GIGAOM
RESEARCH

Docker and the Linux container ecosystem

A Buyer's Lens Report by Janakiram MSV

Docker and the Linux container ecosystem

12/17/2014

Table of Contents

1. Executive Summary
2. Introduction: the evolution of Docker
3. Container technology stack
4. Key players of the Docker ecosystem
5. Key takeaways
6. About Janakiram MSV
7. About Gigaom Research
8. Copyright

1 Executive Summary

Linux container technology is experiencing tremendous momentum in 2014. The ability to create multiple lightweight, self-contained execution environments on the same Linux host simplifies application deployment and management. By improving collaboration between developers and system administrators, container technology encourages a DevOps culture of continuous deployment and hyperscale, which is essential to meet current user demands for mobility, application availability, and performance.

Many developers interchange the terms “container” and “Docker,” sometimes making it difficult to distinguish between the two, but there is a very important distinction. Docker, Inc. is a key contributor to the container ecosystem in the development of orchestration tools and APIs. While container technology has existed for decades, the company’s open-source platform, Docker, makes that technology more accessible by creating simpler and more powerful tools. Using Docker, developers and system administrators can efficiently manage the lifecycle of tens of thousands of containers.

This report provides a detailed overview of the Linux container ecosystem. It explains the various components of container technology and analyzes the ecosystem contributions from companies to accelerate the adoption of Linux-based containers.

Thumbnail image courtesy of malerapaso/iStock.

2 Introduction: the evolution of Docker

The concept of containers is not new. Traditional Linux OS distributions with kernel 3.8 or above support container technology. Since they are designed as complete operating systems, they include the packages, drivers, and tools to run applications in the traditional form. Contemporary Linux distributions like CoreOS are lightweight and optimized to run applications in containers. This reduces the footprint of the OS, making it ideal to run web-scale workloads.

Container workloads that run on elastic infrastructure like Amazon EC2 and Google Compute Engine deliver a new level of scalability. Each provisioned virtual machine can run multiple containers and scale out on-demand. With reduced dependency on the OS, workloads are packaged and deployed on container-optimized distributions like CoreOS. This new orchestration mechanism handles the provisioning, monitoring, and maintenance of containers on a variety of infrastructure layers.

Google's Kubernetes is one of the leading orchestration tools to manage containers on GCE, Azure, Rackspace, and other clouds. Kubernetes uses a portal that exposes the APIs, simplifying the user experience for developers and administrators. Additional configuration management tools like Puppet, Chef, Ansible, and Salt Stack now support containers. Tools like Nagios, Ganglia, and Opsview centrally monitor the containers running on a host.

FreeBSD Jvbails, Solaris Zones, and Linux-VServers support operating system-level virtualization to isolate processes running a single OS instance. When compared to traditional virtualization, container technology allows multiple isolated, independent processes to run on a single host. Linux Containers (LXC) is growing in popularity, particularly in recent months, for running containerized applications on distributions like Red Hat Enterprise Linux, Ubuntu, SUSE, or any flavor of Linux with kernel 2.8 or above. LXC comes with a basic set of tools and APIs for creating and managing containers on any Linux host OS running supported kernel version.

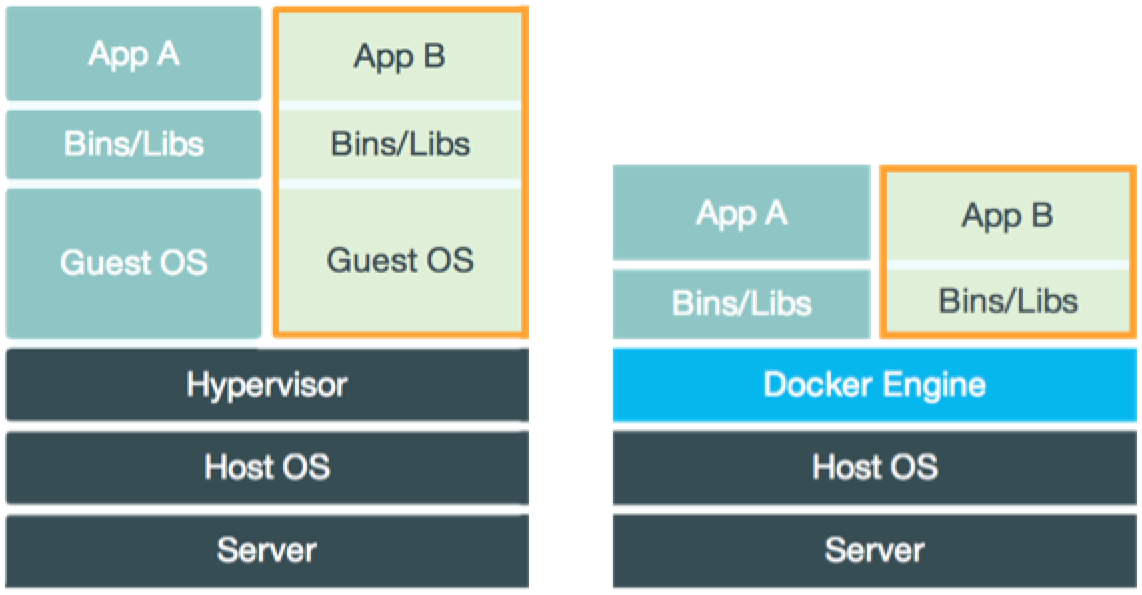
What is Docker?

