THENEWSTACK

THE DOCKER & CONTAINER ECOSYSTEM

EDITED & CURATED BY ALEX WILLIAMS

The New Stack:

The Docker and Container Ecosystem eBook Series

Alex Williams, Founder & Editor-in-Chief

Benjamin Ball, Technical Editor & Producer

Hoang Dinh, Creative Director

Sam Charrington, Editor, Founder & Principal Analyst of CloudPulse Strategies

Contributors:

Atul Jha, Research

Brett Heckman, eBook Technical Consultant

Joab Jackson, Editor

Judy Williams, Copy Editor

Klint Finley, Editor

Lawrence Hecht, Data Research Director

Patricia Dugan, Director of Community Marketing & Development

TABLE OF CONTENTS

INTRODUCTION

We never thought that our last six months at The New Stack would be spent defining an ebook series about Docker and the container ecosystem. It was supposed to be one ebook that we'd do in six weeks or so, but then we started putting it together. It had quite a scope that easily would have made just one ebook more than 100 pages.

Still, even if we could fit the subject matter into one ebook, would it give justice to the subject matter? The answer was no. It made far more sense to make it a series, and take the time to explore how containers apply to the entire stack, as individual units that have quickly come to be associated with orchestration.

It's a new time that is really not about IT. It's now about application development and management at scale. These are the days that will help define an ever-changing technology stack that is far more ephemeral than a technologist might have dreamed in the enterprise heyday. Today, we are talking about applications far more than the machines they run on. The machines are now a resource — not a server farm, a grid or even a cloud. This resource is as real as any physical resource we know of.

But how do we connect the resources? How do these resources become deeper, wider and more powerful in what they provide? How do we make them simple enough so that we don't need to invent something new every time we want to explore the deeper realms of what these infinite data dimensions offer?

In many respects, it's simply a matter of economics that we see in the workings of this shift in how we view technology stacks. Containers have a

cost efficiency, portability and convenience factor that gives them credence as a way to build apps directly from the developer's laptop with much of the process automated and packaged. They have impacts on the cost of managing resources.

These economic considerations and impacts on behavior speak to why Docker is having such a lasting symbolic impact on changing technologies and models.

It's this change to more container-based workloads that will drive the substance of our ebook series. We have a lot of subject matter to cover. We'll run the series into the spring of 2016, and even at that point the landscape will have changed further. At that time, we may even have a view beyond containers and into the infinite continuum that makes this technology age feel so timeless.

I am so lucky to be part of such a great community. Our goal is to analyze how application development and management at scale is changing as the new stack evolves. Every day I get to explore the workings of this world and its dimensions. I get to talk to the smartest technologists in the world. We also have the honor and the privilege of working with fantastic sponsors. In particular, I want to thank our series sponsors: Cisco, Docker and IBM. We could not be here without them.

Finally, there is the team who is working to build The New Stack who I get to talk to every day. I love this group of people. They are the heart and soul of what we do every day at The New Stack. They are my colleagues and are becoming my friends. They are the ones who really define The New Stack and bring with them the excellence we always strive to maintain. I would not be here without them.

Thank you so much for your interest in our ebook series. Please reach out any time. I am always happy to meet and talk with people who care enough to introduce themselves.

Thanks, Alex.

Alex Williams
Founder and Editor-in-Chief
The New Stack

SPONSORS

We are grateful for the support of the following series sponsors:
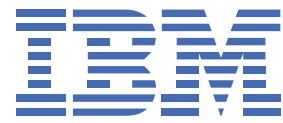
CROSSING THE OCEAN WITH CONTAINERS

by **JEFF SUSSNA**
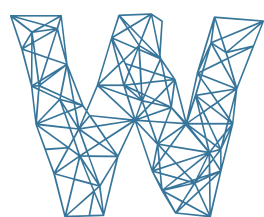
When cloud computing first made its appearance, most people viewed it as a cost-reduction convenience. Soon, though, many organizations began to recognize its power to transform IT on a deeper level. Cloud offered a vision of infrastructure as a dynamic, adaptable resource that IT could use to power 21st-century business imperatives for agility and responsiveness. Terms such as "cloud-native" and "cattle not pets" expressed the understanding that cloud-based IT required a fundamental mindset shift, away from treating infrastructure components as large, expensive, specialized, handcrafted, and slow-to-change.

> **"** Docker has captured the industry's imagination with breathtaking speed.

Containers are taking this transformation to the next level. Docker has captured the industry's imagination with breathtaking speed. It began in

similar fashion to cloud, seeming to provide a more convenient solution to existing packaging and deployment problems. In reality, though, containers point the way towards an even more profound mindset shift than cloud.

While cloud computing changed how we manage "machines," it didn't change the basic things we managed. Containers, on the other hand, promise a world that transcends our attachment to traditional servers and operating systems altogether. They truly shift the emphasis to applications and application components. One might claim that containers, in combination with the microservices software pattern, represent the fruition of the object-oriented, component-based vision for application architecture.

In a testament to the rapidity of Docker's ascent, the conversation has quickly shifted to its readiness for production enterprise use. Blog posts chronicling experiences running Docker in production duel with others detailing the ways in which it's not yet viable. This binary argument misses

the nature of technology adoption. The fact that a craft has proven itself seaworthy doesn't obviate the need to figure out how to navigate the ocean with it. Just as was the case with cloud computing, containers pose as many questions as they answer. These questions arise on multiple levels: architectural, operational, organizational, and conceptual.

Containers make many things possible, without necessarily accomplishing any of them by themselves. Almost immediately after the excitement of recognizing the power of containers, one begins the more laborious process of figuring out how to use them for practical purposes. Immediate issues include questions such as:

- How do containers communicate across operating system and network boundaries?

- What's the best way to configure them and manage their lifecycles?

- How do you monitor them?

- How do you actually compose them into larger systems, and how do you manage those composite systems?

Various answers to these questions have begun to emerge. Packaging tools such as Packer bridge configuration automation with immutable infrastructure. Cluster management systems such as Kubernetes layer replication, health maintenance, and network management on top of raw containers. Platform-as-a-Service offerings such as Cloud Foundry and OpenShift are embracing containers within their own architectural models.

These higher-order systems answer some of the initial questions that arise while trying to deploy containers. They also, though, raise new questions

% Of Organizations Planning To Address Needs in the Next 2 Years

(excludes partnerships)

Growth Potential

Category	by improving existing product	by developing new product	Total
Scheduler / Orchestration / Mgnt / Monitoring	60%	11%	**71%**
App Development / Deployment	60%	11%	**71%**
Developer Tools	43%	28%	**71%**
Security	40%	23%	**63%**
Networking	30%	23%	**53%**
Config Mgnt Tool w/Docker Support	28%	23%	**51%**
Container/OS	36%		**38%**
Image Registry	21%	13%	**34%**
Hosting/Service Providers	26%	4%	**30%**
Database/Big Data	21%	6%	**27%**

Legend: ■ by improving existing product ■ by developing new product (internally or via M&A)

FIG 1: *Our survey results reflect how the community is developing services to make containers more practical for users. Most see offering deeper functionality for developer tools, security and networking.*

of their own. Now, instead of asking how to manage and compose containers, one has to ask how to manage and compose the container management, deployment, and operations toolchain.

This process is a recursive one. At the moment, we can't know where it will end. What does it mean, for example, to run Kubernetes on top of Mesos? Contemplating that question involves understanding and interrelating no less than three unfamiliar technologies and operating models.

More importantly, though, organizations are just beginning to contemplate how to integrate the container model into their enterprise architectures, organizations, and conceptual frameworks. This process will be a journey

of its own. It will consist of a combination of adaptation and transformation. The precise path and destination of that journey are both unknown, and will depend to a large degree on each organization's individual history, capabilities, and style.

Deep technical change is a complex process. It can't be predicted or linearly planned. Implementing it requires the same lean and agile techniques we use for product development. The question, "is Docker ready for the enterprise?" is the wrong question. A better question would be, "how are containers likely to perturb our organization and our ways of doing things?" Answering that question requires conducting experiments and learning from feedback. It also goes far beyond purely technical concerns.

> **" Deep technical change is a complex process. It can't be predicted or linearly planned.**

Adopting a transformative technology such as cloud or containers impacts every aspect of IT. When computing resources pop into and out of existence by the minute instead of the year, and in the hundreds of thousands instead of the hundreds, traditional management methods no longer suffice. Both the configuration management system and the configuration and change management processes need retooling.

New tools and processes, however, are necessary but not sufficient. Technical staff don't just need retraining to use the new tools; they also need to learn new ways of thinking about what systems are and how to solve problems with them. Making Docker enterprise-ready involves not

just making it technically robust and secure, but also figuring out what it implies for staffing, hiring, and training. The constraint that gates a company's ability to absorb change often isn't a new technology itself, but rather the ability to hire people who can comprehend the implications of that technology, and who can operate it based on that understanding.

> **" We need to apply everything we've learned about navigating change and uncertainty, and step beyond the binary success/failure conceptual model of adoption.**

Ultimately, the impact of containers will reach even beyond IT, and play a part in transforming the entire nature of the enterprise. The value of microservices and containers lies in how they enable smaller, faster, more frequent change. In order to take full advantage of this capability, IT organizations will need to restructure themselves socially as well as architecturally. This cascading transformation process will in turn apply to the enterprise as a whole, as it strives to take advantage of its new capabilities for responsive digital service.

Just as container management systems present new sets of questions, so too do new organizational structures. If a company decides to adopt Holacracy as part of its mission to improve agility, it will have to navigate that adoption process. Just as with technological change, effective social and structural change happens through experimentation, failure, and adaptation.

In thinking about enterprise adoption of Docker or any other container technology, we need to understand it for what it is: a trigger for a much larger, more complex, and long-lasting process. We need to cast our gaze beyond containers themselves, towards the socio-technical systems they are just beginning to perturb. We need to apply everything we've learned about navigating change and uncertainty, and step beyond the binary success/fa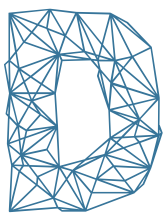ilure conceptual model of adoption. In this way, containers are no different from DevOps, or Lean, or any other organizational transformation.

THE WORLD IS PROGRAMMABLE WITH CONTAINERS

by **ALEX WILLIAMS**

ocker and container technologies symbolize a new economic reality that puts the developer at the center of the transformation from big machines to application-driven systems. The shift from heavyweight to lightweight technologies, and from human to automated systems, is apparent in the ecosystem in a number of ways:

- The Internet is being programmed, and it needs plumbing to work.

- Application development is faster than ever.

- Open source communities are proliferating and becoming more commercial.

- Programming languages are making it easier to build software.

- The need is coming for automated infrastructure and scaled-out distributed resources.

- It will increasingly be more about performance than compatability.

Container technologies have a long history. Docker is simply a new iteration that makes it easier and more convenient to design, deploy and manage applications. Containers are processes, parts of systems that are now mutating into different forms. There are new types of containers, platforms, open source projects, orchestration systems, service discovery tools, schedulers and a shift in market influence.

The Docker and containers shift is forcing companies to rethink how platforms and orchestration services can manage new, lighter workloads. This indicates a change from virtualized infrastructures to container-centric, distributed resources that abstract away the complexities that have historically come with developing apps on cloud services and hosted environments.

Docker operates on top of the infrastructure and syncs with the developer's laptop. Docker technologists will often refer to Docker as a way to ship, build, run and deploy applications. It's an open platform for distributed apps. It works wherever Linux does, which is essentially anywhere; it also works on Windows. Docker is not reliant on a separate operating system; it just takes advantage of already-built technology.

Docker is the work of [Solomon Hykes](#), who founded [dotCloud](#), a platform as a service (PaaS) company. Hykes built Docker as an API that isolates processes. It uses isolation technologies, such as cgroups and namespaces, that allow the containers to run independently on the Linux kernel without the overhead of starting up a virtual machine. It allows Docker containers to run independently, making it easy to move code. Virtualization technology from companies like VMware sits below the operating system and virtualizes the server, not the application. Wherever the virtual machine goes, the operating system has to go with it. It has to

be taken down, then booted back up and configured to run with the database and the rest of the stack that it depends on.

Virtualization is not independent of container technology. VMware, for example, has developed a platform that uses virtual machines to insulate containers. Photon OS, as it's now called, will serve as the agent that gives VMware's vSphere management system visibility into the operations inside containers.

It means containers that include Photon OS will be somewhat different from containers that don't include it. It is an alternative platform to vSphere. This new Photon Platform, as VMware has dubbed it, is intended for "cloud-native" containers only — for data centers intending to deliver software as a service (SaaS) where vSphere is not already established, nor intends to be established.

The premise of an application-centric infrastructure speaks to a shift that is less about the machines and more about the sophisticated software and services that make the world run.

It's this sophisticated infrastructure that makes it possible for startups to build services faster and cheaper. That's what makes the new stack significant in so many ways. It allows companies to be far more agile than others using heavyweight technologies that rely on proprietary software and high IT overhead.

The market is now witnessing a change that affects the companies that have historically built technologies that were designed for desktops and data centers.

The New Efficiencies of Immutable Infrastructure

Market Reality: There are billions of people in the world and almost everyone has had some contact with the Internet, even if they may not realize it. There are millions of developers who are building the new foundations for how we live and work. In the meantime, their operations counterparts are doing the plumbing to make the Internet more programmable.

The Result: The arrival of software and systems that speak to efficiency, convenience and performance over compatability.

Patrick Chanezon is a member of the technical staff at Docker. He presented at DockerCon in June 2015 and made the argument that millions of programmers means new innovations. It is these innovations that will change almost everything we know about software today.

"Container technologies have become a software layer to program the Internet" is, in essence, the argument Docker makes. Its technology is a software layer. It's not a container technology play; it's a play to be the software platform that programs the Internet for the millions of programmers building services for a world that has an infinite number of programmable nodes. According to this view, anything can be a node. Almost anything can become a digital object that can be programmed.

Is it far-fetched to think that containers will be the layer that makes the world programmable? It's more realistic to think of containers as part of a continuum, which is evident by the development of the current market. Serverless architectures are gaining favor as a way to abstract the

complexities of distributed systems. Unikernels are gaining favor for being far more lightweight than container technologies.

Other companies in the container ecosystem are declaring their own ways to define this evolving continuum. Amazon Web Services is in a strong position with a new registry platform that integrates with its EC2 Container Service. This platform is joining a strong lineup of registry and runtime services including IBM Containers on Bluemix, CoreOS Enterprise Registry, JFrog's Artifactory, Google Container Registry, Quay.io and, of course, Docker Trusted Registry.

Despite tremendous demand from people using container technologies, infrastructure has not been transformed. Security policies, load balancing,

FIG 1: *Docker sees container technologies as being a programmable software layer that sits on top of the physical Internet*

The Programmable Internet

Programmers

App App App App App App App App App App App App App

Internet Software (Containers) Layer

Internet Hardware Layer

Servers   Desktops   Phones   Cars   Houses   Public Transit   TVs   Industrial Facilities   Scientific Instruments   Financial System
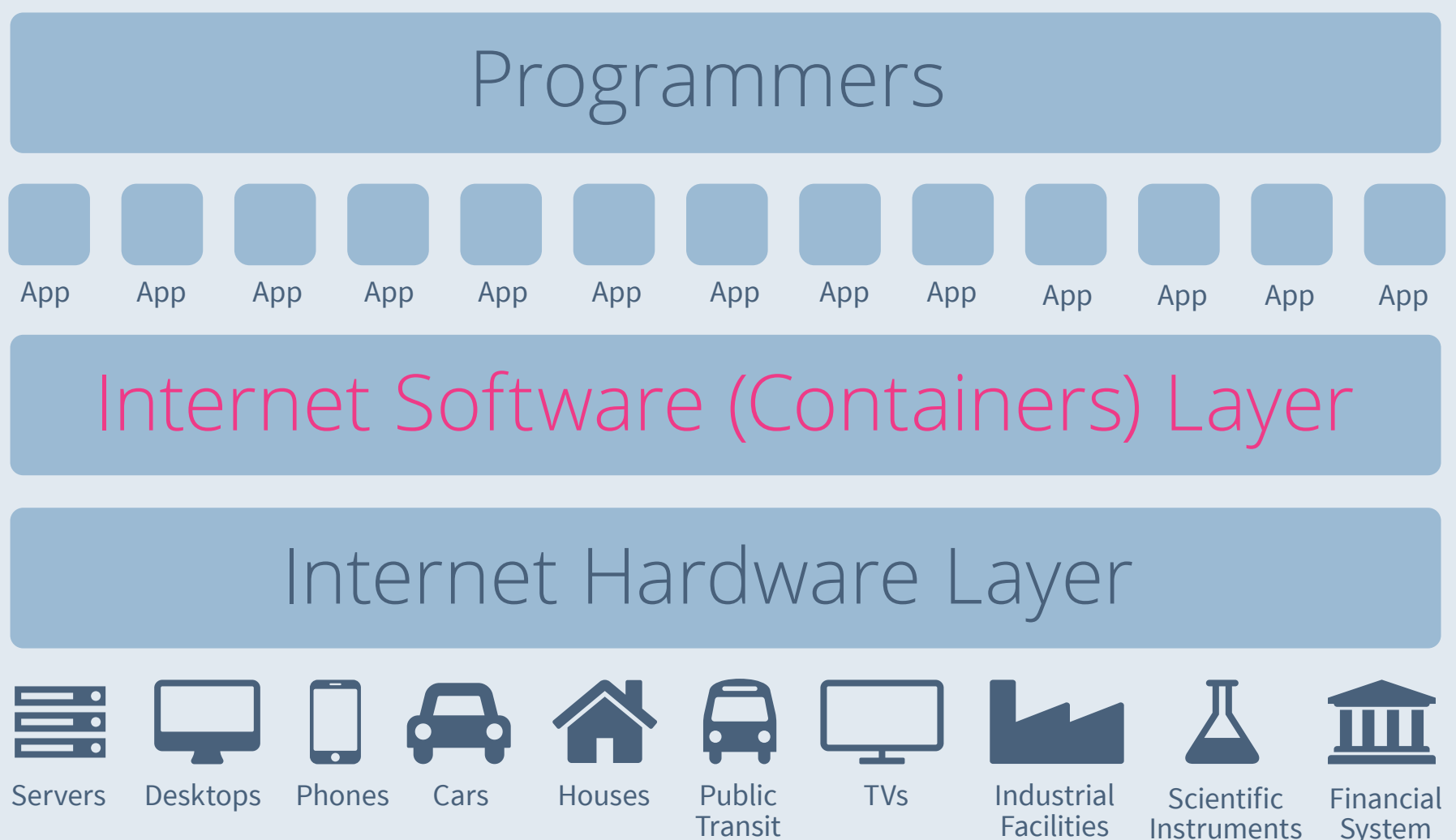
storage management, service discovery, service management, resource management and native container support are largely missing or still inadequate for production workloads.

Virtual machine bloat, large attack surfaces, legacy executables and base-OS fragmentation are a common problem, as pointed out by Darren Rush in a look at a post-container world.

The need is for immutable infrastructure. That means creating something and then leaving it unchanged. Don't update it, just create something new. Once the image is working, only a working image is deployed. The old version of the image can be kept in a container if, for example, there needs to be a rollback of the environment. An entire infrastructure can be timestamped, making it far easier to scale-out horizontally — not just from a faster deployment, but by actually adding more machines to make processing faster.

This new generation of immutable infrastructure reflects how mutable environments have become difficult to manage. System administrators managing servers need to have logins and accounts. They have to manage software with mutable updates that can succeed or fail. These are technologies that have various states of repair or disrepair. Setting up immutable servers that are configured to work once and are deployed as is, removes many of these types of issues. It removes the burden of manual updates. Let the machines take control.

How will this change happen? Adrian Cockcroft of Battery Ventures argues that DevOps is the outcome of this sort of transformation, and that essentially means a reorganization for most companies. But with a microservices approach, an immutable infrastructure can allow for steep

cost reductions and a high rate of change. Developers can build and deploy services in seconds: Docker packages them and the microservices environment runs them in what amounts to fast tooling that supports continuous delivery of many tiny changes.

These new microservices environments are not easy to manage. Think of the speed involved, the scale needed across continents, regions and zones — then you get a picture of how complex it can be. The flow looks more like a ball of tangled yarn than a traditional flow chart. Failure patterns need to be understood across zones.

The Container Combo

Docker and containers offer portability, speed, configuration and a hub, much like GitHub, according to Cockcroft again, who wrote on the topic for The New Stack.

The portability comes through Docker's packaging, which defines the packages of any Linux application or service, Cockcroft wrote. A package that is created and tested on a developer's laptop using any language or framework can run unmodified on any public cloud, any private cloud or a bare metal server.

This is a similar benefit to Java's "write once, run anywhere" idea, but is more robust and is generalized to "build anything once, run anywhere."

Then there is speed. A Docker container can be launched in a second, as opposed to a virtual machine which may take tens of seconds or even minutes. Configuration is not really a matter of concern as each update becomes a new version, or in other words, a new container.

It's this speed that is most transforming. Speed means a lower barrier for taking risks with trying new ways to speed up app development and management. However, we have barely come to understand what the outer dimensions of this new capability means to us all.

"You are going to see a new order of magnitude in terms of swarming of compute running for shorter time periods," said John Willis in a story from The New Stack earlier this year. "Now it is a matter of nanocomposite. It could go from 1,000 to one billion instances starting and stopping in a week."

The startup time for a container is around a second. Public cloud virtual machines (VMs) take from tens of seconds to several minutes, because they boot a full operating system every time, and booting a VM on a laptop can take minutes.

Docker also simplifies deployment. Docker Compose files manage services with higher level abstractions. A Compose file uses links, which are an abstraction above specific IPs. This makes deployment more generic and loosely coupled. Compose files specify a loose set of services and their connections, which is a lighter and more flexible abstraction.

Docker containers are shared in a public registry at Docker Hub. This is organized similarly to GitHub, and already contains tens of thousands of containers. Because containers are very portable, this provides a very useful cross platform for applications and component microservices that can be assembled into applications. Other attempts to build "app stores" are tied to a specific platform (e.g., the AWS Marketplace or Ubuntu's Juju Charms) or tool (e.g., the Chef Supermarket), and it seems likely that Docker Hub will end up as a far bigger source of off-the-shelf software components and monetization opportunities.

The Evolution of Deployment Speed
Speed enables and encourages new microservice architectures

Datacenters	Virtualization	Containers	Serverless
Months to deploy, live for years	Minutes to deploy, live for weeks	Seconds to deploy, live for minutes/hours	Milliseconds to deploy, live for seconds
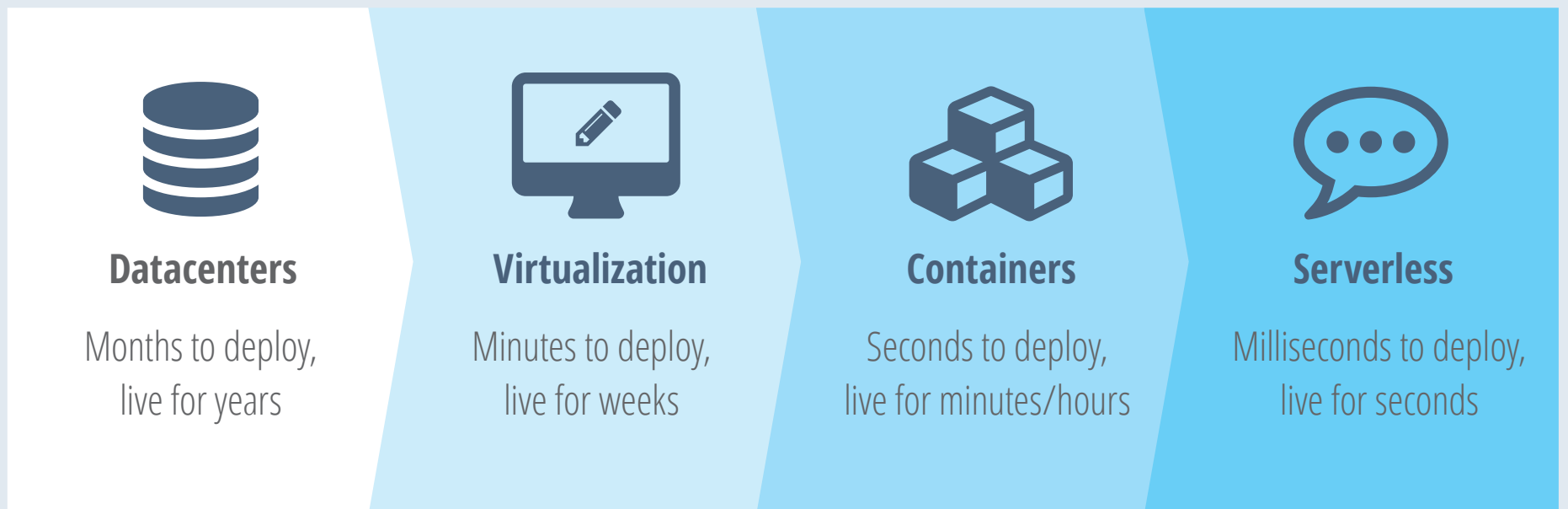
thenewstack.io

FIG 2: *Accelerating deployment speed means a lower barrier for taking risks with trying new ways to further speed up app development and management.*

Summary

In all, an application-centric approach has deep roots in the Linux ecosystem. There is a rich history of tooling that has allowed for a market of compatibility. Linux runs everywhere and everything runs on it. But these systems were not built for efficiency. There is a lot of code and a lot of complexity in the system, including permission checks on the operating system that stem back from a time when massive monolithic systems were built into single machines.

