O'REILLY®

Compliments of ruxit

The State of Containers and the Docker Ecosystem 2015

Anna Gerber

Monitor your Docker environment with Ruxit

Get deep end-to-end insights into your dockerized applications

Ruxit ...

- integrates seamlessly with no configuration
- monitors your microservices and container deployments
- tracks the dynamics of Mesos and Kubernetes environments

Try Ruxit for free!

ruxit

The State of Containers and the Docker Ecosystem: 2015

Anna Gerber

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

The State of Containers and the Docker Ecosystem: 2015
by Anna Gerber

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editor: Brian Anderson
Production Editor: Dan Fauxsmith
Interior Designer: David Futato
Cover Designer: Randy Comer
Illustrator: Rebecca Demarest

September 2015: First Edition

Revision History for the First Edition
20145-09-16: First Release

978-1-491-94138-6

[LSI]

Table of Contents

Foreword

When Docker launched in March of 2013, we had no idea how rapidly the project would capture the imagination of developers and operators alike. We have rapidly seen Docker, containerization, and microservices move into the mainstream. From our vantagepoint, we know that millions of developers and thousands of enterprises have adopted Docker. As of this writing, over 800 million Docker containers have been pulled from the public Docker Hub.

However, as with any technology this new, questions remain about the details behind this usage. This excellent survey provides insights into who is using Docker, how they are using it, what their environments look like, and what complementary technologies (orchestration, OS, networking, etc.) are part of the mix. The survey also delves into use cases and challenges, providing valuable detail to guide those of us in the ecosystem as we chart our future roadmaps and development priorities. While Docker is rapidly becoming part of the mainstream, the survey does an excellent job of highlighting the remaining challenges and opportunities in areas like scaling, clustering, scheduling, and availability. Whether you are already deploying Docker at scale in production using tools like Compose, Swarm, and Notary, or whether you are just experimenting with your first "Hello World" exercise, this survey should provide interesting, useful, and provocative information.

—Ben Golub,
CEO, Docker, Inc.

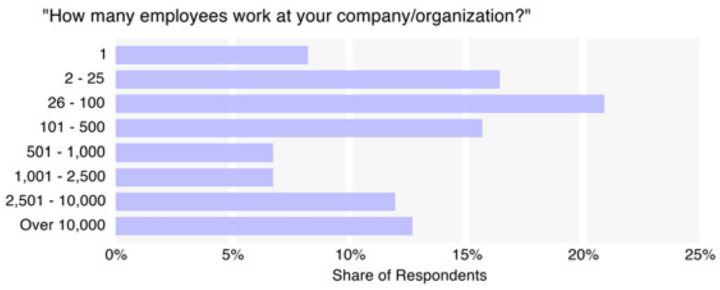
The State of Containers and the Docker Ecosystem: 2015

Introduction

Containers are one of the hottest topics in IT right now, but how are they actually being used? A survey conducted by O'Reilly Media in collaboration with Ruxit throughout May 2015 invited individuals from the O'Reilly community to share how their organizations currently use or plan to use containers. The results reveal that container technologies—and Docker in particular—are rapidly being adopted to make application deployment faster, easier, and more flexible.

The survey, which ran in the lead up to DockerCon 2015, aimed to discover how containers are being used and which container technologies and infrastructures are being adopted, as well as motivations and challenges associated with opting to use containers.

The 138 self-selected respondents came from a range of industries, including software, consulting, publishing and media, education, cloud services, hardware, retail, government, and others. They represent a spectrum of different sizes of companies or organizations, with about half of the respondents coming from organizations with fewer than 500 employees. Thirteen percent of responses are from individuals working at companies or organizations with over 10,000 employees.

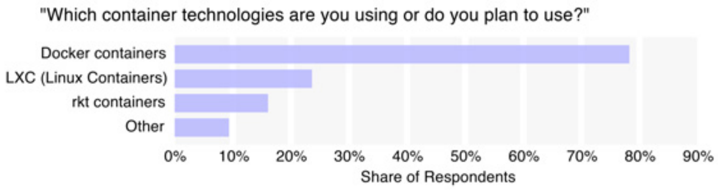


Container Uptake Is High

The results reveal that the adoption of container technologies is high, with 65% of respondents reporting that their organization currently uses containers. Containers are being adopted across organizations of all sizes; however, those organizations running ten or fewer hosts in their infrastructure are more likely not to be using container technologies at present than those with more than ten hosts.