Docker is an open-source platform that automates the deployment of applications into containers through an additional layer of abstraction. This new layer abstracts at the operating system level. Docker adds an application deployment engine on top of the existing Linux OS kernel capabilities like cgroups and namespaces.

Docker architecture has the following components:

- **Docker client and server.** Docker is a client-server architecture. It has a daemon that acts as the server that exposes the RESTful API. Docker clients consist of either the command line interface (CLI) tools or consumers of the server API. Docker ships with a set of CLI tools to interact with the server.
- **Docker images.** Docker images are the templates for creating containers. As the primary build component, they may contain an OS, a web application, or a database language.
- **Docker registries.** A Docker registry is a catalog, either private or public, of Docker images. The public registry hosted by Docker, Inc is called Docker Hub. Docker Hub has official repositories for Ubuntu, MySQL, CentOS, WordPress, mongoDB, and others, as well as community-contributed images. The images are customizable and can be registered behind a private firewall.
- **Docker containers.** Containers are the unit of execution in Docker. Created from the images, containers hold the application and all of its dependencies, including binaries, libraries, configuration files, and scripts. A container runs one or more processes. A Docker container differs from a virtual machine in that it does not require a guest OS within the container. Containers are more efficient than virtual machines because they use a single storage space with more granularity at each level.

Virtual machine versus container



Source: Docker, Inc.

Docker can run on popular 64-bit Linux distributions with kernel 3.8 or later. It's supported by several cloud platforms including Amazon EC2, Google Compute Engine, and Rackspace.

Docker platform

Initially started as a technology to augment Linux containers, Docker rapidly transformed into a platform. With the release of Docker 1.0, the company added Docker Hub, tools for collaboration between developers and operations teams, and enterprise support for running production applications. The Docker community contributed by creating Dockerized applications, tools, and environments that enhance support for host OS and simplify Docker usage. With that, Docker has moved from being just a container engine to an open platform to build, ship, and run distributed applications.