Today, performance is becoming a key-value driver for containers, but they still have an associated complexity. And that's why there is such a diverse ecosystem: it's needed for users to build architectures that can take containers from the laptop and into distributed environments — environments that can manage any number of microservices that are fast, efficient and running at the highest possible performance.

OPEN SOURCE COMMUNITIES DEFINE THE DOCKER AND CONTAINER ECOSYSTEM

by **LAWRENCE HECHT**
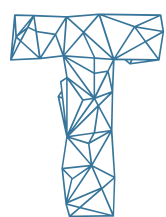
The economics of proprietary technologies are less viable with increasingly complex systems that require constant adaptation, changes and updates. Increasingly, proprietary software and systems are less robust than their open source equivalents. The Docker and container ecosystem is representative of this new market reality.

As many of the container-related projects move into enterprise production uses, a number of open source communities are being influenced by vendors such as IBM, Intel and Google, as well as by large customers such as Goldman Sachs, that are creating new open source foundations. These foundations reflect a new generation of commercial-style open source communities, lead by professional organizations such as the Linux Foundation, which now runs the Open Container Initiative, the Cloud Native Computing Foundation and the Cloud Foundry Foundation.

Created in June 2015, the Open Container Initiative (OCI) is an open specification and runtime for containers. OCI provides the end user with a

view into how the overall market is evolving, the technical differentiations of the providers and a context for looking at the past and future of an application-centric infrastructure.

The roots of OCI can be traced back to Docker and the development of its libcontainer technology. The libcontainer format enabled Docker to allow for different versions of Linux distros to be used, which was more difficult with LXC. In the Docker architecture, the containers running on a single host share the kernel of the running Linux OS. It undocked itself from systemd, the critical launch component of the Linux kernel. As LXC was designed, systemd was responsible for launching and maintaining container processes in a manner that the operating system could manage.

As part of OCI, Docker donated libcontainer to the initiative. The overall goal is to ensure compatibility between systems and the code that utilizes containers, thus freeing the next generation of engineers to focus on innovating higher up the value chain.

With that background, the lines defining the companies become better understood and are exemplified in the data analysis of OCI and the difference that it has with the still-emergent Cloud Native Computing Foundation (CNCF). The CNCF is the newest open source project, initiated by Google and joined by Cisco, Docker, IBM, Mesosphere, Joyent and a host of other companies in the ecosystem that are trying to standardize scheduling and orchestration capabilities.

By reviewing GitHub activity statistics, The New Stack found significant activity and industry cooperation within the tight-knit group creating the specifications. For the actual underlying runtime code (runC), which was

migrated from libcontainer, there has been robust participation from both independent developers and companies.

For background, CoreOS last Fall announced rkt, the specification developed by CoreOS for its Rocket runtime system. At CoreOSFest this past spring, the company announced App Container (appc), its own open source project based upon the rkt technology. Google, VMware, Red Hat, Hybrid Cloud OS maker Apcera and a gathering coalition of industry partners backed appc.

CoreOs has funding from Google Ventures. Furthermore, the CoreOS technology integrates deeply with Kubernetes, Google's open source container management platform. Google, for its part, is focusing on the recently announced Cloud Native Computing Foundation, which has an emphasis on container management.

Docker, without a doubt, competes with both CoreOS and Google. They also cooperate with each other, which reflects the nuances of this fast changing world of application development and management at scale, as there is really no one universal solution.

Specification and Who is Leading the Project

OCI's initial technical leadership included representatives from Docker, Red Hat, CoreOS, Google, and an independent developer with a company called InfoSiftr. Now that the project has been operating for a few months, we can see who is actually involved with the development of specifications, which is where the real power lies in this project. We found that there have been 24 contributors to the opencontainers/specs

repository, with the most involvement from Huawei, Docker, Red Hat, IBM, Fujitsu and CoreOS.

It is noteworthy that the previous top three contributors to appc's spec — CoreOS's Jonathan Boulle and Brandon Philips and Red Hat's Vincent Batts — are actively contributing to OCI's specs. Without the support of these leaders, activity in the appc project has slowed down tremendously since OCI's announcement.

As the specifications continue to mature, it is important to note how important their development is to keeping some companies involved. For example, CoreOS released an updated version of rkt based on appc, yet in the future plans on releasing the same runtime based on the OCI spec.

FIG 1: *The companies with the most active in contributors to OCI spec are Huawei, Docker, Red Hat, IBM, Fujitsu and CoreOS.*

Company Association of Contributors to OCI Specifications
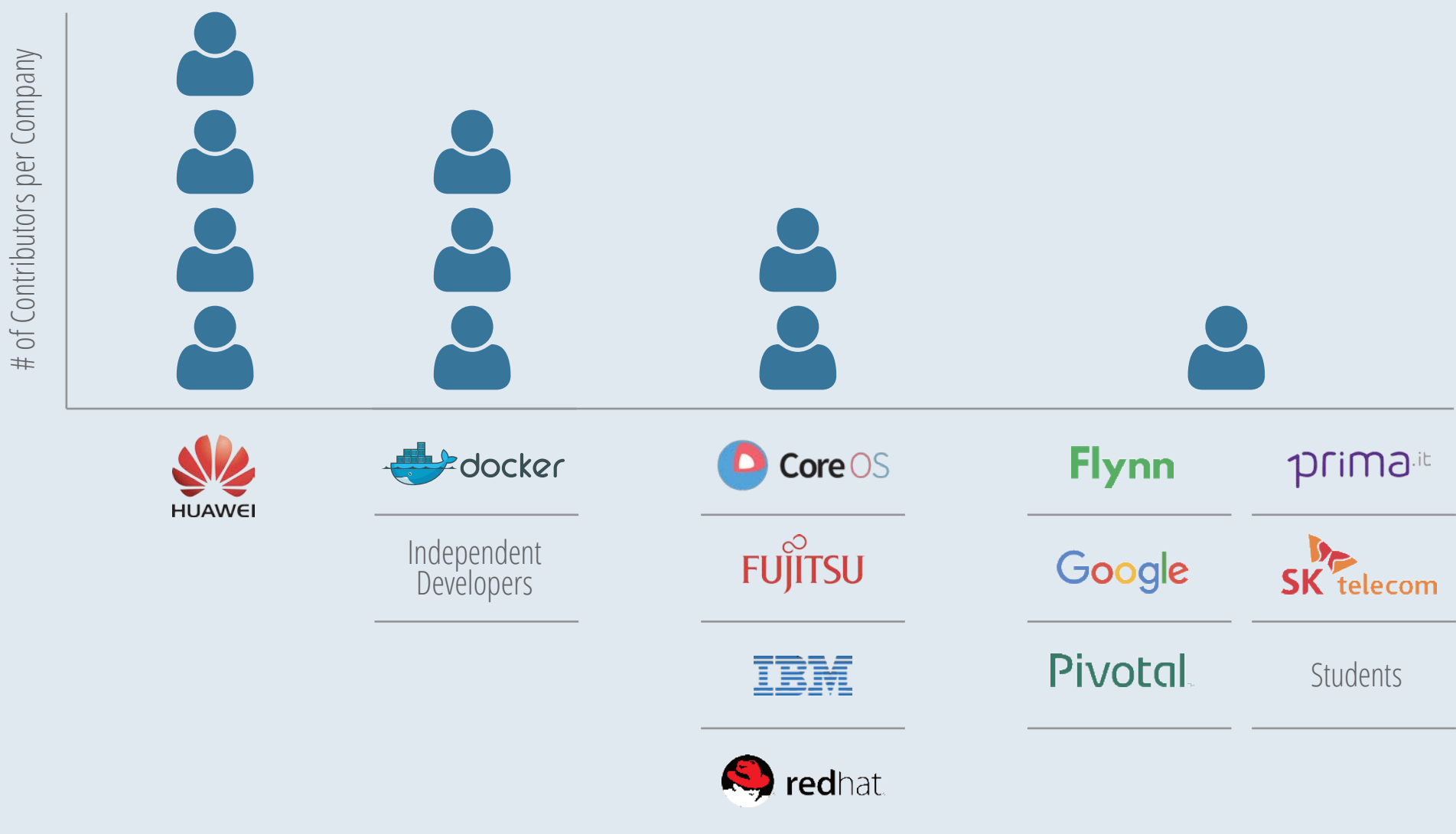
If OCI is to be successful, it will at a minimum need companies like CoreOS to use agreed upon standards that will allow it to continue developing its own runtime while at the same time ensuring interoperability with applications built for Dockerfiles.

Interestingly, CoreOS is still hedging their bets on OCI's success. While heavily involved with the specifications, no CoreOS employee has made a contribution to the runc repository.

Involvement With runC

While specifications are the core of what OCI is about, Docker's donated libcontainer is the actual meat of what developers are currently using. The level of GitHub activity for runC has actually picked up as compared to the work done in libcontainer, with 24 percent of the contributions in the repository happening since libcontainer migrated to runC.

There has also been an uptick in new contributors since OCI was

FIG 2: *The majority of the OCI runC spec is code migrated from libcontainer.*

Contribution to the OCI's runC

24%
New Contributions

76%
Contributions Migrated
from libcontainer
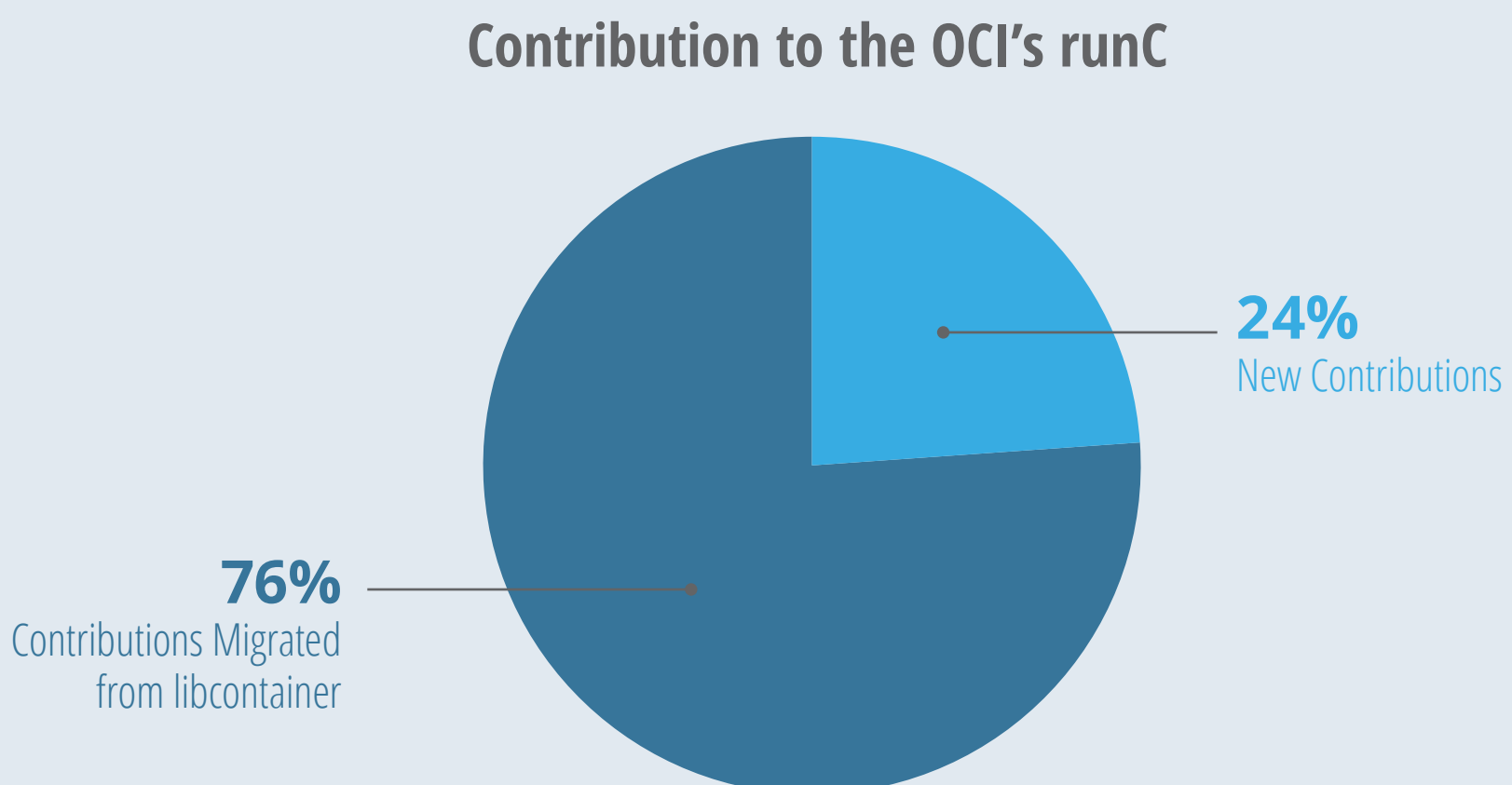
announced, with 29 of the 127 contributors joining in the last two months. However, it is noteworthy that most of the contributions have come from Docker and Red Hat employees. Absent from the list of contributing companies is CoreOS. It is also notable that Google accounted for only two contributions (0.5 percent of the total), as opposed to the 18 percent it accounted for in the original libcontainer repository.

Which OCI Members Are Contributing

Our final analysis was to compare the companies contributing to runC and the membership roster of the OCI. We found that half of OCI's member companies still do not have an employee contributing to the project.

FIG 3: *The OCI members contributing most are Docker, Red Hat, Huawei and IBM.*

Contributions by Company Since runC Was Created

Company	Percentage
Docker	~39%
Red Hat	~33%
Huawei	~5%
IBM	~3%
GE Healthcare	~2.7%
Fujitsu, Prima.it	~2.5%
Unknown	~2.2%
Odin	~1.5%
SK Telecom	~1.2%
Pivotal	~1%
EMC, Microsoft, Independent Developers	~0.8%
Google, Heroku (Salesforce.com), Mantika, Student	~0.5%
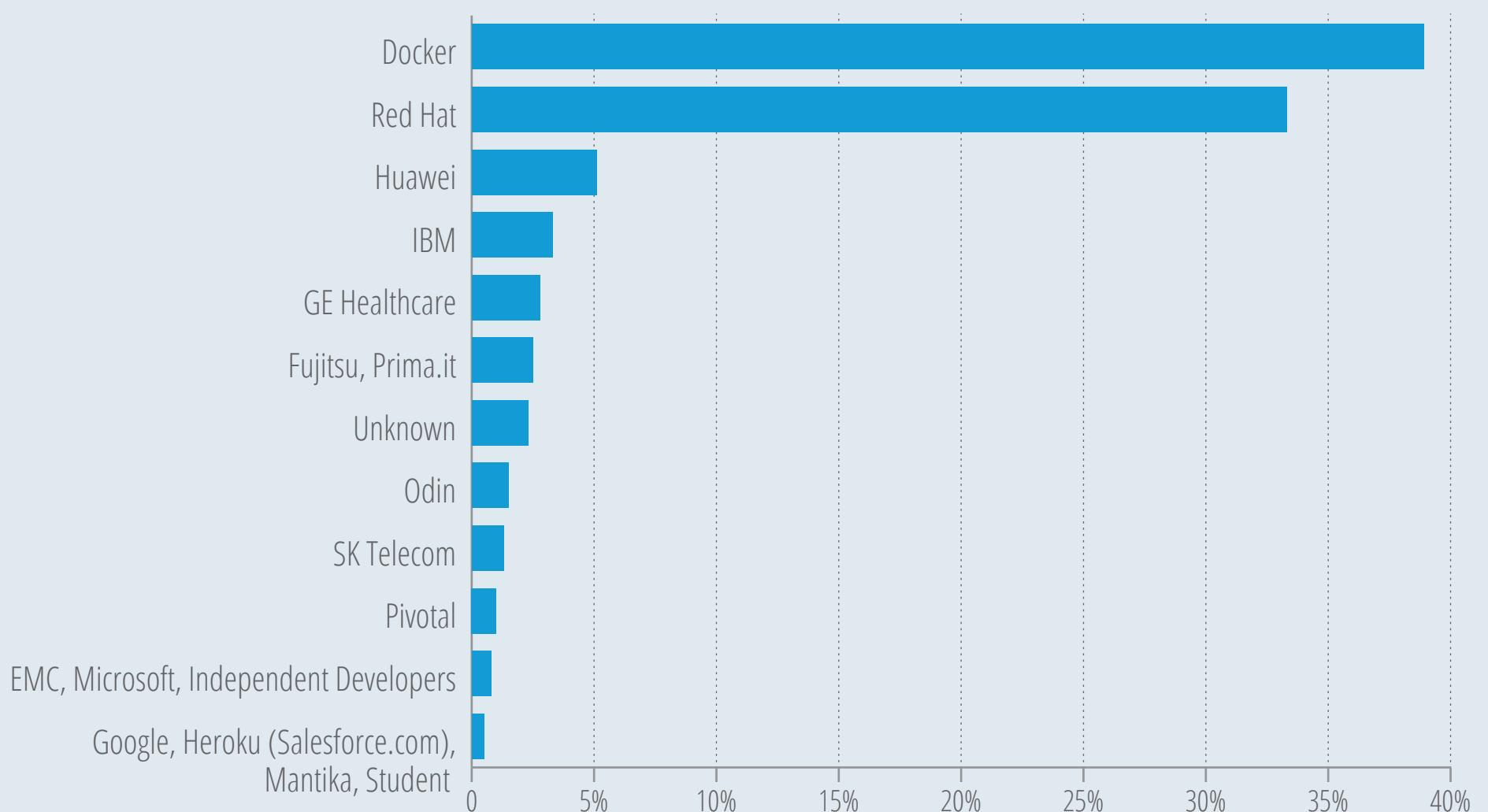
What does a lack of participation mean? Of course, a company can adopt or support standards without actually contributing to an open source project. If that is the case, then OCI will have accomplished at least part of its mission by creating a standard that everyone can build upon.

But it does speak to what becomes of open source projects that become larger organizations with participation from commercial technology providers. The gap widens. Large companies make up most of the contributions and the smaller providers represent a tiny percentage of the total.

CISCO: NETWORKING THE HYBRID CLOUD

by **MARK BOYD**
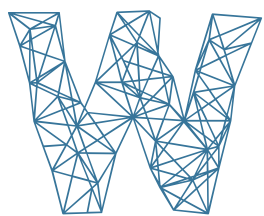
With its expansive history in IT network hardware, [Cisco Systems](#) has been acting as much of the "pipes" of the Internet, responsible for routing a large chunk of the world's data traffic. From their vantage point as network provider, Cisco has seen growth and disruption cycles play out amongst the companies that are building the data nodes and the companies that are creating the digital products that travel within the networked systems.

They have also seen the boom and bust of the dot-com cycles, having at one point been valued at a nose-bleeding $500 billion, but now with a current estimated market cap of around $134.83 billion. Perhaps because of having lived through these sorts of these hyperbolic heights and more pragmatic flats, Cisco has in recent years come to define their core business as networking, not necessarily hardware networking.

"Every company is becoming a software company," says [Ken Owens](#), CTO of Cloud Services at Cisco Systems. Owens believes that enterprises and

whole industries risk losing market share and relevance if they do not understand this basic tenet of digitization.

By having seen Blockbuster overtaken by Netflix, Barnes and Noble by Amazon, the birth of a new digital generation of social media, Uber and Airbnb, Cisco Systems has seen first hand that hardware just isn't enough. To avoid facing the same disruption in their own hardware networking business, they needed to develop competencies and products suited to the new software-defined network (SDN).

Now, as hybrid clouds become increasingly acceptable to enterprises looking to make the best use of data and computational storage across a global network of distributed application architectures, Cisco still wants to be the company that others turn to for networking it all together.

> ❝ The open source project Mantl is at the core of their SDN product suite offering.

Owens says that while recognizing the threat of disruption at an industry level, Cisco Systems also saw within the new SDN approach that more and more often developers were being required to string together the data infrastructure that enabled their applications to function.

In a hybrid cloud environment, containers may be running in Docker, be orchestrated with Kubernetes or Mesos, rely on Consul or etcd for service discovery, etc; but more than that, each cloud or bare metal environment may be using a different configuration of tools and infrastructure components. In true hybrid fashion, this may mean some systems of

record were stored in on-premise data centers, while more temporary, real-time data was stored in the cloud on Amazon Web Services or through OpenStack for immediate processing.

"You can get infrastructure very quickly now, but you're leaving all the heavy lifting for the developers, to try to figure out how to plug all of their components together — how they're going to test it, how they're going to provide assurance of their software, and how they're going to continuously integrate and deploy that software over time," he says. "You need to be very quick and software-centric, but you've made it harder than ever for developers to accomplish what they need to accomplish and maintain their code over time."

Mantl aims to offer DevOps engineers an end-to-end testing solution that can be used as a data center and cloud-agnostic microservices platform. Using a range of open source components, Mantl enables DevOps to focus on how their application is running, without having to link together varying infrastructure components (and resolve the integration challenges that come with working across a hybrid cloud environment). Owens describes the typical experience for a developer using Mantl:

> 66 *I build my application like I always do, on my local laptop. I do a Git push to push it off to the cloud environment. From there, I can select my deployment environment, whether it's Amazon or Google or Cisco or a VMware-based cloud, and I can push that code out to that location from my Git push location. Then I can continuously integrate and continuously deploy using that shipped interface."*

After that point in the process, Mantl takes care of continually running and monitoring the applications in production.

THE NEW STACK SUMMARY

Traditionally known for their network infrastructure, Cisco sees containers as a key technology for enabling customers' digital transformation.

KEY PROJECTS

MANTL - Integrates Kubernetes, Mesos, Consul and other open source projects to deliver container-based application deployment for large-scale hybrid cloud environments.

KEY PARTNERS

Google, Hashicorp, Mesosphere

KEY ACQUISITIONS

Piston CloudOS

https://github.com/CiscoCloud

A member of the Cloud Native Computing Foundation and Open Container Initiative.

thenewstack.io

It is part of a new approach that Owens is calling "Hybrid DevOps." Along with Mantl — aimed at solving developers' pain points — is a new suite of hybrid cloud products that includes the InterCloud Fabric — aimed at enterprises deploying applications across hybrid-cloud architectures and at cloud providers themselves who want their customers to easily access a whole hybrid ecosystem. InterCloud offers a single interface for enterprise to manage their hybrid-cloud workloads: security, compliance and governance can all be managed regardless of whether the enterprise is running its architecture on bare metal, on an OpenStack cloud, on VMware-based clouds or — as is increasingly the case — a mix of all.

"Being able to develop your application, to separate that development from the deployment scenario, and to enable that deployment to work across the InterCloud, meant that you could truly develop your app once, deploy it into whichever infrastructure you want (or whatever infrastructure your company has selected to deploy into), including the public domain. Then you can run and manage it from that location" says Owens.