Docker Is a Widely Adopted Container Technology

Reflecting the industry excitement around Docker, 78% of respondents are using or planning to use Docker, compared to 24% opting for the older and decidedly less trendy Linux containerization technology LXC, and 11% using or considering using other technologies such as Cloud Foundry's Warden or Microsoft's Hyper-V.



Although quite new to the game, 16% of respondents are already using or planning to use rkt, an open application container runtime from the CoreOS project, first released as a prototype in December 2014. rkt was developed with the aim of providing a small, independent, and composable tool for running containers, and supports running Docker containers as well as its own ACI (App Container Image) format.

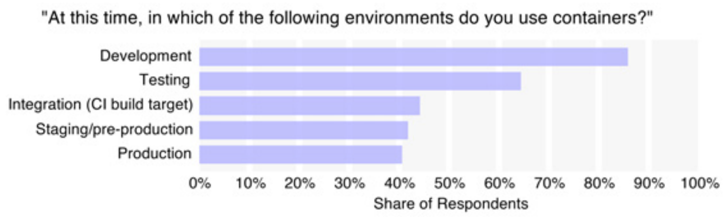
Both Docker and CoreOS are part of the newly formed Open Container Initiative (OCI) that was announced at DockerCon on June 22nd. Hosted by The Linux Foundation, OCI comprises a consortium of industry leaders, including Amazon, Google, Mesosphere, Pivotal, Cisco, IBM, Microsoft, Intel, RedHat, Oracle, Verizon, and others, who are working toward developing an open industry standard for container runtimes and containers based on Docker's container format. Docker also announced at DockerCon 2015 that it would be splitting out core pieces of infrastructure "plumbing" into smaller tools, starting with the lightweight, portable container runtime, runC. Docker has contributed runC to OCI to be used as a basis for the standardization process.

As rkt is still only in early stage release, it will be worth keeping an eye on how it impacts on Docker's share of the container market once it reaches a stable release. However, if OCI succeeds in its goal of achieving interoperability of containers across different platforms, tools, and providers, choice of container technology may become a minor detail, as it will be easier to migrate between or to create hybrid solutions by combining tools from the various container platforms.

Microsoft's recent preview release of Windows Server with support for managing Windows Server and Hyper-V Containers using the Docker CLI tools is one example of how industry partnerships are already making the process of deploying and managing containers more consistent across platforms.

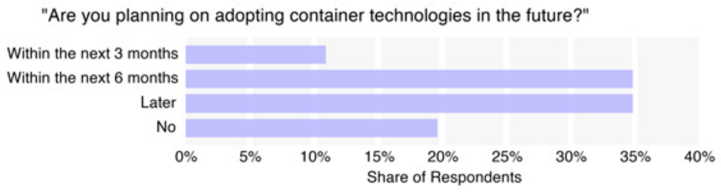
Lifecycle Adoption Starting to Pick Up in Production

The survey results indicate that containers are still working their way into the application development lifecycle, with container usage highest in the earlier phases of development and testing, while dropping off in integration, staging, and production environments: 86% of those respondents using containers are using them for development and 64% for testing, while only 40% are using containers in production.



Production Usage Projected to Increase

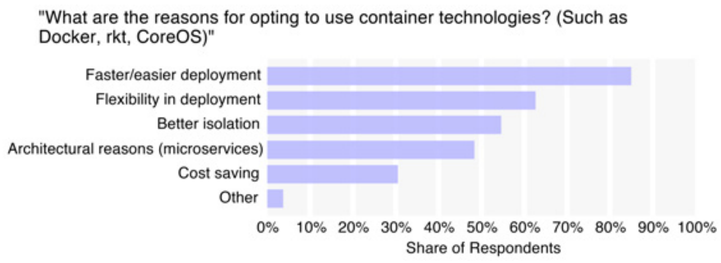
However, adoption of containers is rapidly on the rise: of those respondents who are not already using containers, over 80% intend to adopt containers, with 11% indicating that they plan to adopt containers within the next three months and a further 35% within the next six months.



