey.

A 360-Degree Partnership

The relationship between Docker and HPE is mutually beneficial and deep. Providing strategic, tactical, and supportive guidance, HPE and Docker enable organizations' digital transformation efforts.

With best-in-class hardware and software from HPE and with market-leading container capabilities from Docker, the HPE/Docker combination is a potent one and enables world-class datacenter care.

A Recipe for Success

Customers typically take a multiphase approach to container adoption (see Figure 9-1). HPE has the personnel and offerings required to guide and accelerate the customer's journey through these phases:

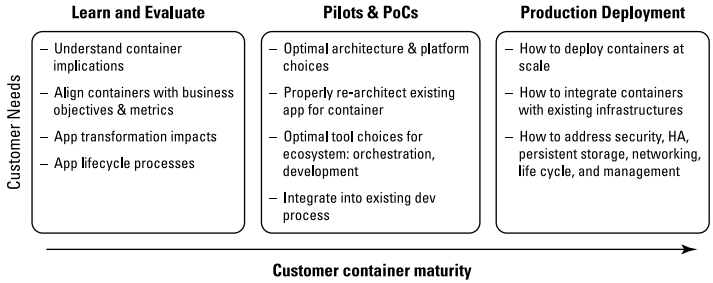

FIGURE 9-1: A multiphase approach to container adoption.

» Learn and Evaluate

- HPE Pointnext and its partners offer various assessment and architecture planning services that help educate customers on container technology, identify projects that are suitable for containerization, and provide a design blueprint.
- IT has adopted Docker technology for its DevOps process and gone through a successful containerization transformation using HPE's container solution. Many sidebars throughout this book include excerpts of interviews I conducted with internal HPE IT Docker experts. HPE IT's experience will be published in detailed case studies to assist other customers going through the same journey.

» Pilot and POC

- HPE Pointnext and its partners offer a series of development accelerator services that can assist customers with architecture planning all the way to design implementation.
- HPE provides various container and DevOps reference architectures and blueprints to help ease and provide best practice guidance for infrastructure planning during the design phase.

- HPE partners, such as MicroFocus (HPE software merger) offers a broad set of tools that help customer design, debug, and test container applications.
- HPE hyperconverged and ProLiant platforms with its ease of use and superior scalability are great choices to host container DevOps projects.

» Production Deployment

- HPE offers a one-stop shop for customers looking to procure Docker licenses and enterprise-level support for its Docker production environment. HPE provides a "single throat to choke" for the Docker stack with level-one and -two support at a worldwide level.
- HPE offers a broad choice of platforms to run all types of container use cases ranging from ProLiant for CaaS, Hyperconverged for DevOps, and Synergy for container and mix workload deployment at scale.
- HPE is working closely with Docker to accelerate the deployment of Docker on HPE platforms using OneView integration and Synergy Image Streamer technology. Customers will be able to automate and quickly spin up their Docker clusters on HPE in minutes.
- HPE offers a native Docker volume plug-in for HPE 3PAR StoreServ arrays and StoreVirtual Storage. This plug-in is fully supported by all Docker tools and enables the use of unified, flash-optimized HPE 3PAR StoreServ arrays or StoreVirtual software-defined storage as the underlying storage for bare-metal container deployments using Docker.
- HPE Distributed Cloud Networking (DCN) overcomes manual IT provisioning by automating physical and virtual multitenant network infrastructure provisioning with application and cloud orchestration.
- HPE FlexCapacity service for containers offers customers a "pay as you grow" resource consumption model to help customers optimize their CAPEX spend and provide standby on-demand compute and storage resources for their Docker clusters.

A Single Throat to Choke

Due to the complex interactions between data center components, it's difficult to figure out who to call when something goes wrong. By moving to a sole source for all data center hardware and software, you get a single phone number to call when you need help. That can be a powerful tool in your recovery arsenal.

The Strategic Alliance between the Companies

The HPE and Docker Alliance is a comprehensive solution that spans the entire application life cycle and helps you cut costs, reduce complexity, and increase performance for even the most demanding enterprise workloads.

Joint Research and Development

HPE and Docker are joined at the hip when it comes to being at the forefront of the container revolution. The companies are working together to ensure that the full spectrum of data center offerings acts as one, to help organizations simplify IT, reduce costs, and drive business forward.

HPE Hardware Solutions Are Optimized to Run Docker

HPE and Docker are working together to make sure that HPE server, storage, networking equipment, and management tools provide the best possible performance, the highest level of reliability, and optimum value for their joint customers.

HPE's Experience

Many sidebars in this book are excerpts of interviews I conducted with internal HPE IT Docker experts. These experts are at the forefront of the container revolution, and they're the brain trust behind HPE's own efforts to revolutionize their environment and adopt a modern architecture that can scale as the business grows.

Docker Is Leading the Revolution

Docker is far and away the leading provider of container tools and services, serving tens of thousands of enterprises large and small. Docker is driving container standardization across the industry.

Ready-to-Go Blueprints

Kick-start your container project with HPE's container and DevOps reference architectures and blueprints.

Hybrid Cloud Expertise

HPE Distributed Cloud Networking (DCN) overcomes manual IT provisioning by automating physical and virtual multitenant network infrastructure provisioning with application and cloud orchestration.

Embark on the next era of application infrastructure!

Enterprises are on a journey toward the idea economy. This pursuit of a world where enterprise IT practitioners can deliver applications quickly, across both incumbent and modern infrastructure, while reducing overall costs to the organization, has led to the popularity of containers. Docker, the enterprise container platform running on HPE's infrastructure, delivers the combination enterprises require to make the idea economy a reality.

Inside…

- Discover why containers are so popular
- Learn how containers compare to virtualization
- Uncover popular container use cases
- See how enterprise teams deliver app faster
- Use containers to scale your business

Scott D. Lowe has been in the IT field since 1994. After spending time in the trenches, Scott spent ten years as a CIO. Today, he is a partner in ActualTech Media, as well as a consultant providing insight and solutions to his higher-ed. clients.

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

for dummies®
A Wiley Brand

ISBN: 978-1-119-41611-1

WILEY END USER LICENSE AGREEMENT

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