Without a move into hybrid-cloud networking, Cisco could potentially be at greater risk of facing disruption themselves, as they have seen across

industries that have traditionally used their cable and WiFi networks. "Like we did with the Internet," Owen says, "we're trying to do the same thing with clouds, because we feel like there's a very close analogy to what happened with the Internet back in the 90s, and what's happening with cloud today."
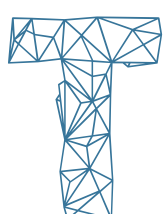
Listen to The New Stack Makers podcast with Ken Owens on:

» Soundcloud or » YouTube

Cisco is a sponsor of The New Stack and The Docker and Container Ecosystem ebook series.

HOW THE GO PROGRAMMING LANGUAGE HELPS DOCKER AND THE CONTAINER ECOSYSTEM

by **JOAB JACKSON**

To build Docker, developers made the somewhat novel choice of using Google's relatively new Go programming language. Given that Docker's virtualization technology would need to be responsive, a more traditional choice might have been a low-level language such as C.

Going with Go turned out to be a wise decision. Go gave the Docker team a simple and powerful language for working with their initial operating system of choice, Linux.

More importantly, it also provides users with an easy path to incorporate Docker's capabilities into their own environments. Go was built specifically to enable fast development on distributed systems. It allows developers and system administrators to quickly build programs and system tools for cloud computing environments without worrying too much about issues such as dependency management or concurrent programming.

For the developer, Go makes it easy to package pieces of code functionality, and then build applications by assembling these packages. The packages can then be easily reused for other applications as well.

And just as Go helps in development, it also makes life easier during deployment. A Go program can be simple to implement because, once it is compiled, it typically does not require external libraries. A large standard core library provides all the functionality a programmer may need, in most cases.

This means an administrator can develop a Go program, copy it over to any remote server with an installed version of Go, and not worry if the program will require additional libraries that the server may not have.

The whole process is made even easier with Docker. There is a special version of the language, available in the Docker Hub, that can package an application within a Docker image during the compilation process, so that it can be immediately deployed. Developers can even build their applications from within the container itself. They don't even need to have Go on their own computers.

Just as it is easy to install and run Go programs within Docker, it's very easy to manage Docker containers with Go. Docker provides an API library for interacting with Docker containers using Go.

Customer Adoption

Go was launched in 2009, and was ready for production use by 2012. Since then, it has been embraced by developers worldwide, particularly in China, where it is most heavily used (though there is discussion about if this difference is as large as the data suggests).

In fact, a surprising number of large enterprises, not the quickest of adopters, have already adopted Go, including Chevron, Verizon, Disney, Walmart, Comcast, General Electric and even Microsoft. IBM has used Go to build a security library for the Internet of Things, as well as for network connectivity software for a swarm of Docker containers.

Many web-scale companies have adopted Go as well. Not only Google, but Facebook and Amazon have gone with Go.

Of course, Google itself provides a hosted Go runtime, as part of its Google App Engine collection of platform services, though other cloud providers, such as Amazon Web Services, support the language as well.

As of latest count, GitHub hosts over [86,600 Go-based software projects](#), and those are only the open source public projects. No doubt, many private and proprietary software projects are built in Go. And it's not just a matter of what companies have used Go — many developers report they are planning to use Go in the near future.

In a 2015 StackOverflow developer survey, 19 percent of DevOps developers said they were planning to use Go in the future, representing a predicted increase from the 10 percent currently using Go. As a whole, eight percent of developers plan to use Go in the next year.

In addition to Docker, other container support software has been developed in Go, including the Kubernetes container management software and Hashicorp's Nomad cluster management software.

It is pretty easy to see that Go will be the core language for the container ecosystem for many years to come. There is a synergy at work here between Docker and Go: if someone sees a bug to fix in Docker, or wishes

Plans to Use Go in the Future
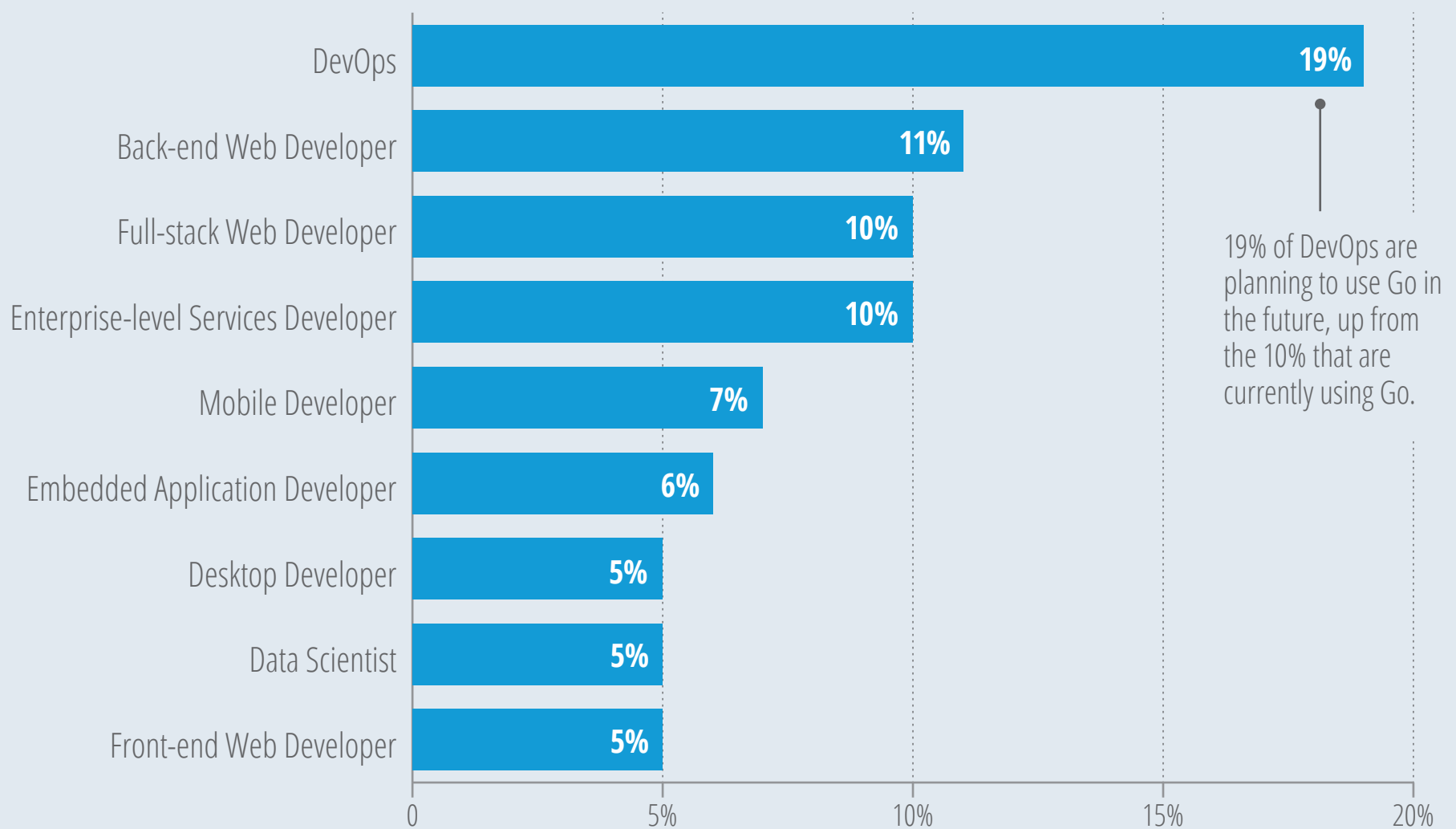
Role	Percentage
DevOps	19%
Back-end Web Developer	11%
Full-stack Web Developer	10%
Enterprise-level Services Developer	10%
Mobile Developer	7%
Embedded Application Developer	6%
Desktop Developer	5%
Data Scientist	5%
Front-end Web Developer	5%

19% of DevOps are planning to use Go in the future, up from the 10% that are currently using Go.

thenewstack.io

FIG 1: *Breakdown of developers planning to adopt Go in the near future.*

to improve the software in some way, they use Go, and so they join the league of Go programmers worldwide.

Best of the Old and New

Go's appeal comes in part from how it combines the best practices from old-school statically-typed programming languages and the newer, dynamically-typed languages often used for building web applications.

Go was created by what could only be called a supergroup of veteran programmers: Robert Griesemer, who worked on the V8 JavaScript engine and the Google File System; Rob Pike, Unix and the UTF-8 character encoding format; and Ken Thompson, Unix and C co-creator. By 2007, all

three had worked at Google, and all were frustrated with the limitations of existing languages. The languages of the day just weren't well-suited for systems programming. So they set off to build Go to tackle this task directly.

Traditional statically-typed languages, such as Java and C++, offer more rigorous controls over data types, which make them safer and faster. Their enforced disciplines, however, make them more difficult to use.

As a result, many web-facing startups over the past decade have gravitated towards a new breed of dynamically-typed languages, such as Ruby or JavaScript, which do not enforce the strong typing of data.

While this dynamic typing can simplify work for the programmer, additional costs sneak out during the runtime in the form of increased cost for testing and debugging. Debugging becomes a lot more of a headache, especially with large programs, if the data types aren't enforced from the outset. Programs written with dynamic languages also tend be slower than their statically-typed counterparts, especially when scaled up to large workloads.

The beauty of Go is that the language combines the ease of development offered by dynamically-typed languages with the rigor and speed of the statically-typed languages.

As a result, Go programs run more quickly than those written in dynamic languages, which translates directly into lower monthly usage bills for those companies running their web applications in the cloud.

One company that learned this was consultant Iron.io, which built a messaging system for its clients that it later released as a product, called

IronWorker. The first version of IronWorker was built in Ruby, which the company found didn't scale well. After, the program was rewritten in Go, Iron.io was able to cut the number of virtual servers they needed to run IronWorker from 30 to two.

In addition, Go is geared for distributed computing. It has many built-in features to support concurrency, or the ability to run a program across multiple processors. It can execute many low-level system calls, giving it the ability to work directly with the operating system, which speeds up process time.

The language is also friendly to the programmer. Go was built so that the developer can call up the documentation through a single command line. Testing can be executed through a single command as well.

In short, Docker couldn't have found a better partner than it did in Go.

MANAGING CONTAINERS ACROSS DISTRIBUTED RESOURCES

by **ALEX WILLIAMS**
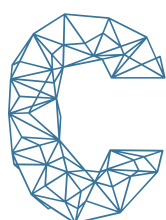
ontainer Solutions is a consulting group out of Amsterdam. They frame the current discussions about programmable infrastructure in terms of machines.

Virtual machines can require large amounts of resources, especially when developing on older machines. Running hypervisors can lead to stress on a system if resources are not allocated correctly, or without a user specifying that a certain amount should be used to compile logs or complete other tasks as demanded by the project in question.

Efficiency is a primary issue with scaling clusters. Application latency is one outcome of high intensity workloads that do not scale appropriately. Resources with inefficient architectures have a high cost when applying traditional solutions to the problem. This might mean adding more storage and networking boxes, which do not have the elasticity that more modern architectures offer.

Phil Winder for [Container Solutions](#) writes that scaling in distributed systems depends on the service that is getting scaled. A stateless app can scale effectively. It needs compute resources and a record in a load balancer. A database, on the other hand, is much more difficult to scale horizontally. Winder [writes](#) that the "application must decide what has responsibility for synchronisation of the data; is that the database's responsibility, or some function of a distributed file system?"

There are a host of other factors that come into building out distributed architectures. And not surprisingly, there are all sorts of orchestration environments to consider.

Orchestration aids in running apps across multiple containers, instead of just one. In The New Stack's survey of container-related vendors, we asked about orchestration, including it in a category with scheduling, management and monitoring tools. In this context, we found that almost two-thirds of the vendors had offerings in this area for the second quarter of 2015. While 71 percent also plan orchestration-related offerings in the future, several indicated they were working on revising their product or developing a partnership. Later in the survey, companies also told us how they are managing containers internally. There was no dominant tool, with "no product" being cited by more than four percent of respondents. The New Stack Container Directory has 55 products and projects associated with orchestration management, with half of those representing an open source effort.

A Brief History of Automated Resources

In 2006, Amazon launched Amazon Web Services (AWS), establishing itself as a pioneer in offering virtual resources. AWS created a sudden

availability of almost unending compute ability, and customers were billed only for the resources they used. Almost overnight, apps could be deployed without the worry of getting a data bill with a comma. It was relatively simple and the costs were less — two core factors that led to behavioral changes and market shifts.

AWS made a powerful argument for moving away from the traditional models of IT. Amazon could operate at levels of efficiency that even the most well-managed IT shop would be hard-pressed to match.

The next evolution involved some new thinking about how to get the most of unlimited resources in a self-serve manner, fully automated with resources more oriented for developers and their application requirements.

This concept of programmable infrastructure is part of a much bigger trend that Google calls "warehouse-scale computing," a term that came from a Google paper in 2009, which is at the core of why Google is so interested in containers.

Here are some examples of components that reflect warehouse-scale computing, as described by CoreOS CEO Alex Polvi in an interview:

- Commodity hardware and underlying compute resources.

- Commodity switching and networking fabric.

- Application packaging (containers).

- Resource scheduling and orchestration.

- Linux for running orchestration and packaging on commodity hardware.

What is actually different this time is that we are building systems that have intelligent software and simple hardware — what Google is known for, Polvi said. It means more compute resources can be added to get more capacity in applications. It means any individual server is meaningless. We will think about everything in terms of applications, not individual servers.

"Dynamic scheduling" is a key component of Google's definition of "cloud-native computing as Google's Craig McLuckie described it on The New Stack.

The line-of-business manager should be able to run an application on some on-demand infrastructure without the help of the system administrator. The manager shouldn't have to worry about servers or any other physical infrastructure. Instead, they should think about deployment only in terms of logical computing resources.There is a sea of compute available where any job can be scheduled to run.

> 66
> *It turns out there are some things that computers do better than people. One of those things is really thinking about, in real time, where your application should be deployed, how many resources your application should have access to, whether your application is healthy or unhealthy, whether some level of remediation needs to happen.*
>
> *By moving away from a world where it's an operator-driven paradigm — where you're creating these static, dead things — to a world where your application is alive, and being actively managed, and dynamically, reactively, observed and watched by a very smart system ... [that] changes the game."*

In other words, software algorithms can schedule and allocate the jobs against the available resources with such efficiency, that doing so would lead to significant cost savings. No more calling the IT department to requisition a server that may show up six weeks later. It also makes the enterprise much more agile by lending it the ability to quickly spin up new applications, move resources to optimal usage and stay ahead of competitors.

So how do you build a smart system from a data center filled with dumb servers? This is where tools like Google Kubernetes and open source Apache Mesos data center operating system come in. Also of note is Docker's platform, using its Machine, Swarm and Compose tools.

Google Kubernetes

Google developed Kubernetes for managing large numbers of containers. Instead of assigning each container to a host machine, Kubernetes groups containers into pods. For instance, a multi-tier application, with a database in one container and the application logic in another container, can be grouped into a single pod. The administrator only needs to move a single pod from one compute resource to another, rather than worrying about dozens of individual containers.

Google itself has made the process even easier on its own Google Cloud Service. It offers a production-ready version of Kubernetes called the Google Container Engine.

The cost is crazy inexpensive as well: managing the first five clusters is free, and the service costs a miniscule 15¢ per hour for each cluster after the fifth one.

Between pods, labels and services, Kubernetes offers different way to interact with clusters:

- Pods are small groups of Docker containers, able to be maintained within Kubernetes. They are easily deployable, resulting in less downtime when testing a build or QA debugging.

- Labels are exactly as they sound, used to organize groups of objects determined by their key-value pairs.

- Services are used for load balancing, providing a centralized name and address for a set of pods.

- Clusters on Kubernetes eliminate the need for developers to worry about physical machines; the clusters act as lightweight VMs in their own right, each capable of handling tasks which require scalability.

Apache Mesos

Apache Mesos is a cluster manager that can help the administrator schedule workloads on a cluster of servers. Mesos excels at handling very large workloads, such as an implementation of the Spark or Hadoop data processing platforms.

Mesos had its own container image format and runtime built similarly to Docker. The project started by building the orchestration first, with the container being the side effect of needing something to actually package and contain an application. Applications were packaged in this format to be able to be run by Mesos.

Mesos is supported by Mesosphere, which offers the Mesosphere Datacenter Operating System (DCOS). As the name implies, DCOS

promises the ability to pool resources and then dynamically schedule jobs against them, as if all the servers worked together as a single entity.

Mesos is an open source software, originally developed at the University of California at Berkeley. It sits between the application layer and the operating system and makes it easier to deploy and manage applications in large-scale clustered environments. It can run many applications on a dynamically shared pool of nodes. Prominent users of Mesos include Twitter, Airbnb, Netflix, PayPal, SquareSpace, Uber and more.

The distributed systems kernel was born out of UC Berkeley's AMPLab about five years ago. Benjamin Hindman was a PhD student at Berkeley at the time; he went on to work at Twitter for four years before joining Mesosphere. He is one of the original creators of the Apache Mesos project.

For years, an IT executive would build out the hardware and then run the software across it, treating all those servers as pets. In today's world, it's the applications that come first. The hardware gets abstracted, treated more like cattle than anything else. This pets-versus-cattle comparison is used often these days to explain how modern data centers must be treated in today's application-centric and data-intensive world. The bottom line is that apps come first.

The data center is a virtual corral. Resources are pooled and apps are launched in much the same way as operating systems work on computers. Mesos has the capability to scale the number and size of apps as well as the compute, storage and other resources that are needed for different types of workloads. Its core is in the kernel, which performs the main functions, such as allocating resources to apps.

Docker

Docker Machine, Docker Swarm and Docker Compose are designed to work as an orchestration system. Docker also works closely with the Mesos community.

Docker Machine

According to Docker, Docker Machine enables one-command automation to provision a host infrastructure and install Docker Engine. Before Docker Machine, a developer would need to log into the host and follow installation and configuration instructions specifically for that host and its OS. With Docker Machine, whether provisioning the Docker daemon on a new laptop, on virtual machines in the data center or on a public cloud instance, you only need a single command.

The pluggable backend of Docker Machine allows users to take full advantage of ecosystem partners providing Docker-ready infrastructure, while still accessing everything through the same interface. This driver API works for provisioning Docker on a local machine, on a virtual machine in the data center, or on a public cloud instance.

In its current alpha release, Docker Machine ships with drivers for provisioning Docker locally with Virtualbox, as well as remotely on DigitalOcean instances; more drivers are in the works for AWS, Azure, VMware and other infrastructures.

Docker Swarm

Docker Swarm is a clustering and scheduling tool that automatically optimizes a distributed application's infrastructure based on the application's lifecycle stage, container usage and performance needs.

Swarm has multiple models for determining scheduling, including understanding how specific containers will have specific resource requirements. Working with a scheduling algorithm, Swarm determines which engine and host it should be running on. The core aspect of Swarm is that as you go to multi-host, distributed applications, the developer wants to maintain the experience and portability. For example, it needs the ability to use a specific cluster solution for an application you are working with. This would ensure cluster capabilities are portable all the way from the laptop to the production environment.

The Swarm API is for ecosystem partners to create alternative or additional orchestration tools that override Docker's Swarm optimization algorithm for something more nuanced to particular use cases.

This is what Docker has been calling their "batteries-included-but-swappable" approach. Some users may be comfortable with using Docker Swarm to identify optimized clustering of a multi-container, distributed application's architecture. Others will want to use the clustering and scheduling part of Swarm to set their own parameters, while still others will look to an ecosystem partner's alternative orchestration optimization product to recommend the best cluster mix.

Multi-container applications running on Swarm can also be built using Docker's [Compose tool](#). The Compose tool uses a declarative YAML file to maintain a logical definition of all application containers and the links between them. Compose-built distributed applications can then be dynamically updated without impacting other services in the orchestration chain.

Docker Compose

Docker Compose enables orchestration across multiple containers. Database, web and load balance containers, for example, can all be assembled into a distributed application across multiple hosts. The orchestration is composed by expressing container dependencies in a YAML file and, again, managing via the Docker user interface.

Summary

Orchestration is still a topic that few people know little about, but it will be crucial for companies building microservices environments. There are questions to consider about virtualized infrastructure and how to deal with issues, such as stateless and stateful services. There are the schedulers, the service discovery engines and other components that make up these new kinds of management systems. What these orchestration platforms do more specifically will be questions we answer later in the ebook series.

DOCKER AS THE DEVELOPER-FACING TOOLBOX FOR THE INTERNET-AS-OPEN-PLATFORM

by **MARK BOYD**
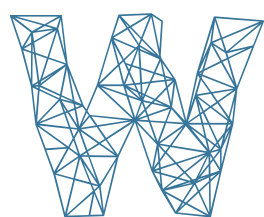
While Docker's core mission remains the same, its process of incremental innovation now means that today's Docker already looks quite different to how the technology platform was seen even a year ago.

"We are seeing the beginning of tools coming out of the Docker community that can be used outside of the Docker platform," says CTO and Founder Solomon Hykes. "We are going to see a hundred times more of that. These are tools that improve the software plumbing of the Internet. And on top of that plumbing, there is a separate developer community that wants to build apps, that wants to get stuff done."

Hykes says that developers are rethinking how they build everything, with a lot of developers realizing that Docker is providing the "developer-facing toolbox" that makes use of containers, virtual machines or other distributed architectures to help scale applications. "It's not just containers," says Hykes.

SVP of Marketing at Docker, David Messina, explains that this new toolbox era for Docker still reflects the twin objectives at the heart of the company's mission statement:

> **"We are about building tools of mass innovation, and building a programmable layer for the Internet.**

"Now we have use cases that are anything from the largest government institutions like the U.S. GSA [General Services Administration] to the smallest, most innovative startups. We have use cases from genomic sciences to IoT devices," says Messina. Throughout this, he says, the focus is on providing a platform that is looking at the whole lifecycle of an application: "At the heart of it, our focus is on taking developers and operations from the very start all the way through production."

How Docker has metamorphosed this year — and how its leader Hykes sees it as continuing to evolve — reflects the process of incremental innovation that has been central from the emergence of containers in Linux to the current use of Docker in Microsoft, Git and distributed systems.

Hykes describes how 2002 to 2010 represented a time of experimentation, when two projects in particular moved the needle on containers in Linux. The first was VServer. Hykes describes VServer as a means to patch the Linux kernel in order to split things up into virtual servers, an early version of what today we would call containers. The second project was OpenVZ, which Hykes says "transformed the Linux kernel so that you could run containers in production. They were very successful at that." Despite its

success, OpenVZ never managed to get the technology merged into the Linux kernel and always required a patch to make it possible.

The turning point came in 2010 with the introduction of the LXC project. At this time, cgroups and namespaces were introduced, making containers a functionality available within the Linux kernel. "It became possible to use something that looked like a container without patching your kernel. And that made all the difference in the world," describes Hykes.

At the time, Hykes was leading dotCloud, an infrastructure platform as a service (PaaS) that was committed to applying standards in the deployment of distributed architecture for applications. "We spent three years running a cloud platform in production using LXC, so we had a lot of operational experience. We learned that tool was not practical, so we wrote a tool that was more stable." And thus Docker was born.

Messina says that this is when Docker then began its own process of integrating innovation into its platform, identifying each challenge along the way and addressing them one by one. "At Docker, every step along the way has involved incremental innovation," he says. "It started with how do

docker

https://github.com/docker

A member of the Cloud Native Computing Foundation and Open Container Initiative.

THE NEW STACK SUMMARY
Standing on the shoulders of giants, namely the Linux kernel, Docker created a set of tools that is intiating a sea-change in developer experience and enterprise computing.

KEY PROJECTS
Docker - The Docker platform for building, deploying, shipping, and running container-based applications is available on Github as open source. Docker, Inc. donated its container format and runtime to the newly-formed Open Container Initiative.

KEY PARTNERS
Amazon, Microsoft, IBM, Red Hat

KEY ACQUISITIONS
Socketplane (networking), Orchard Labs (orchestration)

we build a model that separates the application concerns from the infrastructure concerns. Then it became a distribution problem, and as you move forward, then the bigger problem of orchestration and networking becomes the challenge."