Development and testing remain the main focus for future adoption of containers; however, more than half (53%) of all respondents indicated that they intend to adopt containers in production within the next 6–12 months, up from 40% at present. This upward trend indicates that containers are increasingly being considered production-ready.

Why Are Organizations Adopting Containers?

More than 93% of the survey respondents are already using or plan to use containers in the near future, and the majority (78%) of them are opting for Docker. So why are organizations moving to containers?



Fast and Easy Deployment

The main driver for moving to a containerized infrastructure, identified by 85% of respondents, is that containers make it faster and easier to deploy applications.

Speed is one of Docker's biggest selling points, and Docker's tools and portable image format are designed to make launching containers simple and reproducible across environments. Docker stores the images that are used to launch containers as a series of read-only versioned layers built up from a base image, representing a series of instructions to setup dependencies, copy application files, set up data volumes, and so on. As only the diffs are stored, and the layers are cached during the build process, it's very efficient to store and fast to build new derivative images, for example, to rapidly build images for different versions of an application.

Docker's portable image format allows images to be published to a registry to make distribution and deployment across environments easier. Within the DockerHub registry or a private image registry that you host yourself, you can create a repository of public or private images that are ready to deploy for each version of your application. Docker's image registries provide APIs to make it possible to automate pulling images from within your deployment scripts or tools. Keeping a repository of versioned ready-to-run images also makes it fast and easy to rollback to a previous version of an application.

Flexible Deployment

Sixty-two percent of respondents identified flexibility in deployment as a reason for choosing containers. Many popular cloud services providers, including Amazon, Google, Microsoft, IBM, VMWare, and Joyent, offer container services based around Docker containers. However, unlike many early PaaS solutions, Docker containers can also be run on any bare-metal or virtual host, locally or within a public or private cloud (and for multi-container applications, potentially distributed across multiple clouds), and from within a variety of Linux distributions, hence avoiding lock-in to any particular provider or platform. Support for some of the other container platforms and tools is not as widespread as for Docker, but with many of these vendors now participating in the OCI, compatibility of containers across hosted solutions and container platforms should continue to improve, providing even more options for deployment.

Isolation and Security

Better isolation is another reason for opting for containers, given by 54% of respondents. Containers are designed to isolate processes—they encapsulate the dependencies (including libraries and shared binaries) and configuration for an application. Containerization is sometimes called Operating-System-Level Virtualization: Each container runs on a shared host OS, but is isolated from the OS and any other containers running on the same host OS. Within Linux-based containerization platforms like Docker, LXC, and rkt, resource isolation for each container is implemented using control groups (cgroups) to control access to shared resources, including memory, CPU, and disk IO. Because application dependencies are contained and isolated, it is possible to run two containers with conflicting dependencies (e.g., different required versions of a library) side-by-side on the same host, and dependencies installed on the host server itself can be kept to a minimum.

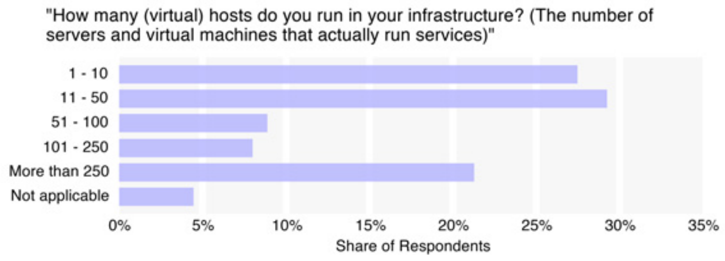
Microservices

Just under half (48%) of the respondents indicated that a desire to move to a microservices architecture was a motivating factor in their decision to adopt containers. Microservices are an approach to architecting scalable cloud computing applications, where complex applications are built up from small, discrete, modular services that each handle a single aspect of an application's business functionality. Adopting a microservices architecture is generally considered to be a better practice for scalable cloud applications than developing a monolithic application, because it breaks the development of complex cloud applications into smaller manageable pieces, and it it is possible for parts of the application to be independently upgraded, rather than having to re-deploy the entire application each time something needs to be changed.