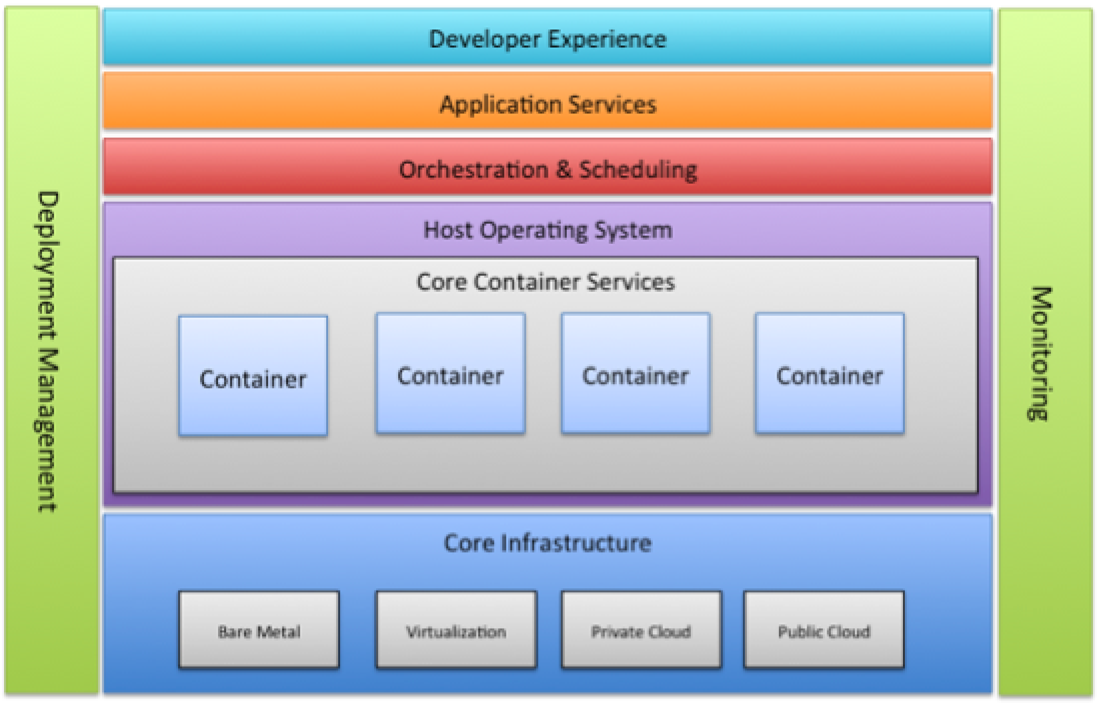
3 Container technology stack

With Docker serving as the foundation for running an application in containers, the ecosystem started to build additional layers to drive the adoption. Various players in the community ranging from OS distribution companies like Red Hat, Canonical, and CoreOS to cloud infrastructure as a service (IaaS) providers like Google, Digital Ocean, and CenturyLink all contributed to the initiative. Many independent developers and individuals created Docker related projects on github.

Like many successful open-source projects, Docker's early adoption can be attributed to the community. There are currently multiple projects that make Docker a viable platform to run scalable, production applications.

Before analyzing the major community contributions, it is important to understand the technology stack.

Linux container architecture



Source: Gigaom Research

- **Core infrastructure.** The host OS running Docker containers is agnostic to the underlying core infrastructure, which can be physical, virtual, or cloud. To provision hosts, the core infrastructure layer exposes the API that enables container scheduling and orchestration. This layer includes bare metal, hypervisors, private cloud management platforms, and public cloud providers.

- **Host operating system.** Each Docker container runs within a host OS. The Docker daemon interacts with the kernel by calling the appropriate system calls. This layer provides support for LXC, namespaces, and cgroups, which are the essential building blocks of container technology. Linux OS distributions are shipping with the packages and tools that are required to run Docker containers. Some distributions are highly optimized and solely rely on Docker to run applications.

- **Core container services.** This layer acts as container management. Core container services include the daemon that runs on each host, the API layer, command line tools, image management, and container lifecycle management. While Docker is the popular container management service, there are other options for container management.

- **Orchestration and scheduling.** Docker images and containers are analogous to virtual-machine templates and provisioned virtual machines. Similar to the way the cloud-management platform (CMP) coordinates the provisioning of VMs on the physical hosts by interacting with the hypervisor, container orchestrators coordinate with the hosts that are responsible for running the containers. This layer is responsible for provisioning, scheduling, tracking, and managing single or multiple containers.

- **Application services.** Application services offer another abstraction layer that enables developers to treat multiple containers as a single unit of deployment by linking workload containers together. Application services bridges the gap between containers and mainstream development tools and platforms. Application services provide what a Platform as a Service (PaaS) layer would otherwise offer to developers.

- **Developer experience.** Developer experience seamlessly integrates container technology with popular integrated development environments (IDE), enabling one-click deployment of code to Docker. This layer abstracts all the key components of the Docker platform through a web portal that offers developers rich user experience and self-service capabilities to manage containers.

- **Deployment and configuration management.** Deployment and configuration management enables the operations team to efficiently deal with containers. Fundamental to DevOps, it enables developers and operations teams to use a common set of tools and technologies. This layer is responsible for maintaining the repository of images and their dependencies, living within private or public registries. Teams implementing continuous integration (CI) and continuous delivery (CD) heavily rely on container deployment management tools.

- **Containers running persistent services.** These are part of mission-critical workloads need to be closely monitored. With Docker adding a layer of abstraction, it is important to monitor the container, host OS, and the physical machine that is running the OS. Existing monitoring technologies are quickly evolving to support containers.