Messina points to Docker Machine, Swarm, Compose and, most recently, Notary as examples of how this incremental innovation has played out.

Hykes, meanwhile, is already looking at the next frontier. "We started pulling the thread with Notary. The Notary announcement is the beginning: how do you do large scale, secure distribution of content across the Internet without relying on a closed platform like iOS or Android?" Hykes sees one of the next tasks as enabling "the open platform that is the Internet" to let DevOps "safely and at super-large scale get content distributed across any machine to any application."

He believes that the tech to make this possible will require peer-to-peer technologies, cryptography advances and more experimental solutions.

> 66*That will get to a really scalable solution that also preserves the privacy of users. I think that is something that is very important. Making the Internet more decentralized, more secure by default. Shifting the center of gravity more towards the individual developer and away from these big centralized cloud services. There are a lot of problems to solve there, but it is possible. It's going to happen."*

Messina agrees, seeing Docker's future roadmap centered on supporting developers and operations to utilize the Docker toolbox that enables control of the application lifecycle all the way from initial development on a single laptop to global distribution in production.

"The ramifications of what we are doing are so broad. Our focus is about continuing to enable a platform that is looking at the whole lifecycle of an application, so our focus will be on adding more and more innovation; our job is to simplify and accelerate that journey of the application lifecycle for organizations."

Docker is a sponsor of The New Stack and The Docker and Container Ecosystem ebook series.

THE CONTINUUM: FROM CONTAINERS TO SERVERLESS ARCHITECTURES AND UNIKERNELS

by **ALEX WILLIAMS**
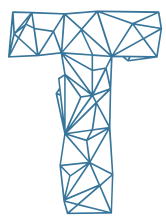
The complexities of managing Docker and containers in production is one of the greater challenges that comes with its adoption. For millions of programmers and plumbers the issue is about simplicity. It's evident that containers represent a major progression from virtual machines.

From a more universal perspective, this progression represents a continuum that will lead to serverless architectures and other more efficient means of managing complexity, such as unikernel technology. Containers represent a new continuum, but the architecture is quite complex, which raises questions about what technologies will follow.

Building a Containerized Architecture

Rally Software has a container architecture stack that engineer Matt Bajor used at the Strange Loop conference to discuss the complexities of microservices architectures, containerized architectures and unikernels.

Architectures vary on different platforms. The Rally containerized architecture has dynamic DNS and load balancers to manage the traffic, along with etcd for service discovery. There are object stores, memory stores and relational stores. The services are tied together with Kubernetes or Mesos and are managed with configuration management tools, such as Puppet or Chef. These systems, as a whole, are effective, but they are quite complex.

According to Bajor, the Rally application stack is built on hardware and drivers. It has a virtual hardware infrastructure with a hypervisor that serves as a partitioning system for dividing the hardware. On top of the hypervisor are operating systems, such as CentOS. On top of that is the Docker runtime with the the shared libraries and language runtime. We then have the application and the application configuration.

Application stacks have many layers of abstraction and different forms of isolation, Bajor said. How things interact at any given time is difficult to predict. There are many upgrade cadences to match. A large portion of it is redundant. Isolation occurs at the hypervisor layer, in the user processes and in the container. This is all for a single app on a single server, or a single user.

The Linux Challenge

There are lots of layers that the developer knows about implicitly, but does not have an explicit understanding of how they all work. Systems are historically heterogeneous and largely over-generalized. In new architectures, hardware is largely commoditized, with virtual machines working on hypervisors and virtual device drivers.

Bajor makes the point that the Linux kernel and users are natural enemies, as considerable complexities are built into Linux to keep apps safe from users, users safe from other users, and apps safe from other apps. That means a lot of code and complexity in the system. There are also lots of permission checks on the operating system. These checks have roots in an era when time sharing was necessary on larger systems. There were lots of apps and users, all working on the same hardware, all interacting and working together.

There are lots of inefficiencies to manage, Bajor said. There are virtual drivers on the system, but there may also be hard drives and even tape. Storage and RAM is often underutilized and taking up additional resources.

The Linux kernel has a large attack surface, making it easier to get into the system. Security patching is done, by an operations team, which creates incompatibility issues with the developer teams who are writing the code. It can be difficult to track the interactions between the two teams. Each changeset comes in different shapes and sizes, creating issues, such as outages.

How Did We Get Here?

There is a long progression of technology that has its roots in mainframes, the client/server era and now distributed systems. The integration of heterogeneous environments comes down to systems and software compatibility. For example, the hypervisor has to work with the hardware and operating system. The container has to run on the virtualized infrastructure or on bare metal. It requires a lot of code and complex integrations to make things work. Configuration management, load balancers and a host of other technologies have all contributed to this rich

and varied ecosystem. For example, microservices platforms require multiple services to work together. Schedulers, service discovery engines, data store environments and all the associated APIs have to be compatible in order for the microservices environment to work.

Efficiencies have been sought by customers as they seek ways to make infrastructure more cost effective. Former VMware CTO Steve Herrod said in an interview that selling VMware's virtualization technology in its heyday became a matter of helping a customer not buy another server. In those days, it was an easy sale, as customers needed ways to reduce, not increase, the number of machines running in their data centers.

Now with the advent of containers, we see a real effort to make things simpler. Compatibility is hard to do, but it is now easier to do because of rich tool ecosystems, which increasingly have their roots in open source. Microservices are easier to build. Containers are easier to deploy. Now the market is ready for the next phase, and that's built on the premise of efficiency, and perhaps most of all, performance.

Unikernels: Meeting Today's Performance Needs

Bajor points out that performance is a key-value driver for containers, but they do have an associated complexity. For even higher performance gains, there is a growing interest in technologies such as unikernels which, proponents say, simplify the technology stack.

Unikernels are uniquely specialized virtual machines, similar to an application stack — they have application binaries and virtual hardware underneath, Bajor pointed out. In the middle is a library operating system

that has its own network stack. Unikernels are self-contained and have far fewer layers compared to a container stack.

The unikernel has minimal code, but still operates on the same hardware. There are no permissions, nor is there isolation, Bajor said. Unikernels implement the bare minimum of traditional operating system functions. They do just enough to enable the application it powers.

By removing the traditional operating system layer, unikernels remove the unneeded bulk of standard operating system environments, along with their associated attack service. Unikernels are extremely light, allowing higher density on commodity hardware. They can run their own services that are born when the need appears, and die as soon as the need disappears. Some of these transient microservices may have lifespans measured in seconds, or even fractions of a second. They are just-in-time computing services, which exist only when there is work to do, allowing you to maximize the use of your computing infrastructure.

Unikernels have a corollary to the new serverless architectures gaining popularity. We see this with services such as AWS Lambda, which Amazon is investing in deeply as is evident in [The New Stack's coverage](#) from the 2015 AWS re:Invent conference.

Lambda was devised to run user-generated functions in the cloud, without the need for the user to worry about any of the supporting stack running said functions. It's not so much a Platform-as-a-Service as a Function-as-a-Service. Lambda is a stateless computer service, meaning it runs a user-defined function that collects data from an outside service, works on the data and delivers the output to another service.

The code has to be triggered by an external event, such as an incoming

call from a mobile app, web service, or by another AWS service. A change in an Amazon S3 bucket or DynamoDB table can also trigger a function call.

Lambda is novel in that it strips away all need to worry about any supporting infrastructure. No more maintaining EC2 instances just to run a single function. Infrastructure issues, such as scaling or maintenance, are whisked away in abstraction. Typical Lambda jobs include image conversion, change notifications and file compression — in fact, AWS has prebuilt functions available to handle those specific tasks.

Amazon touts Lambda as a way to coordinate operations of so-called Internet of Things systems. It can do both data ingestion and command-and-control by sending those requests and commands down to the end device.

A Lambda job, for instance, could be used for inbound rule processing. Spam detection could be a Lambda function, one that gets evoked for each new email. Log files can be monitored in real time.

Summary

In summary, what does this say about the continuum that we see as the world develops new technology stacks? There is a new set of application patterns and deployments. We have achieved a degree of compatibility, and the next effort is to get better performance.

Containers and unikernels are similar technologies, with unikernels described as "a Docker container on a diet." By bringing unikernels to Docker, it could allow for greater familiarity with the technology.

If one is building mission-critical systems, unikernels give developers explicit control over core security areas of their application. Developers can choose the output result while working with unikernels. In comparison, Docker containers have everything they need to run enabled by default. In unikernels, many features are turned off by default, meaning more initial setup and choices for a project team. According to unikernel proponents, once these choices have been made, the result is resilient new stacks that are secure, while also being customizable to the needs of the project.
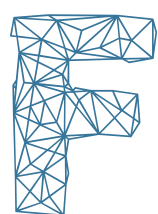
What does that mean for containers? In an interview, Docker CTO Solomon Hykes said the ramifications for what Docker is doing are pretty broad.

> **" Our focus is about continuing to enable a platform that is looking at the whole lifecycle of an application ... our job is to simplify and accelerate that journey of the application lifecycle for organizations."**

But in the meantime, the real win will come to the organizations that can offer the speed that comes with lightweight systems. How they will do that will require a new thinking about how to transition from monoliths to microservices and all that goes with it, as we move along the continuum to an ever more programmable world.

SURVEY: HOW THE I.T. LANDSCAPE WILL SHIFT TO ACCOMMODATE CONTAINERS

by **LAWRENCE HECHT**

From a standing start, an ecosystem has quickly grown to support the use of containers. 2015 picked up where 2014 left off, with continuing VC investment, new company formation and a slew of new products and technology initiatives. To better understand the dynamics of this ecosystem, The New Stack initiated a survey of companies that provide or plan to provide services associated with containers. We found that vendors are adjusting their portfolio of services to meet the demands of both application developers and IT operations. Executives at these companies tell us their products have an impact on the IT landscape, with PaaS and provisioning technologies affected most. The survey also gave us the opportunity to learn how these vendors, leading adopters of container technologies themselves, have adapted their own technology stack.

The container ecosystem consists of a wide variety of companies that support the use of containers with their own products. Some of these

Participating Companies

thenewstack.io

FIG 1: *Senior executives at 48 companies described current and planned offerings to support the use of containers.*

companies are what might be called "pure-plays," meaning their offerings were specifically designed to support Docker and containers; others are traditional vendors, quickly adapting their own offerings to remain relevant in a containerized future.

In April and May of 2015, The New Stack invited 106 vendors associated with Docker and containers to participate in an ecosystem survey. About 45 percent of these companies responded. The companies surveyed represent a broad cross-section, as can be seen by comparing the 48 responding companies listed above with a broader view provided by our Container Ecosystem Directory.

Key Takeaways

- Management and orchestration are top priorities for Docker and container ecosystem companies.

- Companies plan to invest in developer tools, networking and security functionality.

- Vendors fear competition from end-to-end solutions offered by IBM, AWS and Heroku.

- The PaaS market is evolving quickly as vendors offer platforms to deploy containerized applications.

- PaaS providers will face new competitors but are likely to benefit from increased market demand.

- VMware and configuration management systems like Chef and Puppet Labs face disruption.

- Ecosystem vendors are targeting enterprise/ISV practitioners of DevOps.

Vendor Offerings

When asked to broadly categorize their current offerings, the top two categories of products reported by surveyed companies indicate that early players are taking broad, platform-oriented approaches to the market, tackling areas such as application development and deployment platforms, schedulers and orchestration (Figure 2). The categories that these products are being marketed under varies. Companies in this space market tools targeting similar pain points as compared to PaaS platforms,

Vendors Focused on Deployment Platforms, Orchestration, Developer Tools
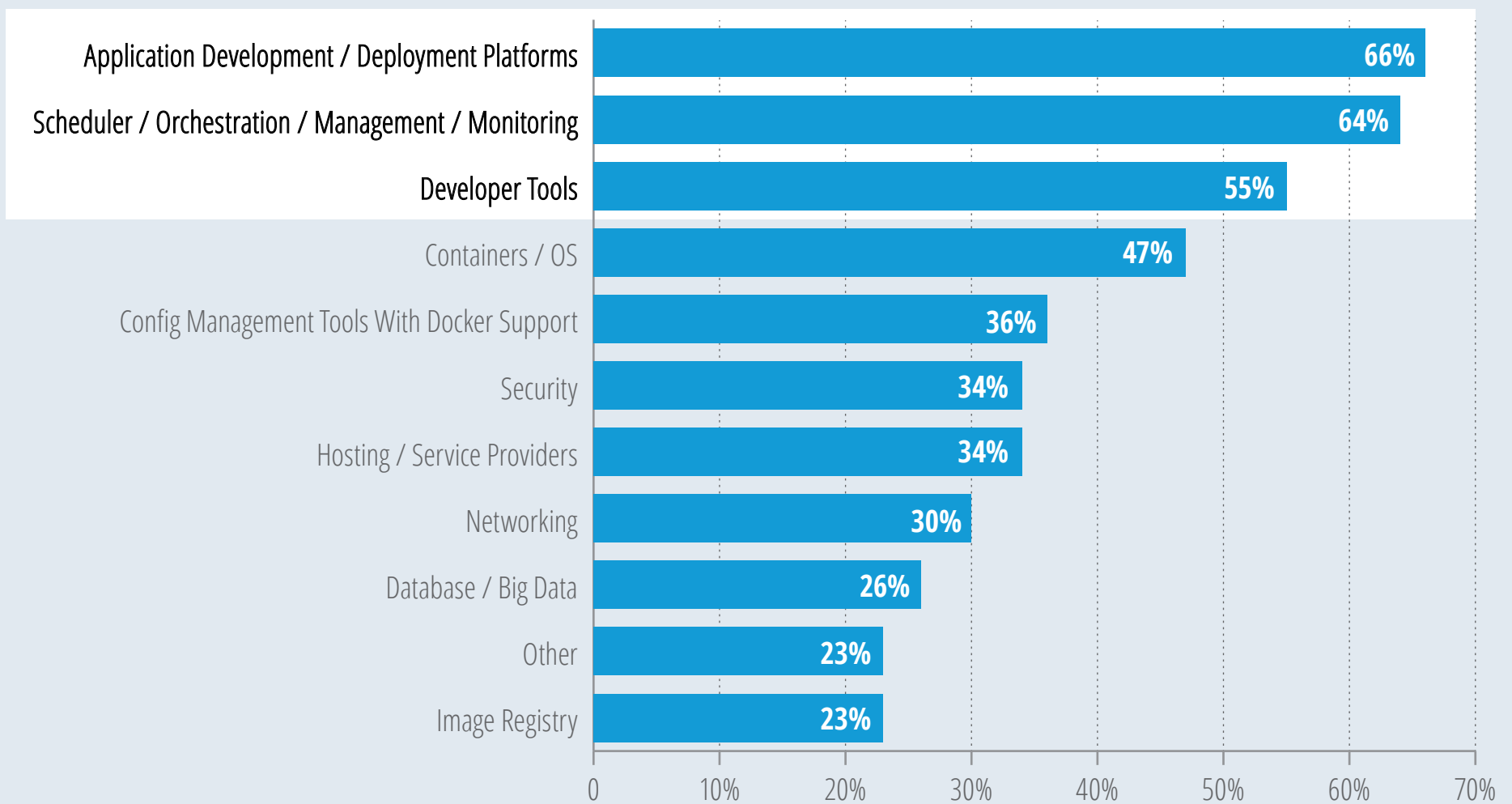
Category	Percentage
Application Development / Deployment Platforms	66%
Scheduler / Orchestration / Management / Monitoring	64%
Developer Tools	55%
Containers / OS	47%
Config Management Tools With Docker Support	36%
Security	34%
Hosting / Service Providers	34%
Networking	30%
Database / Big Data	26%
Other	23%
Image Registry	23%

Q: In what categories of the container ecosystem do you CURRENTLY have products or services? n=47.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

FIG 2: *Two-thirds of vendors offer ways to develop and deploy platforms. Almost as many companies provide tools to orchestrate, schedule or otherwise manage container-based applications.*

cluster management systems, data center operating systems, CI/CD tools and hosting environments.

Many companies in the container ecosystem sell products and services traditionally viewed as being for developers, but which offer functionality to manage infrastructure — and vice versa (Figure 3).

The perception that application development and orchestration are interconnected is a strong theme. Half of the companies surveyed currently have offerings that purport to be in both spaces. In fact, among companies that are currently in one of the two main categories, over 80 percent expect to have products in both spaces within two years.

Current Capabilities: Industry Veterans vs. Startups
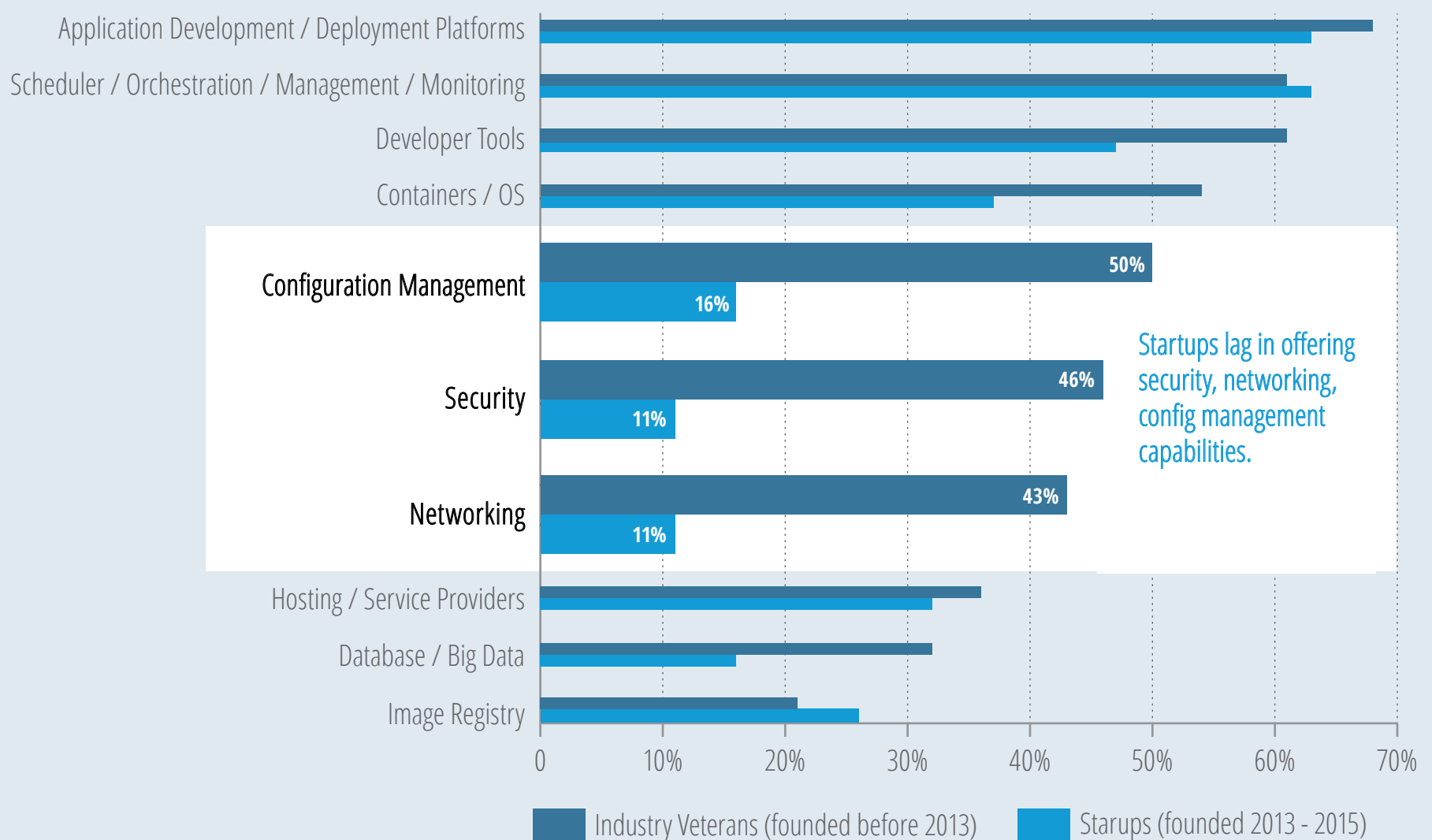
Configuration Management: Industry Veterans 50%, Startups 16%

Security: Industry Veterans 46%, Startups 11%

Networking: Industry Veterans 43%, Startups 11%

Startups lag in offering security, networking, config management capabilities.

Legend: Industry Veterans (founded before 2013) | Starups (founded 2013 - 2015)

Q: In what categories of the container ecosystem do you CURRENTLY have products or services? Veterans = 28. Startups = 19.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

FIG 3: *Startups are much less likely to offer container-related security, networking and configuration management capabilities.*

A company's current product portfolio is the result of its journey over the last few years. Many of the younger companies in the survey seem to be building solutions based on broad visions of how customers should be using container technologies.

We expect that as the market matures and clear leaders in the platform space become apparent, activity in the space will shift to offering niche solutions to specific customer pains, such as security. The larger, older companies we surveyed were significantly more likely to offer configuration management tools, security and networking functionality.

This is partially because many of the leading configuration management vendors have started to add Docker support to their existing products. It

is also likely that many of the larger companies were describing a security or networking tool they sell that may or may not actually be purpose-built to deal with containers. We also asked the surveyed companies where they saw their product portfolios going. The answers we received reflect vendors' focus on solutions that address more narrowly defined pain points (Figure 4).

Developer tools, networking, security and configuration management functionality with support for Docker and containers all rank high on vendor roadmaps. Nearly one-third of companies are planning to invest in developer-focused tools, either via internal development or acquisition.

FIG 4: *In addition to improving existing offerings, firms are most likely to focus M&A and internal development efforts on adding new functionality for developer tools, networking, security and Docker support for configuration management.*

Ecosystem Firms to Invest in Tools, Networking, Security and Config Management

Category	%
Developer Tools	28%
Networking	23%
Security	23%
Config Management Tools With Docker Support	23%
Image Registry	13%
Scheduler / Orchestration / Management / Monitoring	11%
Application Development / Deployment Platforms	11%
Database / Big Data	6%
Hosting / Service Providers	4%
Containers / OS	2%

■ % of vendors with plans to develop new functionality/products internally or via M&A

Q: In 2 years, how do you plan to primarily address the functionality and needs within each category of the container ecosystem? n=47.
Source: The New Stack Container Survey, completed May 2015

Every company that is planning to develop either new developer tools or configuration management tools with Docker support is already providing application development/deployment platforms. As these platforms mature, it makes sense that their providers will want to offer additional tools to fill gaps in the developer experience.

While many companies surveyed are offering broad product suites, there is also recognition that they cannot do everything themselves (Figure 5). We found companies most likely to seek a partner for hosting services. While companies like Tutum offer end-to-end Docker-centric hosting environments, many vendors are offering software that can be deployed to any cloud or on-premise infrastructure.

FIG 5: *Instead of developing capabilities internally, companies expect to work with partners to address the hosting and data-related needs of their customers.*

Ecosystem to Partner to Deliver Hosting, Data Capabilities

Category	%
Hosting / Service Providers	45%
Database / Big Data	38%
Containers / OS	30%
Networking	26%
Image Registry	26%
Security	21%
Scheduler / Orchestration / Management / Monitoring	17%
Config Management Tools With Docker Support	15%
Developer Tools	9%
Application Development / Deployment Platforms	9%

9 of the 12 companies planning partnerships to provide networking functionality were founded in 2013 or later.

■ % of vendors with plans to provide functionality via partnerships

Q: In 2 years, how do you plan to primarily address the functionality and needs within each category of the container ecosystem? n=47.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

Container vendors will likely look to partnership programs offered by Amazon Web Services (AWS), Microsoft Azure and Google Compute Engine (GCE) to provide a more complete solution for customers, while remaining leery of those companies' own container offerings.

Similarly, while companies recognize the importance of databases and big data capabilities, most choose to leave those capabilities to specialists. For networking functionality, almost half of the younger companies in the ecosystem are seeking to partner to address this gap in their offering.