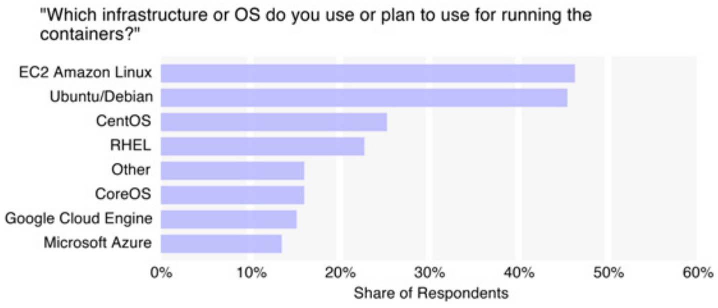
Containers are a great fit for implementing a microservices architecture, particularly for stateless microservices, where each microservice can be isolated to a single container, with services communicating only via lightweight APIs running on controlled ports. Microservices also have the advantage of being able to be independently scaled, and running the services inside containers helps to make efficient use of cloud resources as memory, I/O and processing resources can be precisely allocated, for example, using cgroups with Docker containers, to suit the requirements of the services running in each container, but with very little runtime overhead from the containers themselves. Additionally, microservices go hand-in-hand with Continuous Integration and Deployment (CI/CD), making it possible to respond to changing customer requirements and issues more rapidly, and thus improve the overall quality of the application.

Deployment Size

Respondents represented a range of different sizes of infrastructure, with the largest groups being those running 1–10 hosts (27%), 11–50 hosts (29%), and more than 250 hosts (21%).



When asked about the infrastructure or OS used to run containers, 46% of respondents indicated that they use or plan to use EC2 Amazon Linux, while 45% use or plan to use Ubuntu/Debian. Roughly a quarter of respondents selected CentOS and a similar number for RedHat Enterprise Linux. “Other” was also a popular option, reflecting the diversity of solutions available for running containers, and this category included RancherOS, OpenStack, CloudFoundry, Rackspace, OpenShift, BlueMix, and Microsoft Hyper-V, among others.



For the OS running within the containers themselves, Ubuntu/Debian was the most popular choice, with 67% of respondents using or planning to use it for their containers.

How Are Containers Impacting Infrastructure?

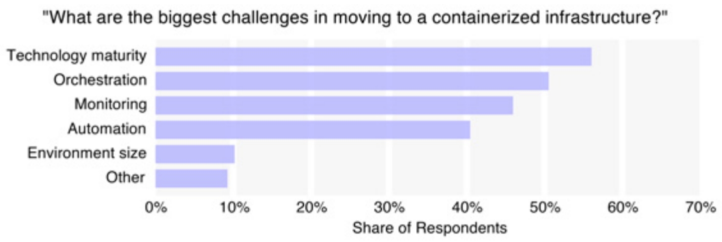
The survey also asked respondents to reflect on how their infrastructure has changed since adopting containers.

Mirroring the motivations for adopting containers, many respondents indicated that containerization has simplified and sped-up deployment, has made it easier and faster to test and iterate, and easier to rollback if required. Another change identified is that adopting containers improves operational management—it is easier to automate deployment and to integrate with DevOps tools. The end result is more frequent deployments.

By far the most common theme identified throughout the responses to how infrastructure has changed is that containerization has resulted in fewer hosts, because of the increased density of services running per host, due to multiple containers being run on each host. Some respondents indicated that decreasing the number of hosts by moving to containers has reduced operating costs. Cost Saving (see the graphic in the section “Why Are Organizations Adopting Containers?”) was identified by 30% of respondents as a motivating factor in adopting containers, so this is a promising observation. However, a few respondents pointed out that moving to containers introduced the overhead of running their own private container registries, and that containerization had resulted in their monitoring and alerting infrastructure changing significantly. So, for organizations running only a small number of services, containerization introduces complexity and may not actually reduce the number of hosts required or result in cost savings.

Challenges

Forty percent of respondents are currently using containers in production compared to 86% for development. Given that Docker containers are designed to "build once, run anywhere," why aren't containers being adopted more widely in production? Challenges to adopting containers identified by respondents include technology maturity, orchestration, monitoring, automation, and environment size, as well as other reasons, including the difficulty of convincing clients, management, or the development team of the benefits of adopting containers.



Technology Maturity a Barrier to Adoption

The biggest barrier to moving to a containerized infrastructure, identified by over 56% of respondents, is the maturity of the technology.

It's difficult to keep track of all of the container-related tools and projects released over the past two years, and with some of them changing names and many key tools still in beta, this rapid rate of development has not yet painted a picture of stability or maturity.