Target Market

Most vendors see themselves as automating traditional application development or IT processes, but a few are targeting more specific areas. When we asked about what types of end users they are targeting, about three-quarters said either "IT Operations & Admin" or "Application Development." However, an unusually large number of participants opted to write-in answers that mentioned DevOps or a combination of IT operations and application development. One respondent wrote: "We consider DevOps — ops people who code or developers who have to think about running an application — as our target customer." The New Stack believes that if DevOps were offered as a choice, it would have likely been chosen more often. Another respondent noted:

> **"***Today, the primary use case is to provide application development environments to build new applications that will benefit from Linux container packaging.***"

Companies cited targeting both large enterprises and SaaS/ISV firms in approximately equal numbers (Figure 6). This indicates an attempt at

balancing the pursuit of early-adopter technology companies with a desire to tap into the large budgets of traditional "big enterprise."

As use of containers expands, vendors are learning the types of functionality this target market wants. As a whole, executives believe that customers are looking for security, simplified container management, and new orchestration and deployment features.

Some participant quotes about technical issues driving user demand:

> **"***Complexity in getting from dev to production efficiently. Existing alternatives are complex and constraining, and have a large management overhead.***"**

FIG 6: *Large enterprise and software providers are the types of customers vendors in the container ecosystem want to acquire.*

Companies Targeting Large Enterprises and ISV/SaaS Vendors

Category	%
Non-IT Large Enterprises	24%
Independent Software and SaaS Vendors	22%
Other	11%
No Primary Target	11%
Data Center and Service Providers	11%
Small Businesses	9%
Midsize Enterprises	9%
Don't Know	2%

0   5%   10%   15%   20%   25%

■ % of vendors targeting different enterprise types

Q: What type of enterprise are you primarily targeting? n=46.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

" *Challenges dealing with security and compliance in public, private, hybrid and multi-cloud environments, including gaining visibility into the security state of containers and microservices.*"

" *The need to modernize infrastructure in order to capitalize on the business opportunity driven by the emergence of container-dependent technologies in the cloud services marketplace.*"

" *Docker/containers in production is a lot of work and hassle... moving to microservices architectures comes with a lot of hurdles.*"

" *Managing their own continuous delivery platform is a distraction from working on their product... They much rather have somebody else take care of scalability, security and maintenance.*"

IT Ecosystem Impact

Without a doubt, the rise of Docker and container technologies is making waves in IT. The New Stack asked a series of questions about how vendors think their products are affecting customers' technology stacks. Their answers point to those vendors and technologies most impacted by the industry's adoption of container technologies.

Only 15 percent of companies say their products are being used for specific, narrowly-defined use cases. As one respondent noted:

" *Tools and services for this new type of infrastructure [are] full of promise to one day replace or supplement significant portions of data center environments.*"

PaaS is the category that surveyed companies expect to be most impacted by the introduction of their new container-related products (Figure 7), with almost three-quarters of survey respondents indicating this choice.

This might seem like a high number in light of various studies indicating that actual adoption levels of PaaS are significantly lower than for other cloud services. The New Stack believes, however, that a primary cause of this disconnect is a gradual redefinition of PaaS that has taken place over the past year, significantly broadening the meaning of this term beyond the tightly integrated stacks that it traditionally referred to.

FIG 7: *Most vendors say they are affecting their customers' use of PaaS, IaaS or provisioning and configuration management tools. Although some vendors are totally replacing existing parts of the technology stack, others are supplementing or automating existing products and processes.*

PaaS, IaaS and Provisioning/Config Management Most Impacted

Category	%
Platform as a Service	73%
Infrastructure as a Service	68%
Provisioning and Configuration Management Tool	66%
Hypervisor and Containers	54%
Cloud Operating System	39%
Software-Defined Networking, Network Functions Virtualization	37%
Storage	34%

% of vendors affecting their customers' technology stack
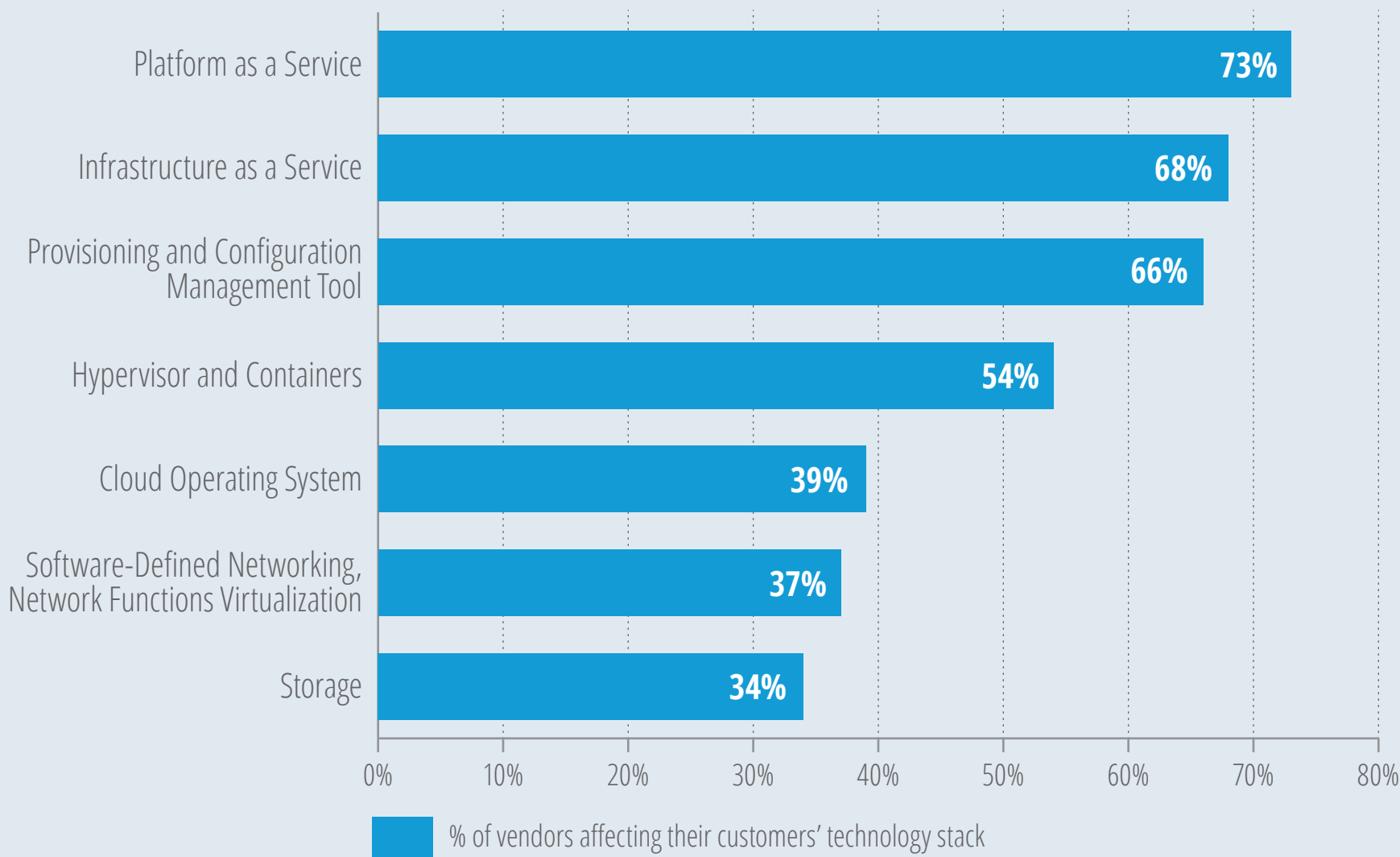
In this context, containers are not eliminating the need for PaaS, but are rather the catalyst for a new generation of IT professionals' reevaluation of how applications are developed and delivered. The companies targeting application development teams sure believe so — all but one said their products are affecting the PaaS part of the IT stack. After PaaS, many respondents see their products influencing the IaaS and provisioning/configuration management categories.

It is not surprising that AWS was the leading IaaS cited. The situation that Chef and Puppet Labs find themselves in is more interesting (Figure 8). Among those that named a provisioning or configuration vendor being affected by their company, 8 out of 17 named both Chef and Puppet Labs.

FIG 8: *Chef and Puppet are the provisioning and configuration management tools vendors see their products having an impact on.*

Incumbent Provisioning/CM Vendors Most Affected
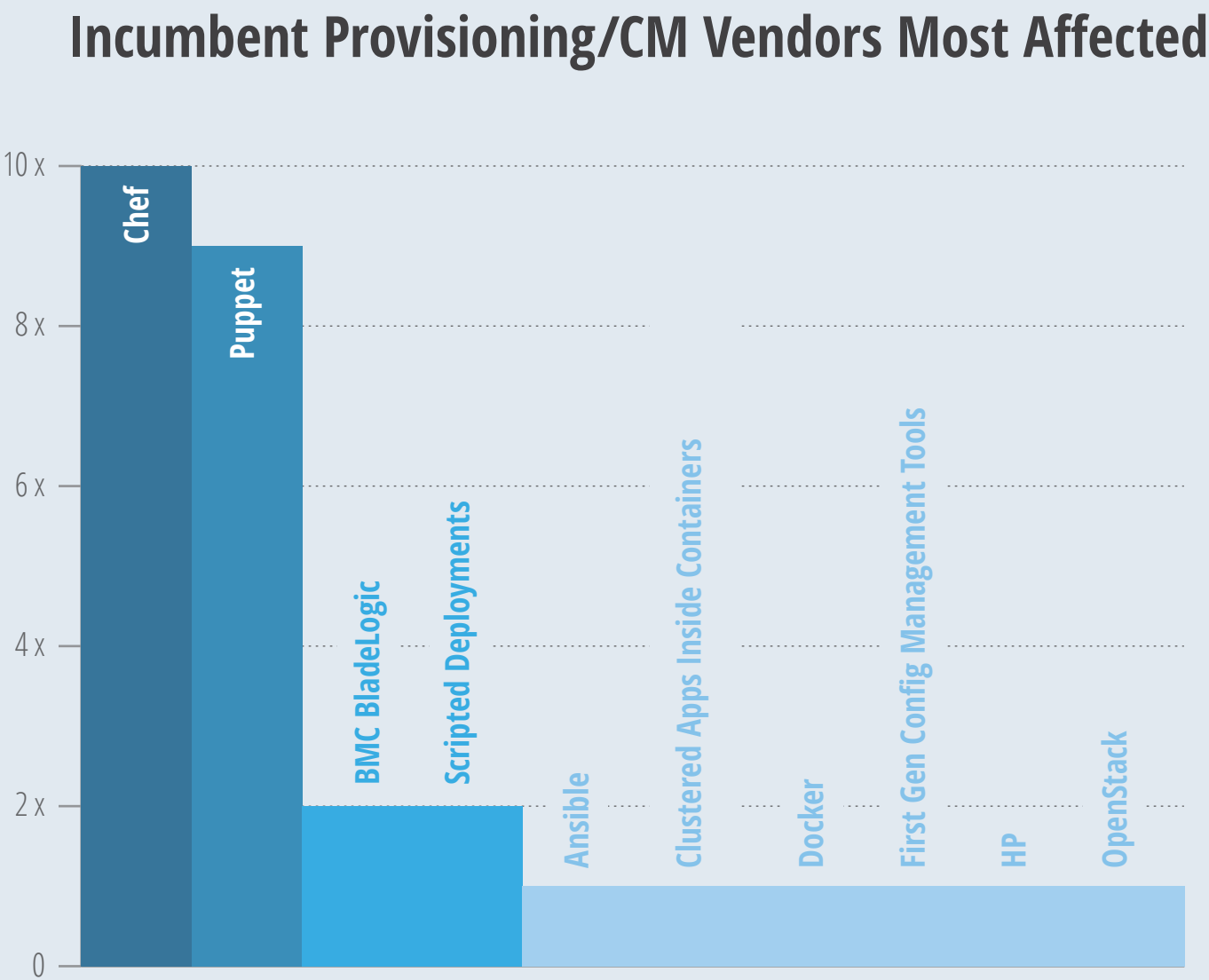
Ecosystem Names IBM, AWS as Top Competition
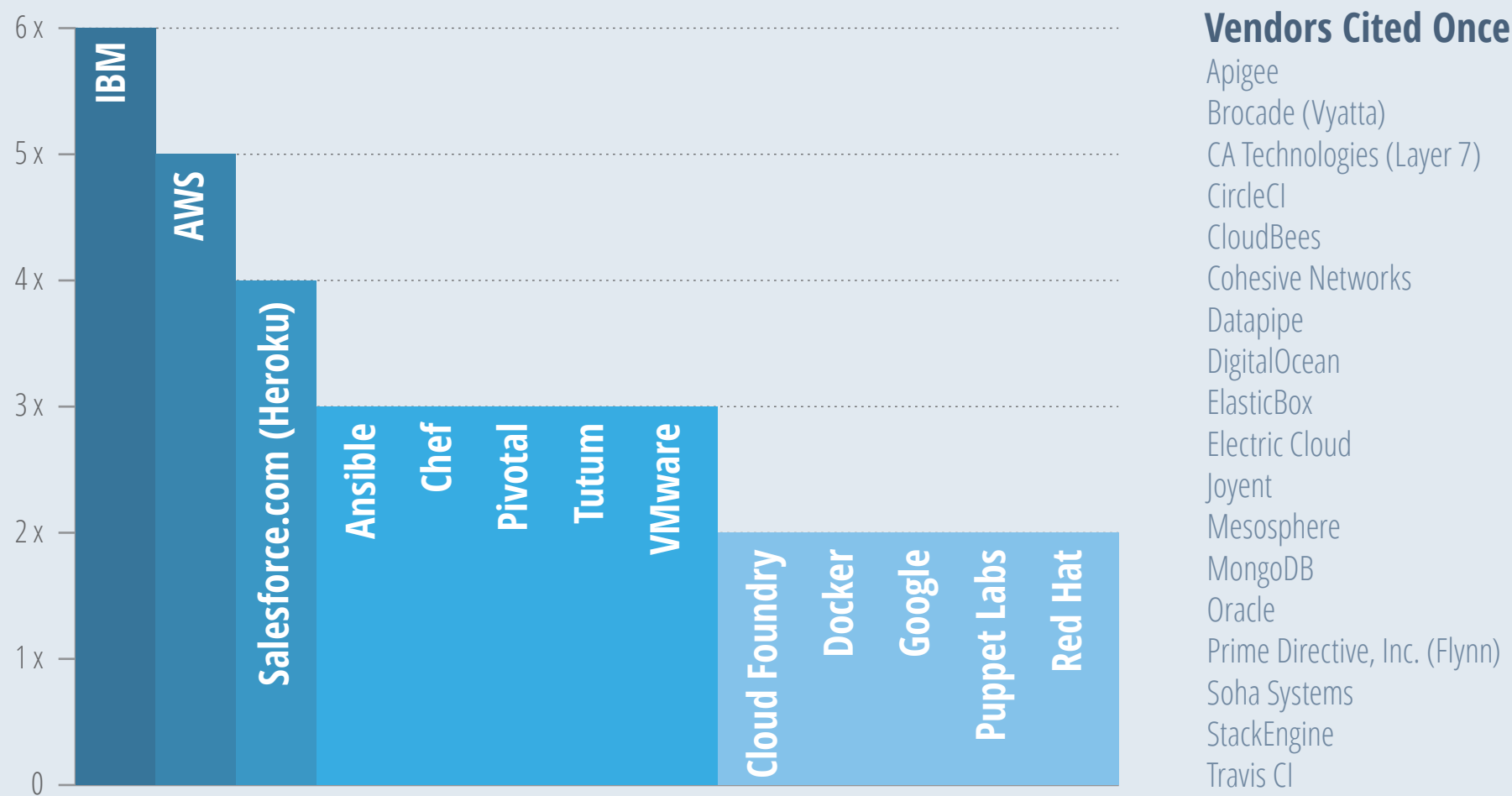
Chart showing competitors by number of citations:
- **IBM**: 6x
- **AWS**: 5x
- **Salesforce.com (Heroku)**: 4x
- **Ansible, Chef, Pivotal, Tutum, VMware**: 3x
- **Cloud Foundry, Docker, Google, Puppet Labs, Red Hat**: 2x

Vendors Cited Once
Apigee
Brocade (Vyatta)
CA Technologies (Layer 7)
CircleCI
CloudBees
Cohesive Networks
Datapipe
DigitalOcean
ElasticBox
Electric Cloud
Joyent
Mesosphere
MongoDB
Oracle
Prime Directive, Inc. (Flynn)
Soha Systems
StackEngine
Travis CI

Q: What orchestration / management / monitoring tools do you use? If you are a provider of these tools, it's OK to indicate that you use your own tools. n=42.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

FIG 9: *When asked who their top one or two competitors are, IBM and AWS received the most votes. Only two companies cited Docker as a primary competitor.*

These results indicate two things: these companies are synonymous with the space itself and they are being threatened with disruption caused by container technology.

As more and more vendors offer "pure-play" container-based alternatives to traditional configuration management, we expect this to present new challenges to Puppet Labs and Chef.

The New Stack found that the surveyed companies are generally aligned with Docker, with 42 out of 43 respondents either currently partnering with Docker (60 percent) or planning to partner with the company in the future (37 percent).

Only one company didn't expect a Docker partnership in the future. Although Docker has good relations with these companies, there is a split about how important Docker's technology plans are to their own strategic direction. While we believe Docker must execute very carefully to secure its position in the container market, it faces limited threat from surveyed vendors. Only two of the 31 companies willing to name their primary competitors cited Docker (Figure 9). Instead, companies are more concerned about PaaS providers, like IBM, with its Cloud Foundry-based Bluemix offering, and Heroku. AWS is also named as a competitor, at least in part because of its EC2 Container Service (ECS).

It is notable that Tutum, a provider of specialized Docker container hosting services, was mentioned by several startups as their main competitor.

These responses suggest that the providers surveyed fear being locked out of participating in end-to-end hosted solutions more than they fear competing with individual point products or projects.

VMware was also named as a competitor by several companies in the survey. Although VMware has been developing new offerings in this space, it is more likely that container vendors are positioning themselves as an alternative to existing infrastructure that relies on hypervisors.

In fact, among companies seeing an impact on their customers' hypervisor or container environment, 80 percent say VMware is the vendor impacted most. As use of containers continues to progress, the threat to VMware is that new workloads will not rely on hypervisors, and that VMs will not only be supplemented by containers, but made superfluous.

Who's Using What?

A common refrain in today's "software is eating the world" era is that traditional enterprises need to act more like software companies in order to be successful. Along these lines, we can learn a lot from looking at what the vendors we surveyed are using to support the development of their own projects (Figure 10).

Being able to operate across multiple cloud platforms is one of the main benefits of using containers. Thus, it is no surprise that companies are supporting the use of more than one infrastructure to deploy their

FIG 10: *With its dominant position in the IaaS market, it's not surprising that over 80 percent of vendors have deployments on AWS infrastructure. However, most vendors are looking to be platform independent and have solutions running on multiple cloud infrastructures.*

Container-Based Solutions Deployed on Variety of Infrastructures

Infrastructure	Percentage
Amazon Web Services (e.g., EC2)	81%
Linux	73%
Bare Metal	56%
OpenStack (any vendor)	48%
Google (e.g., GCE)	48%
Microsoft Azure	46%
VMware	42%
Other Open Source Platform	38%
Windows	33%
Other Virtualized Environment	31%
Other	17%

Q. What infrastructure is your solution deployed on? Check all that apply. n=48.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

Ecosystem Firms Use Variety of Orchestration/Management Tools
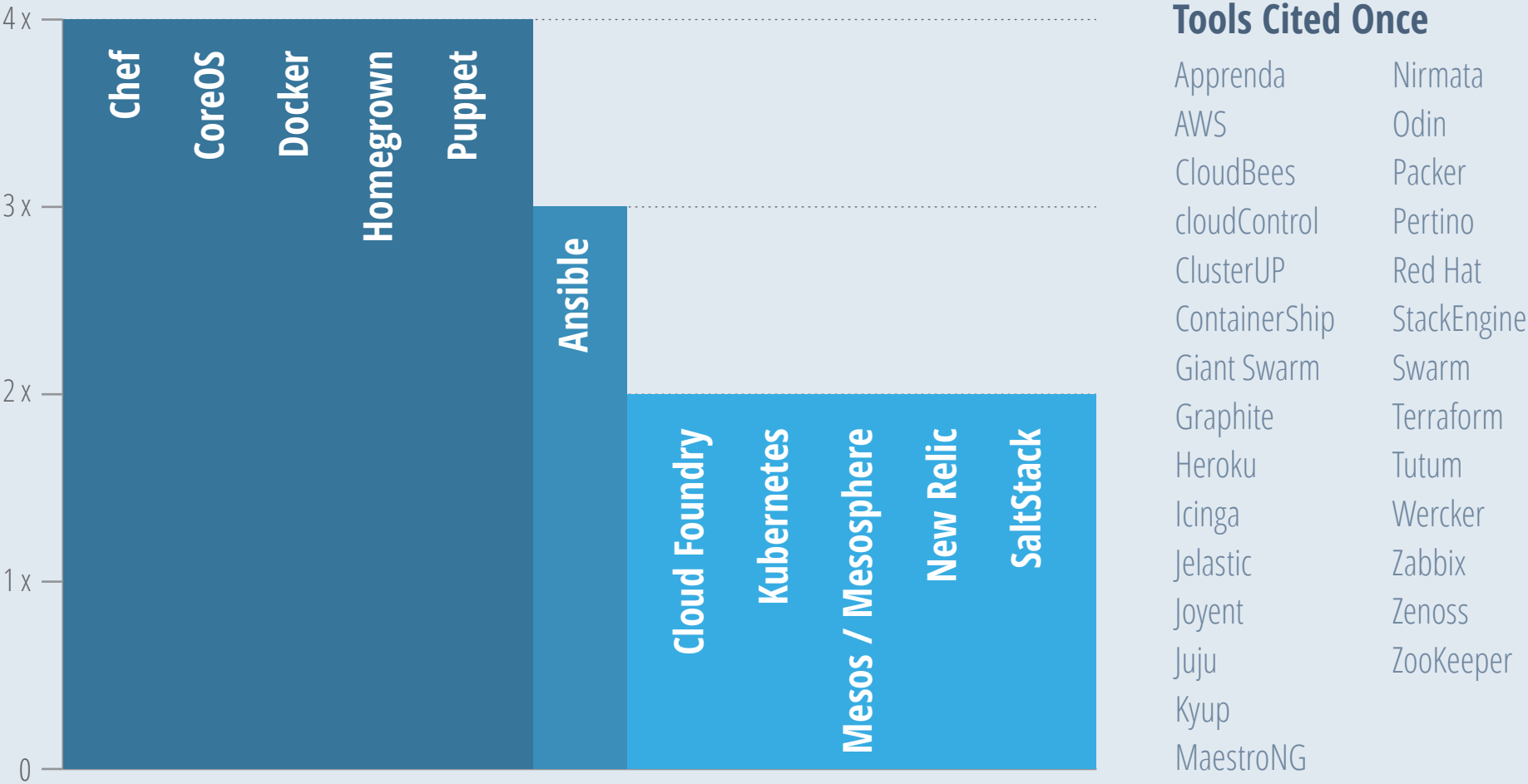
Tools Cited Once

Apprenda	Nirmata
AWS	Odin
CloudBees	Packer
cloudControl	Pertino
ClusterUP	Red Hat
ContainerShip	StackEngine
Giant Swarm	Swarm
Graphite	Terraform
Heroku	Tutum
Icinga	Wercker
Jelastic	Zabbix
Joyent	Zenoss
Juju	ZooKeeper
Kyup	
MaestroNG	

Q: What orchestration / management / monitoring tools do you use? If you are a provider of these tools, it's OK to indicate that you use your own tools. n=42.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

FIG 11: *Although most ecosystem firms use containers, there is no consensus on what tools should be used to orchestrate their own infrastructure.*

services. While it's also no surprise that AWS leads the lineup of supported infrastructures, most companies are supporting it in combination with other platforms. In fact, nearly a third of vendors' products are used on AWS, Microsoft, Google and OpenStack. This indicates a dominant strategy to be platform independent.

Peering Into the Container Ecosystem's Technology Stack

Participating companies eat their own dog food — that is, they're following their own advice about using container technologies. 92 percent of survey respondents use containers in their internal operations and 96 percent

Go, Ruby Use Common in Ecosystem

Chart (bar values, tallest to shortest):
- Go — 9x
- Ruby — 8x
- Java — 6x
- Python — 5x
- None — 5x
- Node.js — 5x
- Docker — 4x
- Linux — 3x
- REST API — 2x
- PHP — 2x
- Application Development — 2x

Languages/Skills Cited Once

Angular	Kubernetes
Bash	Linux
Basic Linux System Admin Skills	Lua
C#	Middleware Operations
Chef	Monitoring/Alerting
Command-line Interface	Puppet DSL
DevOps	Queuing
General Security Practioner	SQL
IaaS Admin	System Administration
JavaScript	YAML
JSON	

Q: What programming languages and/or skills are needed to use your product/service? n=36.
Source: The New Stack Container Survey, completed May 2015

thenewstack.io

FIG 12: *Go and Ruby are the programming languages cited most often as a requirement to effectively use the products and services in the container ecosystem.*

rely on at least one open source project. Yet it's surprising that "only half" of the companies say their technology relies on Docker.

Even if many companies say they do not rely on Docker, they are still heavily invested in open source, with two-thirds having more than five employees contributing to a project. In addition, almost 50 percent of companies say they are using a NoSQL database internally.

In terms of how surveyed companies are managing their own container-based environments, responses were all over the map, with no single response garnering more than four percent of total responses, and some 30 tools mentioned just once (Figure 11). This suggests a very high degree of fragmentation and an impending shakeout.

> **"** *Our PaaS is container-based. Currently still [using] our own custom implementation from before Docker existed. We're most likely in the future migrating some of our internal components to Docker, Diego, Kubernetes or similar open source solutions, now that there are solid ones available."*

Perhaps surprising to those not following closely is the degree to which the Go language is used by container companies (Figure 12). While it is no surprise that the relatively mature Ruby and Java are being heavily used, 25 percent of companies said that Go is needed to use their product or service. We expect Go is even more popular among these companies for internal applications where developers don't need to worry about whether a customer knows the language or not.

IBM WANTS YOU TO USE A CLOUD PLATFORM OPTIMIZED FOR THE FULL APPLICATION LIFECYCLE

by **MARK BOYD**

For decades, IBM has been a leading provider of middleware which enterprises of all sizes have relied upon to run their business systems. Today, IBM has become a leading cloud platform, enabling the next generation of business transformation by providing a rich platform of cloud services and tools.

When building applications, businesses want to spend the majority of their effort on the differentiating business value they are attempting to deliver. IBM Cloud, through initiatives like Bluemix, has created a cloud platform infrastructure that lets these teams draw on cloud services, so they can remain focused on delivering their business applications.

In part, IBM has done that by building a new catalog of cloud services, including containers, logging, monitoring and delivery tools. IBM envisions a hybrid cloud platform, which provides runtime functionalities alongside a "constellation of services" that DevOps can leverage to allow a more rapid composition, deployment and management of applications. APIs are

then used to integrate the various components and deliver the platform in both on and off premise environments.

"[IBM] is trying to look at that broader space: containers combined with resource managers, orchestration, microservices and service discovery. How do we really form a new platform on which applications can get built, and how do those things then interrelate; and that's the next frontier: how do all of these things start to come together in more interesting ways?" says Jason McGee, IBM Fellow, VP and CTO, Cloud Foundation Services at IBM.

"The piece parts are not as important as how the whole thing comes together. The value is going to be how we build that integrated platform. How do we have oversight over an application throughout its entire lifecycle? How do we leverage that so it is easy to build and deploy applications?"

IBM's agenda to make all of this possible has a three-pronged approach. First is the work of building the hybrid cloud platform that has the capabilities to leverage containers and allow access to the cloud service components as needed; the Bluemix platform is an example of providing this sort of product. The platform is the easier part: they can build that with their engineering team, drawing on the culmination of their experience in virtualization, workload management technology and their experience in building systems that control isolation and resource management.

"As a public cloud provider, we need to be able to run a lot of applications at high density with high efficiency on this shared infrastructure, to be able to offer it at a competitive price, and to be able to operate it

efficiently," says McGee. "We're doing that in a multi-tenant way, where users can share resources, and we're doing that in ways that can exploit underlying infrastructure, like running on bare metal."

The second prong of the approach is to rally the industry toward a common definition of this new platform: it will take a village. McGee sees the second half of this strategy as requiring an active conversation within the industry and amongst the developer communities, encouraging debate and discussion to identify common solutions to hybrid cloud integration challenges.

IBM is hoping they can use their market position, industry reputation, background in helping move Linux to a container-based operating system and support of open source tooling to foster developer communities coming together to solve the next wave of application platform challenges.

"The work we're doing with the Open Container Initiative, with the Cloud Native Computing Foundation — the work we've been doing in partnership with Docker and Google and others — is all about how to drive us toward some common, open solutions for these problems, so that there will be less differences in different cloud provider environments in how these problems are solved."

The third prong of the approach is providing a bridge to enable people to connect their existing investments to this new platform. "Building a bridge means providing our existing middleware, such as application servers, mobile and analytics, in a form that enables existing applications to leverage the new cloud platforms," say McGee. "For example, pre-packaging middleware into containers that can be run in the cloud."

IBM

www.ibm.com/cloud

A member of the Cloud Native Computing
Foundation and Open Container Initiative.

THE NEW STACK SUMMARY
The company perhaps most associated with enterprise systems is playing a huge role in driving a variety of cloud and container innovations and helping to shape the communities around them.

KEY PROJECTS
IBM is a contributor to many of the container ecosystem's most important open source projects, including Docker, Cloud Foundry, Linux, Kubernetes.

KEY PARTNERS
Docker, Pivotal, Google

KEY ACQUISITIONS
N/A

McGee points out that enabling middleware via containers means reimagining how middleware is built and delivered. "Instead of giving people installers, you can give people pre-built Docker images, so as a developer it is easier to consume the software you need."

McGee says that this is the first step in reimagining how middleware can be delivered as components in a larger application architecture:

> **"** *As soon as you do that, an interesting thing happens. You start to ask, 'how do I look at the architecture of that middleware and reform that architecture to take advantage of the strengths of containers?'"*

McGee says that the architectural style of microservices is well-implemented with a lightweight, rapid mechanism like containers. By packaging middleware in containers, it leads DevOps to rethink what that middleware needs to be and how it could be further decomposed and decoupled.

Of course, taking advantage of the benefits of a microservices architecture is more difficult when you first need to dismantle a monolithic system and containerize the various components. "When you have something with a

more monolithic structure to it, you don't necessarily understand all the dependencies within it," explains McGee, who says many of these dependencies are only revealed once the work of decomposing a monolith begins.

With this refactoring, the final step in bridging to the cloud is integrating with all of the services the cloud platform provides.

"If you view containers as an element in a broader cloud platform, some of the services the platform is providing are horizontal concerns, like logging and authentication. In a monolithic architecture, you may have solved all of that in the middleware; but in cloud, you want to leverage all of the services in the cloud, and just keep the application. So how do you extract out all of those horizontal functions?"

This integration with the cloud platform services enables these existing applications to take advantage of the power and unique capabilities only provided within the cloud.

Listen to The New Stack Makers podcast with Jason McGee on:

» Soundcloud or » YouTube

IBM is a sponsor of The New Stack and The Docker and Container Ecosystem ebook series.

DOCKER FUELS RETHINKING OF THE OPERATING SYSTEM

by **SUSAN HALL** *and* **SAM CHARRINGTON**

Since the launch of Docker, there's been an explosion of new container-centric operating systems, including CoreOS, Ubuntu Snappy, RancherOS, Red Hat's Atomic Host, VMware's recently announced Photon, and Microsoft's Nano Server.

Of course, you can run Linux Containers (LXC) on any Linux distribution, and most other major operating systems now have some sort of comparable technology. But what sets these new container-centric operating systems apart is that they are much lighter weight than a traditional Linux distribution.

"These traditional Linux distros have just gobs and gobs of packages," said Kit Colbert, VMware's VP and CTO of Cloud-Native Apps. "They've got 4, 6 GB of stuff in there — the application can't see any of that. With a Java app, you've got to have a JRE inside the container for that app to run. It doesn't need anything outside the container to run. So why do you have 4 or 6 GB of stuff that you just don't need anymore?"

At the same time, the containers have to run somewhere, and that host runs on an operating system as well. Hence the rise of these new container-centric "micro OSes."

The idea of a minimalist operating system isn't new. Stripped-down operating systems have long been embedded in electronic systems, ranging from traffic lights to digital video recordings. And minimal operating systems, often based on Linux, that can boot from a CD or even floppy disks, have been around since the 1990s. But these operating systems were typically designed to run only a single node.

> **"** Today's micro OSes are designed for a world in which, thanks to containers, the entire data center is treated as one giant operating system that spans hundreds or even thousands of nodes.

This is leading to new thinking about operating systems. In this chapter we take a look at the major players and their vision for the future of the operating system.

CoreOS

Rivals give props to CoreOS for pioneering the micro OS even before Docker came on the scene. The company recently added $12 million to its coffers with investment from Google Ventures, and unveiled a technology

For Micro Operating Systems, Size Matters

Micro OSes are designed to be small, but some are more successful than others at achieving this goal.

Traditional OS · Micro OS

CentOS 7	Ubuntu 14.10 Server	Microsoft Nano Server	Red Hat Atomic Host	VMware Photon	Ubuntu Snappy	CoreOS	RancherOS
636 MB	576 MB	410 MB	395 MB	317 MB	215 MB	150 MB	20MB
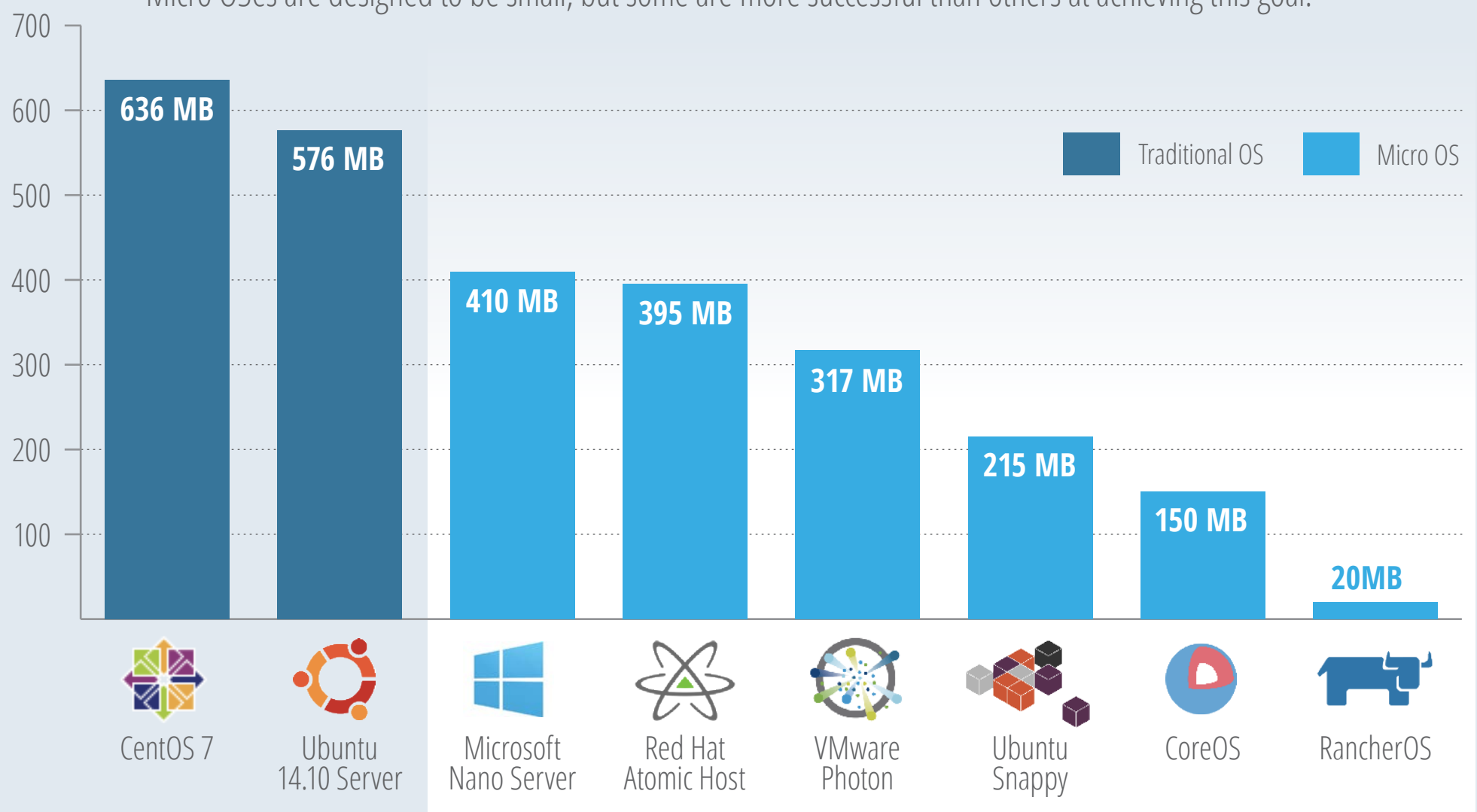
FIG 1: *Micro OSes are designed to be small, but some are more successful than others at achieving this goal.*

called Tectonic, combining its CoreOS portfolio and Kubernetes — Google's open source project for managing containerized applications.

"The container OS is about building the ideal environment for running your application when it's OK to change things," said CoreOS CEO Alex Polvi. "In the traditional Linux server environment, it's hard to change things because applications are just so fragile."

With CoreOS, the operating system is treated more like a web browser, such as Chrome, that is automatically updated as new components are released. Cryptographic signatures help ensure the validity of updates and the integrity of the system as a whole.

While other lean OSes are aimed at a particular flavor of technology, CoreOS aims to be the general-purpose choice. The company officially supports numerous deployment options, and many more community-supported options are available.

"One of the things we've figured out is how to build the OS and run it in all these different environments," Polvi said. "We're very focused on building CoreOS for production-ready environments and it feels like all these other [new OSes] are a long way off from that."

RancherOS

RancherOS, consisting of just the kernel and Docker itself, is one of the smallest micro OSes, weighing in at around 22 MB, co-founder and CEO Sheng Liang said, compared to about 300 MB for VMware's Photon.

While RancherOS also grew out of frustrations similar to those experienced by CoreOS, the company took a different approach in developing its operating system, said Liang. To develop its Docker-optimized micro OS, the company sought to build the minimal technology required to run the Docker daemon on a Linux kernel. To achieve this, it first decided to eliminate systemd, the service-management system built into most Linux distros, and rather use Docker itself to boot the system.

Liang said that systemd is often in direct conflict with Docker. Since containers are often created outside of systemd's view, it often tries to kill them, but winds up only killing the client while the container keeps running. He said that RancherOS worked on finding a way to bridge the camps behind the two technologies for a long time, but is no longer sure there's any way to solve the problem.

"The way the industry's going right now, both the systemd and Docker communities are on a bit of a collision course," Liang said. "Both see themselves as the ever-expanding center of the universe, and it's hard [for either] to listen to another master."

Ubuntu Snappy

Canonical boasts that Ubuntu is the most popular Linux distro for containers, with over seven times more Docker containers running on Ubuntu than any other OS.

Snappy is "a very tiny, thin operating system," said Dustin Kirkland, Ubuntu Cloud Solutions product manager and strategist at Canonical. Snappy Ubuntu Core is the result of applying lessons that Canonical learned in its efforts to create a tiny-yet-robust operating system for mobile devices. To support carriers' and users' needs for reliable system and application updates, the company developed the Snappy technology, which uses "transactional, image-based delta updates" for the system and applications, transmitting only differences to keep downloads small and ensure that upgrades can always be rolled back.

To enhance the security of mobile devices, Canonical created a containment mechanism that isolates each application running on the device. Canonical contends that this same capability offers a level of isolation beyond that available using Docker alone, but few details are available.

In addition to Snappy, Canonical has unveiled a second element of its vision for a containerized world in LXD, a lightweight hypervisor based on LXC, the same Linux feature that made Docker possible. Originally

designed as a mechanism to incorporate LXC-based containers into OpenStack clouds, the launch of LXD caused some to ask whether Ubuntu intends to replace Docker with the technology.

According to Kirkland, however, Docker and LXD are complementary technologies. He adds that Canonical continues to recommend Docker for packaging and running applications. But outsiders aren't necessarily convinced.

Canonical's real motivation is control of the entire stack, said Janakiram MSV, principal analyst at Janakiram & Associates. "Snappy will be their CoreOS, and LXD will become an alternate native hypervisor," he said. "Then they have more VM-like containers. They also have Juju Charms with its own orchestration and provisioning story for the lifecycle of containers."

VMware Photon

The meteoric rise of Docker, itself essentially a virtualization technology, has caused many to anticipate the VMware response, even, if not especially, after the two companies [announced a partnership](#) back in August of last year. Its first response finally came earlier this year, with the announcement of its lightweight OS Photon.

The changing relationship between applications and infrastructure is of key importance to VMware, said Colbert. "When you look at that split, what used to be an operating system used to have some app stuff and some infrastructure stuff," he said. "Now the app stuff is inside the container; the infrastructure is outside of there. Photon is kind of the infrastructure portion of the Linux OS. That's why we want to build that into [its hypervisor] ESX."

Comparing Container Operating Systems

Project	CoreOS	RancherOS	Ubuntu Snappy	VMware Photon	Red Hat Atomic Host	Microsoft Nano Server
The New Stack Summary	Automatic updates ensure always up-to-date, like Chrome browser. Cryptographically signed components enhance security. Broad platform and community support.	Minimal system required to get Docker daemon running on a Linux kernel. Tight alignment with Docker and Docker ecosystem tools.	Grew out of Canonical's efforts to adapt Ubuntu for mobile, resulting in update and security features similar to those of CoreOS. ARM support unique among micro OSes.	Tightly integrated with and optimized for the vSphere ecosystem. Somewhat larger than the smallest micro OSes, but broadest support for container specifications. New yum-compatible package manager.	Uses Red Hat's RPM package manager and built from upstream RPM binaries. Available in RHEL, Fedora and CentOS flavors. SELinux offers enhanced isolation.	A Microsoft ecosystem play. Still very early — little information available beyond a handful of Microsoft blog posts.
Platform Support	Amazon EC2, Azure, DigitalOcean, Google Compute Engine (GCE), OpenStack, Rackspace Cloud, Vagrant, bare metal, plus a variety of community-supported platforms	Amazon EC2, Docker Machine, GCE (Experimental), KVM, Vagrant, VirtualBox, VMware, bare metal	Amazon EC2, Azure, GCE, OVA, Vagrant	Atlas, Vagrant, vSphere, vCloud Air	Amazon EC2, KVM, Vagrant	Hyper-V
Container Support	Docker, rkt	Docker	Generic	Docker, rkt, Pivotal Garden	Docker	Windows Server Containers, Hyper-V Containers
Cluster Management/ Service Discovery	Fleet, etcd	Rancher	Juju	vSphere	Kubernetes	Chef
Source	github.com/coreos	github.com/rancherio/os	launchpad.net/snappy	github.com/vmware/photon	github.com/projectatomic	N/A
License	Apache 2.0	Apache 2.0	Simplified BSD, GPL v3	VMware Tech Preview	GPL v2	N/A
Size	150 MB	20 MB	215 MB	317 MB	395 MB	410 MB

Source: The New Stack

FIG 2: *Micro OSes vary in approach — and technical specifications — based largely on the unique motivations and perspectives of their creators.*

CoreOS will support Photon. Photon is really targeted to the VMware product line, according to CoreOS's Polvi. "To run a container in a server environment, you need Linux, which means VMware now needs to manage above the hypervisor, actually into a Linux OS," he said. "This does not mean that VMware does not allow you to run CoreOS or Red Hat or any other operating system. It means they need a specialized one to add containers to VMware's product. They needed an extension of their hypervisor into an operating system."

Janakiram sees VMware as being forced to bring out its own OS out of fear that the shift from hypervisors to containers will erode their existing business.

> " VMware also knows there's more money to be made in the management layer of containers. They want to repeat the magic of the v-family of products with containers.

Red Hat Atomic Host

Red Hat launched RHEL 7 Atomic Host in March. It has fewer than 200 binaries, compared to the 6,500 in the full RHEL 7 release.

Mark Coggin, senior director of product marketing for Red Hat Enterprise Linux, called it a "happy medium" in the world of lean OSes: lighter than a traditional OS, but not as small as some of its competitors. He cautions against more "extreme" approaches, such as that of RancherOS, saying

that it could lead to more complexity, because it requires running additional system containers in addition to the application container.

"We've said, 'We think these are the core system services that need to be in the host platform that will address 80 to 90 percent of use cases for container applications,'" he said. "If a customer needs something beyond that — like system activity data collection, sys logging, identity — we have containerized versions of those as well, but we put much of what we think is necessary in the host platform.

He said Atomic Host brings to the table a base derived from an enterprise-grade operating system tested by Red Hat's engineering team. According to Red Hat, this is important because there are cross-dependencies that might not be readily apparent, say between the version of the Dynamic Host Configuration Protocol (DHCP) console or the version of Secure Shell (SSH) that you need inside your container, and the kernel.

"We've already gone through the work of ensuring that the version of SSH or DHCP that you're using is going to work with the version inside RHEL Atomic Host," Coggin said.

Red Hat is also working with independent software vendors (ISVs) to certify that the containers they build are based on images that are known, tested and safe.

Microsoft Nano Server

Nano Server, Microsoft's lightweight OS, targets two use cases, according to The New Stack's Scott M. Fulton III. The first is focused on supporting cloud infrastructure, such as clustered hypervisors and clustered storage. The second focuses on supporting cloud-native applications. In this latter

role, Nano Server will be suited to a new class of apps that get developed and deployed on Azure within a new Azure-based development environment — outside of the conventional client-based Visual Studio.

It's these new apps which will serve as Windows developers' entryway to the world of containers.

Developers writing for Nano Server will be guaranteed compatibility with pre-existing Windows Server installations, because Nano Server is effectively a subset of Windows Server. There may be a significant adjustment period, however, until developers become accustomed to the concept of microservices. Windows developers are used to having large libraries of pre-existing functionality available to their code in a global scope. Apps written to Nano Server run on a physical host, a virtual host or in a container. Two types of containers work on both Windows Server and Nano Server: the same Docker containers developed for Linux, and a type developed by Microsoft for its own hypervisor platform, called Hyper-V Containers.

"These provide additional isolation," explained Jeffrey Snover, Microsoft distinguished engineer and lead architect, in The New Stack. "They're really used for things like multi-tenant services, or multi-tenant platform-as-a-service, where you're going to be running code that might be malicious that you don't trust."

The concept draws inspiration from Drawbridge, a containerization system developed by Microsoft Research, mainly for purposes of process isolation and sandboxing untrusted apps that could crash the system.

The Swirling Market

It's not surprising to see so many vendors jockeying for position. Two camps are forming, Janakiram said: Red Hat, Docker and their allies on one side; and VMware, CoreOS and their allies on the other. "The more Red Hat goes closer to Docker, VMware will go farther."

Google, he said, is on the fence. "Google is smiling because they know how to do containers very well," he said.

"Kubernetes will work with rkt and they're working to make it work with any other container. But they want to make money on the public cloud; they want to run containers on the Google cloud. But they don't mind giving away some of their innovation on orchestration."

Microsoft, while closely aligned with Docker, has an uphill battle ahead of it as it works to overcome incompatibilities between Windows and Linux. "In 24 months the dust will settle, and we'll get to see who's the winner," Janakiram said. "Who's still relevant in the market."

ADOPTING CONTAINERS IN ENTERPRISE

by VIVEK JUNEJA
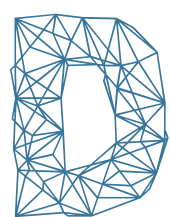
Disruptive new technologies and practices often follow an exhaustive adoption process in large enterprises. The process typically begins with elaborate presentations and discussions, followed by various proofs-of-concept. When adoption does occur, it typically occurs first within small teams building non-critical systems, and mostly for development and test workloads. By the time the technology reaches production, a significant amount of time and opportunity has passed.