However, the survey was closed before the announcement of the formation of the OCI in June, and before the launch of the Cloud Native Computing Foundation (CNCF) on July 21st, which aims to harmonize existing technologies to foster development of container-packaged, dynamically managed, microservices-based applications. It's too early to say what impact these initiatives will have on container technology in the short term: there may be yet another period of flux as the existing technologies are harmonized. However, being backed by industry leaders with existing tools in this space, these initiatives offer an auspice of stability, and will provide an entry point for those seeking a set of stable, integrated solutions to get them started with containers.

Orchestration: A Key Challenge in Large Deployments

Half of the respondents indicated that orchestration is a challenge when adopting containers. Respondents who are already using Docker were more likely to provide this response. Orchestration involves coordinating and connecting containers within multi-container, multi-host applications, and is particularly important for applications with a microservice architecture. For example, a web application might consist of a cluster of containers running web servers to host multiple instances of the frontend (for failover and load balancing) as well as a number backend services each running in separate containers.

Orchestration tools that explicitly support containers include Kubernetes from Google and Mesosphere's Marathon, built on Apache Mesos. Docker's own orchestration tools—Swarm, Compose, and Machine—were released in February, with Compose replacing the earlier Fig tool. When respondents were asked which container-related technologies they were using, 38% indicated that they use or intend to use Docker swarm, while 22% are using or plan to use Kubernetes and 22% have opted for Mesos. Respondents who are using Kubernetes tended to be those from larger organizations.

Docker Swarm and Machine are both still in beta, while Version 1.0 of Kubernetes was released on July 21st at the same time that the CNCF was launched, so it will be interesting to compare the rate of adoption for these tools after they have had a chance to mature.

Monitoring Scalability a Limiting Factor

As containers are being used more widely in production, the demand for tools for monitoring them is also increasing. Forty-six percent of respondents identified monitoring as a key challenge to moving toward containerized applications. Traditional Linux-based monitoring and reporting tools that are designed to run on a single host and that rely on analyzing log files on disk (e.g., under */var/log*) don’t scale well to multi-container clustered applications and aren’t well suited even to monitoring single-container apps because unless they are written to a data volume, disk contents are not persisted when containers are shut down. A centralized approach to logging and monitoring clusters of applications using container-aware monitoring tools or services is a better fit for managing dynamic multi-container applications

Self-hosted solutions for monitoring container health include the open source tools cAdvisor, Sensu, and Prometheus.

Docker has supported a Stats API since version 1.5 for streaming resource usage stats from running containers, and a number of vendors provide hosted monitoring solutions built on top of the Docker Stats API, including New Relic, Scout, DataDog, SignalFX, and AppDynamics. These tools monitor the state of the containers themselves, but they rely on data reported through the Stats API and hence don’t provide much application-specific detail about the applications or services running inside of the containers. Ruxit’s Docker Monitoring supports auto-discovery of new containers, and discovery of the services and applications running within them, allowing app-centric monitoring of dynamic distributed applications and microservices running within Docker containers.

Automation

Automation is another challenge identified by 40% of respondents, and is particularly useful for implementing distributed applications based on immutable microservices, i.e., where a service will be replaced with a new container rather than being upgraded in place. Traditional continuous deployment tools like Jenkins (in combination with a CM tool like Ansible for the deployment side) and container-enabled services like CircleCI support pulling Docker images directly from DockerHub and building and deploying containers directly as part of a CI/CD process. Service discovery tools, including etcd and Consul, are an important part of this story, to enable services to dynamically discover other services that they need to communicate with, rather than having to link them advance.

Conclusion

The high rate of adoption of container technologies by the survey respondents reflects the current industry buzz around containers and Docker. With more than half of the respondents planning to deploy containers in production within the next 6–12 months, there is a clear need for production-ready, stable, integrated solutions that address the key challenge areas identified by the survey: orchestration, monitoring, and automation for distributed containerized applications.

About the Author

Anna Gerber is a full-stack developer with 15 years of experience in the university sector, formerly a technical project manager at The University of Queensland ITEE eResearch specializing in digital humanities, and a research scientist at the Distributed System Technology Centre (DSTC). Anna is a JavaScript robotics enthusiast and maker and enjoys tinkering with soft circuits and 3D printers.