Containers, and the notion of containerizing enterprise workloads, are going through this process right now, in a great many enterprises, fueled by the promise that containers offer to address the varied issues that plague developer productivity and application delivery in the enterprise.

Enterprise Motivations

By providing abstraction around the workload, and making it portable, containers become the foundation for supporting a variety of more

efficient development processes and application architectures. Specifically, enterprise development teams and IT organizations see in Docker and containers a means of addressing:

1. **Process inefficiencies:** Different teams within the enterprise often have their own standards for developing and releasing software. Outsourced development often exacerbates this problem. Many enterprises are exploring containers as a way to standardize the software release process across teams.

2. **Legacy applications:** Legacy applications and systems are commonplace in the enterprise. These systems create constant

FIG 1: *While the point-of-view is decidedly VMware-centric, a recent survey by StackEngine and vmblog.com showed pressure from development and test ranked highly as motivations for considering Docker.*

Motivations to Consider Docker

Q: For VMware users considering Docker, what's motivating you? (check all that apply)

Hybrid Cloud (cloud interoperability, private/public)	45%
VMware Cost	44%
Pressure From Test	42%
VMware Independence	36%
Performance Benefits	23%
Pressure From Development	23%
Consolidation Benefits	12%

0   10%   20%   30%   40%   50%
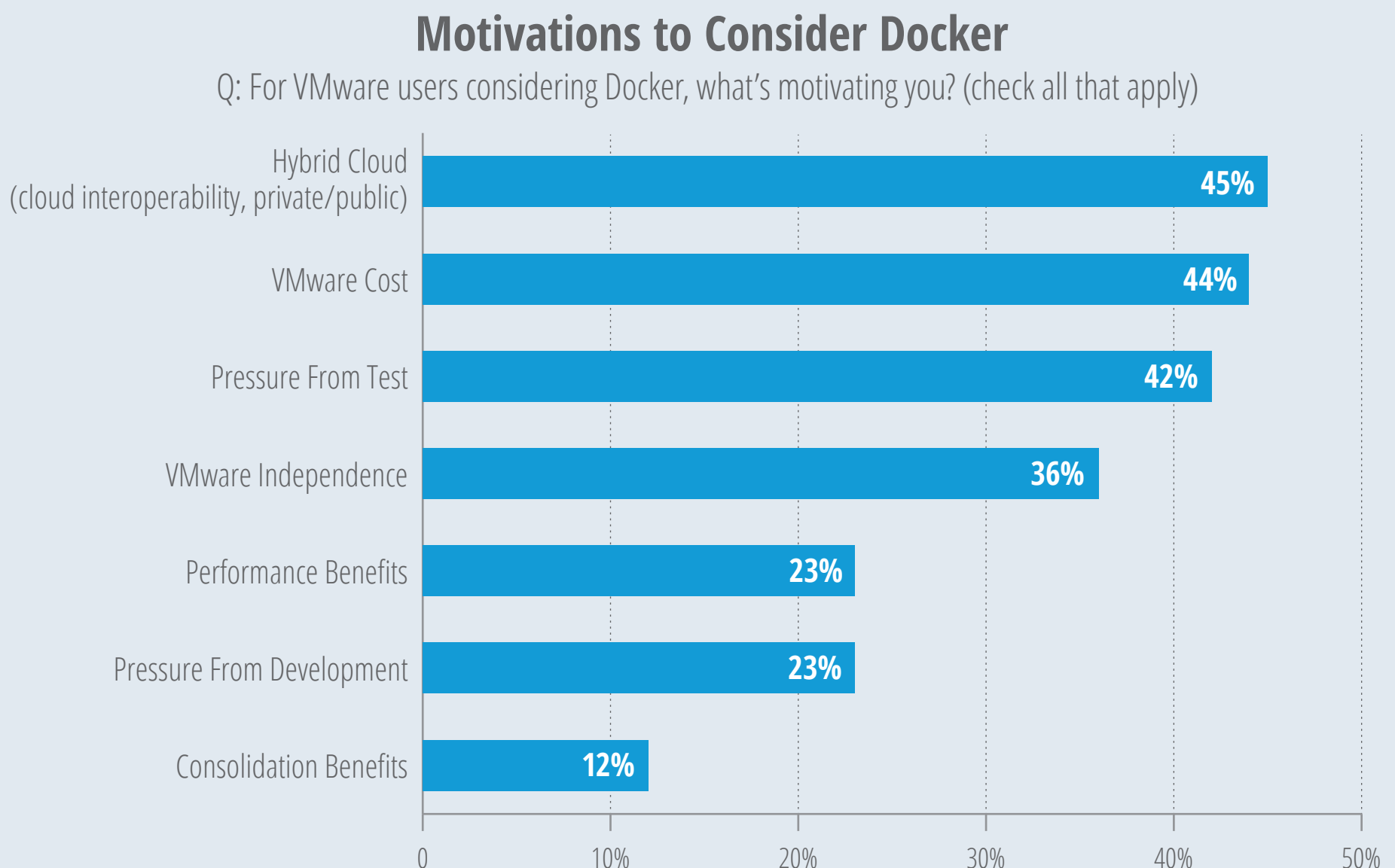
operational and maintenance issues. Project teams struggle to allocate resources to operate on-demand test infrastructure, and as a result frequently forgo testing in favor of releasing new functionality on time. Teams working on legacy applications are attracted to containers for their efficient use of infrastructure.

3. **Collaboration:** Development teams working on large projects have to coordinate software releases, a process that often takes a long time. Teams working on continuously maintained and developed legacy applications, for example, appreciate the ability of containers to help them synchronize software releases more easily.

   A development approach based on container images also helps encourage closer interaction between the development and operations teams, thereby encouraging DevOps.

4. **Dev/prod parity:** Production, development and test environments often have parity issues, drifting apart in ways that lead to constant "runs-on-my-machine" issues. Containers help ensure a consistent runtime environment across various infrastructures, a feature enterprise development teams find extremely useful.

5. **VM sprawl:** As enterprises adopt virtual machines, VM sprawl often sets in, leading to less efficient utilization of the infrastructure. Enterprises often respond by establishing strict governance rules, like approvals and workflows, to limit the sprawl. These rules, however, reverse the benefits of elasticity and self-service for developers. Because the overhead associated with containers is so much lower than that of VMs, many enterprises are excited about containers as a way to mitigate sprawl.

6. **Cloud-native applications:** Microservices and other cloud-native application architectures require a different view of infrastructure than is traditionally assumed. In containers, enterprise development teams see an opportunity to more easily build cloud-native applications and take advantage of emerging trends.

Container Adoption Strategies

When virtual machines started getting popular inside the enterprise, a major selling point to the IT department was the opportunity to consolidate underutilized infrastructure to reduce its operational footprint and cost. Adoption of containers and the ecosystem surrounding them, however, is being driven by agility, not cost reduction.

There are a handful of strategies that enterprises commonly take when evaluating and adopting container technologies:

1. **Go after low-hanging fruit:** When development teams have to struggle to get test environments provisioned and operational, they lose a lot of time, and frustration mounts. Solving infrastructure availability crunches in development and testing is often a good driver for making the leap to containers.

   For projects that don't require dedicated instances, reusing IT infrastructure for test environments deployed as linked containers is a great way to get started, and can provide important operational knowledge of the technology.

2. **Update build and deploy processes:** Build and deployment infrastructure needs to be modified for the enterprise to take full advantage of modern infrastructure. To deploy to public and private

PaaS Most Likely to be Automated by Container-Related Vendors

Q: What are you automating, replacing or supplementing in your customers' technology stack?

Category	Percentage
Platform as a Service	32%
Infrastructure as a Service	29%
Provisioning & Configuration Management Tool	29%
Hypervisors & Containers	17%
Cloud Operating System	12%
Software-Defined Networking, Network Functions Virtualization	10%
Storage	10%

■ Parts of Technology Stack Being Automated by Company's Product/Service
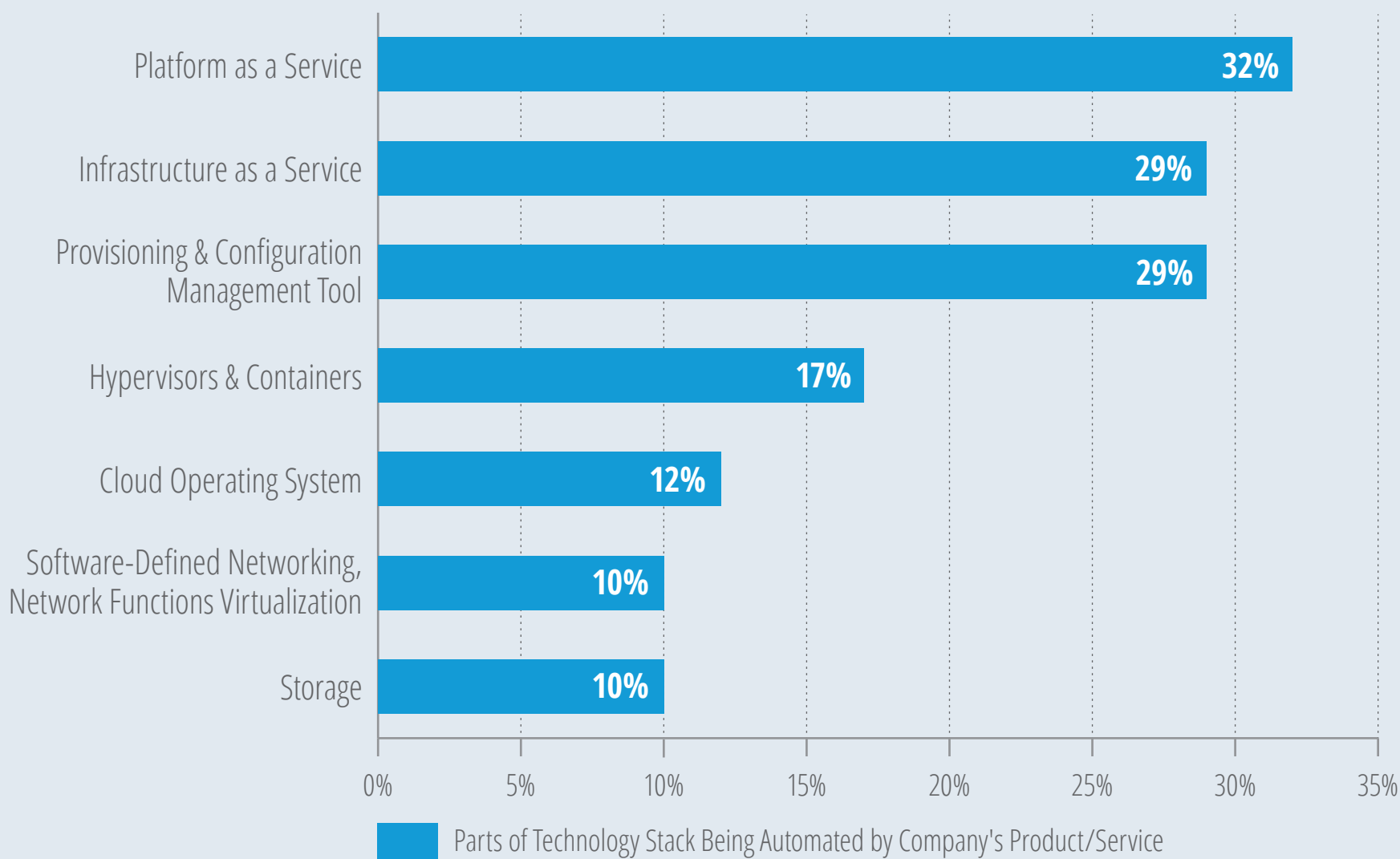
thenewstack.io

FIG 2: *Nearly a third of container ecosystem vendors responding to a TNS survey plan to automate PaaS, IaaS or other provisioning systems in the enterprise.*

clouds, some enterprises deploy the end application as a set of machine images, reducing the time required to set up newly provisioned on-demand infrastructure. The same practice can be extended to container images.

In a container-based deployment model, the build step can generate the new container image using a pre-existing base image for the environment. The deployment step can take this image and run it on any infrastructure supporting the chosen container technology.

3. **Go "container first":** Adopting a container first approach for all new projects is another common way to drive adoption. This means that all

new projects must build and release software with containers, unless there are specific reasons why they cannot. Going container first encourages development teams to consider containers a first-class element of their application topology and spurs the development of container-native applications. In addition, ramping up new development teams with pre-containerized project environments is far easier than with traditional approaches. This helps get teams started quickly, without the complexity of transitioning existing project processes.

4. **Standardize base images:** It's important for operations teams to formalize and release standard container images that all projects can use. These customized base images can be hosted on a private registry used by development teams when building projects. Changes to the standardized base images could mean new releases on the registry, which can then be transparently used in the development process. Enterprises are accustomed to hosting private repositories for firewall rules and other digital IT assets, so this should not be a strange adoption step for them.

Key Issues Faced

Enterprises face two key challenges when incorporating containers into their overall IT strategy: security and a lack of mature tools.

While web-scale companies like Google and Twitter have been using containers in production for years, it's still early days for both open source and proprietary products. The issue of tool maturity is largely one of time and experience. With more adoption and demand, the tools will inevitably stabilize and improve.

> " Enterprise IT organizations are looking to containers to help them achieve more modern application architectures and development processes, in the process allowing them to innovate more quickly.

Security, on the other hand, has been a dominant focus among early adopters. While a concern for all firms using containers in production, it is an even greater worry for companies using containers with production workloads in multi-tenant environments. The issue at hand is one of container isolation. To date, containers don't exhibit the strong and tested ability to isolate disparate workloads that has become assumed of the various virtual machine hypervisors.

Last year, Canonical introduced LXD, which implements ideas around stronger container security and makes them available via Ubuntu and OpenStack integration. New Projects like Hyper are also emerging as interesting alternatives to containers alone. And VMware has been promoting traditional virtualization as a complement to containers for this very reason.

Containers in Practice

As organizations continue to drive agility and innovation, they naturally aim to eliminate inefficiencies, such as resource constraints and process bottlenecks. At the same time, they are turning to modern application

architecture styles like microservices, and development processes like continuous delivery, which together allow developers to iterate much more quickly.

Last year, the chief archhitect of ING spoke at Dockercon Europe about how they are using Docker to enable continuous delivery. BBC News also highlighted how it architected its continuous integration solution around Docker, and shared the caveats and compromises that they had to deal with in doing so. The presentations by these two large companies is a sign of how quickly the enterprise is getting serious about container technologies.

By taking one or more of the approaches outlined in this article, enterprises new to the container game can take their first steps down the path of enlightenment, knowing that they're following a trail that has been trod many times before.

CONTAINER ECOSYSTEM DIRECTORY

DEVELOPER TOOLS, APPLICATION DEVELOPMENT/ DEPLOYMENT AND IMAGE CREATION

	Product/Project (Company or Supporting Org.)	Also categorized under:
	Artifactory (JFrog)	
	JFrog provides software developers with a binary repository management solution that integrates into CI/CD pipelines.	
	Atlas (Hashicorp)	
	Atlas unites Hashicorp development and infrastructure management tools to create a version control system for infrastructure.	
	Bitnami (Bitnami)	
	Bitnami is a library of server applications and development environments that can be installed with one click. Bitnami is beta testing functionality that will allow users to create container images.	
Open Source	**Brooklyn (Apache Foundation)**	» Orchestration and Management
	Apache Brooklyn is a library and control plane for deploying and managing distributed applications.	
Open Source	**Centurion (New Relic)**	
	Centurion is a deployment tool for Docker. It ships containers to and from a registry, running them on a fleet of hosts.	
Open Source	**Chronos (Apache Foundation)**	» Orchestration and Management
	Chronos is a distributed and fault-tolerant scheduler that runs on top of Mesos that can be used for job orchestration. Chronos is natively able to schedule jobs that run inside Docker containers.	
Open Source	**Cloud Foundry Lattice (Cloud Foundry Foundation)**	
	Lattice is an open source project for running containerized workloads on a cluster. Lattice bundles up HTTP load balancing, a cluster scheduler, log aggregation and streaming and health management into an easy-to-deploy and easy-to-use package.	
	CloudBees Jenkins Platform (CloudBees)	
	CloudBees Jenkins Platform is a PaaS that supports continuous integration and delivery of mobile and web applications. It is an enterprise version of Jenkins that has multiple plugins to support Docker.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
Open Source	**Cloudbreak (Hortonworks)** Cloudbreak helps users launch on-demand Hadoop clusters in the cloud or to any environment that supports Docker containers.	
Open Source	**CloudSlang (Hewlett-Packard)** CloudSlang is a flow-based orchestration tool for managing deployed applications. It aims to automate DevOps workflows and offers Docker capabilities and integrations.	
Open Source	**Codefresh (Codefresh)** Codefresh combines Docker tools with a web IDE based on Eclipse's Orion. The result is what they call a complete development environment for Node.js with DevOps and QA teams.	
Open Source	**Codenvy (Codenvy)** CloudSlang is a flow-based orchestration tool for managing deployed applications. It aims to automate DevOps workflows and offers Docker capabilities and integrations.	
	Codeship (Codeship) Codeship provides a hosted continuous delivery platform for web applications. It can be used to test Docker apps on different operating systems.	
	ContainerShip (ContainerShip) ContainerShip is a self-hosted container management platform, capable of running on any cloud, used to manage containers from development to production.	» Runtimes, Platforms and Hosts
	DCHQ (DCHQ) DCHQ is a deployment automation and governance platform for Docker-based application development. The solution provides self-service access to Docker-based applications using an agent-based architecture for orchestration.	» Orchestration and Management
Open Source	**Decking.io (Full Fat Finch)** Decking aims to simplify three core areas: building images based on local Dockerfiles, creating containers and orchestration of containers. It originally was an alternative to Fig, now Docker Compose.	
Open Source	**Docker Compose (Docker)** Compose is a tool for defining and running multi-container applications with Docker.	
Open Source	**Docker Hub (Docker)** Docker Hub is a cloud-based registry service for building and shipping application or service containers. It provides a centralized resource for container image discovery, distribution and change management.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	Docker Image Build Service (IBM)	
	A cloud-hosted build service for the compilation of Dockerfiles into Docker images on IBM Bluemix. Images are published to a private Docker registry on Bluemix.	
Open Source	**Docker Machine** (Docker)	
	Docker Machine lets you create Docker hosts on your computer.	
Open Source	**dockersh** (Yelp)	
	Dockersh is a login shell for machines with multiple users; it gives access to multiple users but allows for isolation between them.	
	Drone (Drone)	
	Drone is a continuous integration system built on top of Docker. Drone uses Docker containers to provision isolated testing environments.	
Open Source	**Dusty** (GameChanger Media)	
	Dusty is a development environment that provides OS X support for containers. Their documentation compares the tool to Vagrant and says it uses Docker Compose to orchestrate containers.	
Open Source	**Fabric8** (Red Hat)	
	Fabric8 is an open source DevOps and integration platform that is built as a set of microservices that run on top of Kubernetes and OpenShift V3.	
Open Source	**Ferry** (N/A)	
	Ferry is a big data development environment. It lets you define, run and deploy big data applications on AWS, OpenStack and your local machine using Docker.	
	Flockport Apps (Flockport)	» Image Registry and Security
	Flockport is a Linux container-sharing website. It also provides tools that make it easier to install and use LXC containers.	
	Harbormaster (Crane Software)	
	Crane's main product is Harbormaster, which is a Docker Release Management platform. It focuses on helping DevOps teams build, deploy and manage containers in production.	
	Hortonworks Data Platform (Hortonworks)	
	Hortonworks Data Platform is a commerical Hadoop offering. With its 2015 purchase of SequenceIQ, Hortonworks acquired Cloudbreak's ability to launch on-demand Hadoop clusters in the cloud or to any environment that supports Docker containers.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
Open Source	**Lorry (CenturyLink)**	
	Lorry is a CenturyLink Cloud utility and open source project that provides a graphical user interface for Docker Compose YAML validation and composition.	
Open Source	**Mantl (Cisco)**	
	Mantl is an open source platform for rapidly deploying globally distributed services. It works with tools such as Marathon, Mesos, Docker and Consul.	
Open Source	**Micro (MYODC)**	
	Micro is a microservices toolchain consisting of a suite of libraries and tools to write and run microservices.	
Open Source	**Packer (Hashicorp)**	
	Packer is a tool for creating machine and container images for multiple platforms from a single source configuration.	
Open Source	**Panamax (CenturyLink)**	
	Panamax is a containerized app creator with an open source app marketplace hosted on GitHub. Panamax provides an interface for users of Docker, Fleet and CoreOS.	
Open Source	**Powerstrip (ClusterHQ)**	
	Powerstrip is a tool for prototyping Docker extensions.	
Open Source	**runC (Linux Foundation)**	» Runtimes, Platforms and Hosts
	runC is a CLI tool for creating and running containers according to the Open Container Initiative's specification. It is the result of a multi-company coalition and is based on Docker's libcontainer project.	
	Runnable (Runnable)	
	Runnable works with GitHub and other tools to automatically deploy commits and launch containers in your sandbox when branches are created.	
	Shippable CI/CD (Shippable)	
	Shippable is a continuous integration platform built natively on Docker and using Docker Hub to deploy.	
	Shippable for OpenShift (Shippable)	
	Shippable's continuous integration platform is now available natively on OpenShift in beta.	
Open Source	**Shutit (N/A)**	
	Shutit is a tool for managing the Docker image building process; it expands on some of the capabilities of Dockerfiles.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	Spoon (Spoon)	
	Spoon is a platform for building, testing, deploying Windows applications & services in isolated containers.	
	StackDock (Copper.io)	» Infrastructure Services
	Copper.io is a full stack developer toolset. They produce StackDock, which helps deploy containers. In addition, it is developing storage and backup functionality for StackDock.	
	StackEngine (StackEngine)	» Orchestration and Management
	StackEngine is an end-to-end container application management system that provides a way for dev and enterprise IT teams to deploy Docker applications.	
Open Source	**Terraform (Hashicorp)**	
	Terraform is a tool to build and launch infrastructure, including containers.	
Open Source	**Totem (N/A)**	
	Totem is a continuous delivery pipeline tool designed for microservices.	
	Travis CI (Travis CI)	
	Travis CI is an open source continuous deployment platform; it is able to run on Docker-based infrastructures.	
	UrbanCode Build (IBM)	
	UrbanCode Build is a continuous integration and build management server optimized for the enterprise. It is designed to make it easy to scale the configuration and management of your build infrastructure and seamlessly plug into development, testing and release tooling. Supports Docker build and integration with Docker registries.	
Open Source	**Vessel (N/A)**	
	Vessel automates the setup and use of dockerized development environments. It is requires both OS X and Vagrant to work properly.	
	Virtuozzo (Odin)	
	Odin's Virtuozzo is a container virtualization platform sold to providers of cloud services.	
	Weave Run (Weaveworks)	
	Weave Run provides routing and control for containers, implemented as microservices.	
	Wercker (Wercker)	
	Wercker is a platform for automating the creation and deployment of applications and microservices.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	xDock (Xervmon)	» Orchestration and Management
	Xervmon is a cloud management platform. Its xDock lets users deploy, manage and monitor Docker images in the cloud.	
Open Source	**Zodiac (CenturyLink)**	
	Zodiac is a lightweight tool, built on top of Docker Compose, for easy deployment and rollback of "Dockerized" applications.	

RUNTIMES, PLATFORMS AND HOSTS

	Product/Project (Company or Supporting Org.)	Also categorized under:
	Apcera Hybrid Cloud OS (Apcera)	» Orchestration and Management
	Hybrid Cloud OS (HCOS) manages access to compute resources across a cluster of servers. By focusing on managing policies across multiple environments, it aims to secure workloads and containers in enterprise production environments.	
	Apprenda (Apprenda)	
	Apprenda provides a PaaS for enterprises that supports the hosting of containers.	
	Azure Container Service (Microsoft)	» Orchestration and Management
	Azure Container Service simplifies the creation and configuration of a cluster. The default configuration of this cluster includes Docker and Docker Swarm for code portability and Marathon, Chronos and Apache Mesos to ensure scalability.	
	Bluemix (IBM)	
	Bluemix is a PaaS for creating, deploying and managing applications in the cloud. As part of Bluemix, IBM Containers allows you to run Docker containers in a hosted cloud environment.	
Open Source	**Boot2Docker (N/A)**	
	Boot2Docker is a lightweight Linux distribution made specifically to run Docker containers. It is intended to be used with Docker Machine.	
Open Source	**Cloud Foundry Diego (Cloud Foundry Foundation)**	
	Diego is Cloud Foundry's next generation runtime. It is made up of a variety of small components, each with its own repository. It supports the scheduling and monitoring of Docker workloads.	
	cloudControl (cloudControl GmbH)	
	cloudControl offers a PaaS to facilitate development and scaling applications in the cloud. cloudControl acquired dotCloud from Docker in 2014.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	ContainerShip (ContainerShip)	» Developer Tools, App Development...
	ContainerShip is a self-hosted container management platform, capable of running on any cloud, used to manage containers from development to production.	
	CoreOS (CoreOS)	
	CoreOS is a lightweight OS based on the Linux kernel and designed for providing infrastructure to clustered deployments, while focusing on automation, ease of applications deployment to containers, security, reliability and scalability.	
	DaoCloud (DaoCloud)	
	DaoCloud is a China-based cloud computing company focusing on providing Docker services.	
Open Source	**Deis** (Engine Yard)	
	Deis is a lightweight, open source PaaS, built on Docker and CoreOS, that makes it easy to deploy and manage applications on your own servers.	
	DigitalOcean (DigitalOcean)	
	DigitalOcean is an IaaS provider that targets developers. It provides "one-click installs" to deploy Docker.	
Open Source	**Docker (Engine)** (Docker)	
	Docker Engine is a lightweight runtime and tooling that builds and runs your Docker containers.	
	Docker Subscription (Docker)	» Image Registry and Security
	Docker's subscription model includes commercially supported Docker engines for the servers running your application and a commercial registry service (Docker Trusted Registry or Docker Hub) of your choice.	
Open Source	**Dokku** (Engine Yard)	
	Dokku is a mini-PaaS that is inspired by Heroku's workflow, and is now sponsored through Engine Yard's Deis.	
	EC2 Container Service (Amazon Web Services)	
	Amazon EC2 Container Service helps companies manage clusters of containers on AWS infrastructure.	
Open Source	**Flynn** (Prime Directive, Inc.)	
	Flynn is a micro-PaaS built on Docker containers and optimized for service-oriented applications.	
	Giant Swarm (Giant Swarm)	» Image Registry and Security
	Giant Swarm is a hosted container solution to build, deploy and manage containerized services.	
	Google Container Engine (Google)	
	Google Container Engine is cluster management and orchestration system that lets users run containers on the Google Cloud Platform.	

Product/Project (Company or Supporting Org.)	Also categorized under:
Helion (Hewlett-Packard)	
Helion is Hewlett-Packard's hybrid cloud offering that includes a PaaS for developers. HP recently acquired Stackato from ActiveState and plans to integrate it into future offerings.	
Heroku Buildpacks (Salesforce)	
The Heroku PaaS provides "buildpacks" for compiling applications. Customers can pay for managed containers that Heroku calls "Dynos."	
Hyper (HyperHQ)	
Hyper is a hypervisor-agnostic Docker runtime engine. Its goal is to make VMs run like containers by eliminating the need for a guest OS.	
IBM Containers on Bluemix (IBM)	
Use IBM Containers to run Docker containers in a hosted cloud environment on IBM Bluemix. IBM Containers provides full hosting and lifecycle management of Docker containers, along with automatic and integrated log analytics and monitoring, elastic scaling with auto-recovery, reliability tools, load balancing and routing, persistent storage and security services. IBM Containers also provides a private image registry, images for WebSphere Liberty and Node.js, as well as support to use private images and images from Docker Hub.	
Jelastic (Jelastic)	» Orchestration and Management
Jelastic provides a PaaS and container-based IaaS on a singular platform that includes container orchestration.	
Joyent Triton Elastic Container (Joyent)	
Triton Elastic Container Service allows you to securely deploy and operate containers with bare metal speed on container-native infrastructure; Joyent provides the Triton Elastic Container service as part of its larger IaaS offerings.	
Kyup Cloud Hosting (Kyup)	
Kyup provides cloud-based container hosting.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
Open Source	**LXD** (Canonical)	
	LXD is a hypervisor for Linux containers, composed of three components: a system-wide daemon, a command-line client and an OpenStack Nova plugin.	
Open Source	**OpenShift Origin** (Red Hat)	
	OpenShift Origin is the upstream open source version of OpenShift and is meant to allow for development of cloud-native applications. OpenShift is a PaaS built on Docker containers that orchestrates with Kubernetes. It also has Atomic and Red Hat Linux components.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	Packet (Packet Host)	
	Packet is a bare metal PaaS that supports Docker, CoreOS, Deis, Mesosphere and Rancher.	
	Pivotal Cloud Foundry (Pivotal Software)	
	Pivotal Cloud Foundry is a cloud-native enterprise PaaS based on Cloud Foundry. It offers support for building applications as deployable containers.	
Open Source	**Project Photon** (VMware)	
	Photon is a Linux container host created by VMware as an open source project; it is focused on running containerized applications in a virtualized environment.	
Open Source	**Project Atomic** (Red Hat)	
	Project Atomic hosts run applications in Docker containers with components based on RHEL, Fedora and CentOS. In addition to Atomic Host, the project includes Nulecule, a container-based app spec that enables the use of existing containers as building blocks for new applications.	
	PureApplication (IBM)	
	PureApplication offers both on- and off-premises solutions that use automated patterns to rapidly deploy applications, reduce cost and complexity, and ease management. Docker containers can be included in patterns along with non-container components to take advantage of the PureApplication enterprise-grade lifecycle management. Support includes a private Docker registry pattern deployable as a shared service.	
	Quay.io (CoreOS)	
	Quay.io provides secure hosting for private Docker repositories.	
	Rancher (Rancher)	
	Rancher is a complete infrastructure platform for running containers in production.	
Open Source	**RancherOS** (Rancher)	
	RancherOS is a 20MB Linux distro that runs the entire OS as Docker containers.	
Open Source	**runC** (Linux Foundation)	» Developer Tools, App Development...
	runC is a CLI tool for creating and running containers according to the Open Container Initiative's specification. It is the result of a multi-company coalition and is based on Docker's libcontainer project.	
	Scalingo (Scalingo)	
	Scalingo is a PaaS for containers; users push their code to Scalingo and it creates an image and allocates resources to run the application in its cloud.	

Product/Project (Company or Supporting Org.)	Also categorized under:
Snappy Ubuntu Core (Canonical)	
Snappy Ubuntu Core is a new version of Ubuntu with a minimal server image and the same libraries, but applications are provided through a simpler mechanism.	
Tutum (Tutum)	» Image Registry and Security
As of October 2015, Tutum is still in beta. Tutum automates the build, test, deployment and management of containerized applications. It also has a free private registry to store Docker images.	
WaveMaker (WaveMaker)	
WaveMaker provides a PaaS for development and management of custom enterprise apps on private infrastructure. It supports the running of Docker applications.	
Windows Server Container (Microsoft)	
Microsoft is working with Docker to ensure that Windows applications can be run on Docker containers.	

ORCHESTRATION AND MANAGEMENT

Product/Project (Company or Supporting Org.)	Also categorized under:
Ansible Tower (Ansible)	
Ansible is a configuration management tool that can orchestrate the deployment of Docker containers. In addition, it lets users create container images.	
Apcera Hybrid Cloud OS (Apcera)	» Runtimes, Platforms and Hosts
Hybrid Cloud OS (HCOS) manages access to compute resources across a cluster of servers. By focusing on managing policies across multiple environments, it aims to secure workloads and containers in enterprise production environments.	
Azure Container Service (Microsoft)	» Runtimes, Platforms and Hosts
Azure Container Service simplifies the creation and configuration of a cluster. The default configuration of this cluster includes Docker and Docker Swarm for code portability and Marathon, Chronos and Apache Mesos to ensure scalability.	
BanyanOps (BanyanOps)	
BanyanOps launched in 2015 and does not yet have a product. It is focused on analyzing images and wants to accelerate IT operations with containers.	
Open Source **BOSH (Cloud Foundry Foundation)**	
BOSH is an open source toolchain for orchestration of large-scale distributed services that can be used with Docker containers. BOSH installs and updates software packages on large numbers of VMs over many IaaS providers.	
Open Source **Brooklyn (Apache Foundation)**	» Developer Tools, App Development…
Apache Brooklyn is a library and control plane for deploying and managing distributed applications.	
Open Source **cAdvisor (Google)**	
cAdvisor (Container Advisor) is a Google-support project that analyzes resource usage and performance characteristics of running containers.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	Chef (Chef)	
	Chef is a configuration management and automation platform. It can be used to create, manage and provision Docker containers.	
Open Source	**Chronos** (Apache Foundation)	» Developer Tools, App Development…
	Chronos is a distributed and fault-tolerant scheduler that runs on top of Mesos that can be used for job orchestration. Chronos is natively able to schedule jobs that run inside Docker containers.	
	Cloud 66 (Cloud 66)	
	Cloud 66 is an application provisioning and management service that allows you to build Docker stacks from scratch on any public or private cloud vendor or your own infrastructure.	
Open Source	**Cloud Foundry Containers Service Broker** (Cloud Foundry Foundation)	
	This is a container services broker for the Cloud Foundry v2 services API. It allows for the provisioning of services and binding of applications to a container backend.	
Open Source	**Cloud Foundry Lattice** (Cloud Foundry Foundation)	» Developer Tools, App Development…
	Lattice is an open source project for running containerized workloads on a cluster. Lattice bundles up HTTP load balancing, a cluster scheduler, log aggregation and streaming and health management into an easy-to-deploy and easy-to-use package.	
Open Source	**CloudSlang** (Hewlett-Packard)	» Developer Tools, App Development…
	CloudSlang is a flow-based orchestration tool for managing deployed applications. It aims to automate DevOps workflows and offers Docker capabilities and integrations.	
	Cloudsoft Application Management Platform (Cloudsoft)	
	Cloudsoft's application management platform, based on the open source Apache Brooklyn project, orchestrates services, platforms and infrastructure, including deployment to containers.	
	ClusterUP (ClusterUP)	
	ClusterUp is providing a GUI to monitor and manage the lifecycle of Docker services.	
Open Source	**Consul** (Hashicorp)	
	Consul is a tool for service discovery and configuration. Consul is distributed, highly available and extremely scalable.	
Open Source	**Crane** (N/A)	
	Crane is a lightweight wrapper around the Docker CLI that is used to orchestrate Docker containers.	
	Datadog (Datadog)	
	Datadog is a monitoring and analytics service for IT operations and development teams. It has containerized agents that can monitor container environments.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	DCHQ (DCHQ)	» Developer Tools, App Development...
	DCHQ is a deployment automation and governance platform for Docker-based application development. The solution provides self-service access to Docker-based applications using an agent-based architecture for orchestration.	
	DCOS (Mesosphere)	
	Mesosphere's DCOS is a commercial version of the Mesos OS for managing data centers. It supports both Kubernetes and Docker.	
Open Source	**Docker Swarm (Docker)**	
	Docker Swarm offers native clustering for Docker by turning multiple Docker hosts into a single, virtual host.	
Open Source	**Dray (CenturyLink)**	
	Dray is an engine for managing the execution of container-based workflows. It is a Go application that provides a RESTful API for managing jobs and is most commonly used for containers hosting long-running services.	
	Elasticsearch (Elastic)	
	Elasticsearch is a search and analytics engine based on Lucene.	
	Engine Yard (Engine Yard)	
	Engine Yard is a cloud orchestration PaaS for deploying, monitoring and scaling applications.	
Open Source	**etcd (CoreOS)**	
	etcd is a distributed, consistent key-value store for shared configuration and service discovery.	
Open Source	**Fleet (CoreOS)**	
	Fleet is a distributed init system used to support cluster management and orchestration of containers.	
Open Source	**Flocker (ClusterHQ)**	
	Flocker is a data volume manager for Dockerized applications.	
	Found (Elastic)	
	Elastic's founder created Elasticsearch and they provide it as a service called "Found." It can be used by the Docker community for search and discovery.	
Open Source	**ImageLayers (CenturyLink)**	
	ImageLayers.io allows Docker users to easily discover best practices for image construction, and aids in determining which images are most appropriate for their specific use cases.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	IronMQ (Iron.io)	» Infrastructure Services
	IronMQ is a messaging queue that provides scheduling and communication between services and components.	
	IronWorker (Iron.io)	
	IronWorker is a platform that isolates the code and dependencies of individual tasks to be processed on demand in a containerized environment.	
	Jelastic (Jelastic)	» Runtimes, Platforms and Hosts
	Jelastic provides a PaaS and container-based IaaS on a singular platform that includes container orchestration.	
Open Source	**Kitematic (Docker)**	
	Kitematic is a graphic interface to manage Docker. The sponsoring company was bought by Docker, but the actual software is now part of Docker's toolkit.	
Open Source	**Kong (Mashape)**	
	Kong is a management layer for APIs. It has the capability of orchestrating Dockerfiles.	
Open Source	**Kontena (Kontena)**	
	Kontena is a container orchestration tool. It abstracts containers into application services and establishes an internal network between linked services, making it easy to deploy and scale applications across multiple hosts.	
Open Source	**Kubernetes (Google)**	
	Kubernetes is an open source Docker orchestration tool. Google initially developed Kubernetes to help manage its own LXC containers.	
	Logentries (Rapid7)	
	Logentries provides analytics tools to monitor Docker environments.	
Open Source	**MaestroNG (SignalFx)**	
	MaestroNG is an orchestrator for Docker-based, multi-host environments sponsored by SignalFx.	
Open Source	**Magnum (Open Stack Foundation)**	
	Magnum is an OpenStack project which offers container orchestration engines for deploying and managing containers as first class resources in OpenStack.	
Open Source	**Marathon (Apache Foundation)**	
	Marathon is an Apache Mesos framework for long-running applications. Marathon provides a REST API for starting, stopping and scaling applications. It lets users deploy, run and scale Docker containers.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
Open Source	**Mesos (Apache Foundation)**	
	Apache Mesos is a cluster manager that provides efficient resource isolation and sharing across distributed applications, or frameworks.	
	Nirmata (Nirmata)	
	Nirmata is a cloud-based platform for managing microservices.	
Open Source	**Percheron (N/A)**	
	Percheron is used to manage images and containers.	
Open Source	**Prometheus (Robust Perception)**	
	Prometheus is an open-source service monitoring system and time series database.	
	Puppet Enterprise (Puppet Labs)	
	Puppet Enterprise is a configuration management and automation tool, including built-in tools for building Dockerfiles.	
	SaltStack Enterprise (SaltStack)	
	SaltStack is an orchestration and automation tool that can be used to manage Docker containers.	
	Scout (Scout)	
	Scout provides application and server monitoring; it has plugins to monitor both Docker and LXC containers.	
	Shippable Formations (Shippable)	
	Shippable Formations is an orchestration tool for dev and test environments.	
Open Source	**Shipyard (N/A)**	
	Shipyard enables multi-host, Docker cluster management, and is fully compatible with the Docker Remote API.	
	StackEngine (StackEngine)	» Developer Tools, App Development...
	StackEngine is an end-to-end container application management system that provides a way for dev and enterprise IT teams to deploy Docker applications.	
	Sysdig Cloud (Sysdig Cloud)	
	Based on open source Sysdig technology, Sysdig Cloud monitors containerized environments.	
	Tectonic (CoreOS)	
	Tectonic, which is currently being previewed, will be an enterprise version of Kubernetes.	

Product/Project (Company or Supporting Org.)	Also categorized under:
UrbanCode Deploy (IBM) UrbanCode Deploy orchestrates the deployment of applications across environments, coordinating the deployment of many individual components with inventory tracking. This includes support for Docker Containers, Docker Registries, and IBM Container Service on Bluemix via a community plugin.	
Weave Scope (Weaveworks) Weave Scope offers a real-time monitoring solution for containers.	
xDock (Xervmon) Xervmon is a cloud management platform. Its xDock lets users deploy, manage and monitor Docker images in the cloud.	» Developer Tools, App Development…
Open Source **Zenoss Control Center** (Zenoss) Zenoss Control Center is an application management and orchestration system. It works with the Zenoss platform and other Docker applications. Serviced is a popular repository in this project that provides a PaaS runtime.	

INFRASTRUCTURE SERVICES

	Product/Project (Company or Supporting Org.)	Also categorized under:
Open Source	**Clocker (Apache Foundation)**	
	Clocker creates and manages a Docker cloud infrastructure. It is a part of the Apache Brooklyn projects, and has plugins for Project Calico and Weave.	
Open Source	**Crate (Crate.io)**	
	Crate uses SQL syntax for distributed queries across a cluster.	
	EMC Elastic Cloud Storage (EMC)	
	EMC's Elastic Cloud Storage (ECS) is available as a Docker container. In addition, the storage giant is working with ClusterHQ and its Flocker so that containers can access the underlying storage when deployed on top of bare metal.	
Open Source	**Flannel (CoreOS)**	
	Flannel is a virtual network for hosting containers.	
	IronMQ (Iron.io)	» Orchestration and Management
	IronMQ is a messaging queue that provides scheduling and communication between services and components.	
Open Source	**libnetwork (Docker)**	
	Libnetwork provides a native Go implementation for connecting containers.	
Open Source	**Pachyderm (Pachyderm)**	
	Pachyderm enables storage and analysis of data using containers.	
	Pertino (Pertino)	
	Pertino lets developers build container-level virtual private cloud networks.	
	Portworx PWX (Portworx)	
	Portworx PWX provides elastic scale-out block storage natively to Docker containers.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
Open Source	**Project Calico** **(Metaswitch Networks)**	» Image Registry and Security
	Project Calico provides networking for OpenStack VMs as well as containers in a Docker environment. Each container gets its own IP and security policy. Users of Clocker can use Calico with it.	
	SoftLayer **(IBM)**	
	SoftLayer provides infrastructure as a service (IaaS) including bare metal and virtual servers, networking, turnkey big data solutions, and private cloud solutions. SoftLayer is supported as a provider behind Docker Machine for quickly standing up a cloud-hosted Docker host.	
	StackDock **(Copper)**	» Developer Tools, App Development…
	Copper.io is a full stack developer toolset. They produce StackDock, which helps deploy containers. In addition, it is developing storage and backup functionality for StackDock.	
	VNS3:net **(Cohesive Networks)**	» Image Registry and Security
	Cohesive Networks' VNS3:net allows user to create a virtual network. This solution can provide increased control and security for containers.	
Open Source	**Weave** **(Weaveworks)**	
	Weave is a container-specific implementation of software-defined networking across data centers.	
	Weave Net **(Weaveworks)**	
	Weave Net connects containers into a transparent, dynamic and resilient mesh.	

IMAGE REGISTRY AND SECURITY

	Product/Project (Company or Supporting Org.)	Also categorized under:
	CloudPassage **(CloudPassage)**	
	CloudPassage offers a server security and compliance product purpose-built for elastic cloud environments.	
Open Source	**Docker Bench for Security** **(Docker)**	
	The Docker Bench for Security is a script that checks for dozens of common best practices around deploying Docker containers in production.	
	Docker Subscription **(Docker)**	» Runtimes, Platforms and Hosts
	Docker's subscription model includes commercially supported Docker engines for the servers running your application and a commercial registry service (Docker Trusted Registory or Docker Hub) of your choice.	
	Docker Trusted Registry **(Docker)**	
	Docker Trusted Registry allows users to store and manage Docker images on premise or in a virtual private cloud.	
	Enterprise Registry **(CoreOS)**	
	Enterprise Registry provides a secure registry on an enterprise's own infrastructure.	
	Flockport Apps **(Flockport)**	» Developer Tools, App Development...
	Flockport is a Linux container-sharing website. It also provides tools that make it easier to install and use LXC containers.	
	Giant Swarm **(Giant Swarm)**	» Runtimes, Platforms and Hosts
	Giant Swarm is a hosted container solution to build, deploy and manage containerized services.	
	Google Container Registry **(Google)**	
	Google Container Registry provides secure, private Docker image storage on Google Cloud Platform.	
Open Source	**Notary** **(Docker)**	
	The Notary project comprises a server and a client for running and interacting with trusted collections. Publishers can sign their content offline using highly secure keys.	

	Product/Project (Company or Supporting Org.)	Also categorized under:
	Polyverse (Polyverse)	
	Polyverse uses millions of individually protected containers to help prevent large-scale data breaches.	
Open Source	**Portus** (SUSE)	
	Portus acts both as an authoritzation server and as a user interface for Docker registry (v2).	
	Private Image Registry Service (IBM)	
	IBM Containers on Bluemix provides a private Docker image registry service for hosting private images. The private registry supports group access policies to allow teams to share private images.	
Open Source	**Project Calico** (Metaswitch Networks)	» Infrastructure Services
	Project Calico provides networking for OpenStack VMs as well as containers in a Docker environment. Each container gets its own IP and security policy. Users of Clocker can use Calico with it.	
	Reesd Images (Reesd)	
	Reesd is a private Docker repository and storage service.	
Open Source	**Registrator** (Glider Labs)	
	Registrator automatically registers and deregisters services for any Docker container. It supports pluggable service registries like Consul and etcd.	
	ScriptRock (ScriptRock)	
	ScriptRock validates the integrity of configurations in both test and production environments. It can be used as a platform for automatically sharing configuration tests with multiple tools, including Docker and Puppet.	
	Shipway (Shipway)	
	Shipway automatically discovers GitHub repositories and searches for Dockerfiles. Using a webhook, it automatically publishes new Docker images when you push your repository.	
	Tutum (Tutum)	» Runtimes, Platforms and Hosts
	As of October 2015, Tutum is still in beta. Tutum automates the build, test, deployment and management of containerized applications. It also has a free private registry to store Docker images.	
	Twistlock (Twistlock)	
	Twistlock provides a security framework for developers, allowing them to do security checks before pushing applications to production. It also gives security teams a centralized location to configure and monitor security rules across multiple container clusters.	

Product/Project (Company or Supporting Org.)	Also categorized under:
VNS3:net (Cohesive Networks) Cohesive Networks' VNS3:net allows user to create a virtual network. This solution can provide increased control and security for containers.	» Infrastructure Services
Vulnerability Advisory (IBM) Vulnerability Advisory is a capability of IBM Containers on Bluemix. It gives container developers a view into their image security properties and as well as guidance on how images should be improved to meet common sense best practices and upgrade to known industry fixes. Using Vulnerability Advisor, developers can design secure applications with very little effort.	

CONSULTING AND MISC.

Product/Project (Company or Supporting Org.)	Also categorized under:
Container Solutions (Container Solutions) Container Solutions is a consulting group that specializes in helping customers to shorten the time it takes to deploy.	
DaoClou2d (DaoCloud) DaoCloud is a China-based cloud computing company focusing on providing Docker services.	
IBM Resale of Docker Subscription (IBM) IBM resells Docker Subscription which includes the Docker Engine and Docker Trusted Registry as well as level 1 and level 2 support provided by IBM in North America, Europe, Australia and New Zealand.	
Jetstack Container Services (Jetstack) Jetstack is a consulting company focused on helping companies build a container management infrastructure.	
Kismatic (Kismatic) Kismatic provides enterprise support and production platform tooling for Kubernetes and Docker.	

DISCLOSURES

The following companies mentioned in this ebook are sponsors of The New Stack: ActiveState, Apcera, Apprenda, Basho, Cloudsoft, CoreOS, Datadog, DigitalOcean, Hewlett-Packard, Intel, Joyent, New Relic, Pivotal, ProfitBricks, Red Hat, SAP, Shippable, SignalFx, VMware and Weaveworks.

"Brigg 'the desired peace' of Trondheim crashes during storm, 1798" by the Municipal Archives of Trondheim is licensed under CC BY 2.0 and has been cropped from its original version.