4 Key players of the Docker ecosystem

There is an explosion of tools around Docker and container technology, with new companies emerging almost on a daily basis. While it is not practical to include every company and individual contributing to Docker, this section attempts to map key players of the ecosystem based on their active contributions.

Core infrastructure services

Core infrastructure providers include emerging Docker-specific hosting companies along with traditional IaaS providers. They provide the baseline infrastructure support for running containers.

- **CenturyLink.** CenturyLink is pushing Docker across its IaaS and PaaS offerings. It developed Panamax, an Apache 2 open-source management platform, to make it easy for developers to launch Docker containers on their cloud.
- **Digital Ocean.** Digital Ocean recently launched CoreOS, which natively runs Docker containers. It offers an SSD-only cloud with simple APIs and an intuitive developer dashboard.
- **Google Compute Engine.** Google Compute Engine is one of the first to offer containers optimized on VMs. The Kubernetes Kubelet is an agent to manage the containers. A typical image may include Debian 7, Docker runtime, and a metadata framework.
- **Microsoft Azure VMs.** Microsoft supports Docker containers on Azure VMs through a Virtual Machine Extension. The VM Extension enables the Linux kernel dependencies to function on Windows OS.
- **OpenStack.** OpenStack supports containers through the Nova driver for Docker. Nova enables the management of hundreds of containers across multiple hosts. Docker's integration with OpenStack Glance and Horizon is on the roadmap.
- **Orchard.** Recently acquired by Docker, Inc., Orchard enables rapid container deployment to the cloud. A developer can install the Docker driver and CLI on a laptop and begin deploying on the Orchard cloud.
- **Rackspace OnMetal.** Rackspace's bare-metal cloud runs workloads exclusively on Docker containers hosted on CoreOS.
- **StackDock.** Stackdock from Copper.io is another Docker hosting platform. Inspired by Digital Ocean, it has SSD infrastructure and a simple Docker wizard.

- **Tutum.** Tutum is a pure-play Docker hosting service (Beta) that offers deployment, management, and orchestration.
- **VMware.** VMware announced its support for Docker at VMworld 2014. It is yet to ship the tools to integrate with containers with vSphere, vCenter, or vCloud Air.

Host operating system

Linux OS distributions with kernel 3.8 or above support Docker containers. They provide the host OS service to the containers.

- **Canonical Ubuntu.** Starting with Ubuntu 14.04 LTS, Canonical started shipping Docker natively with the OS distribution. This move makes Docker accessible to all Ubuntu users and developers. Snappy Ubuntu Core is a stripped down version of Ubuntu designed to run web-scale workloads packaged in containers.
- **CoreOS.** Based on ChromeOS, CoreOS is an open-source lightweight operating system designed for clustered container deployments. It does not have a package manager. All applications run inside Docker containers.
- **Project Atomic.** Project Atomic is an initiative from Red Hat to create Atomic Hosts that are minimal versions of Red Hat Enterprise Linux, CentOS, or Fedora designed to run Docker containers. Atomic Hosts have a smaller footprint, better security, and manageability.
- **Red Hat Enterprise Linux.** Red Hat has been supporting Docker since Red Hat Enterprise Linux 6.5. Docker is integrated with the latest version, RHEL 7.

Core container services

Core container services are responsible for the creation, maintenance, and termination of containers. Their primary responsibility is to provide lifecycle management of containers.

- **Docker.** Docker is one of the most popular platforms, providing core container services including tools, lifecycle management, image management, and registry management.
- **FlockPort.** Based on LXC, FlockPort is a container management engine. Though not related to Docker, it is attempting to create its own ecosystem.

Orchestration and scheduling

While Docker takes care of the core container services, orchestration and scheduling engines manage containerized applications across multiple hosts or VMs. They provide mechanisms for deployment, maintenance, and scaling of applications. The orchestration and scheduling tools are evolving rapidly.

- **BOSH.** Originally created as the tool to deploy on Cloud Foundry, BOSH has been updated to deploy and orchestrate persistent Docker containers across many VMs and multiple IaaS providers. The BOSH Deployment Manifest is a YAML file that defines the properties of the deployment components. These blocks, initiated using CLI, instruct the BOSH Director on the new deployment.
- **Centurion.** Built by the popular logging service company New Relic, Centurion takes containers from a Docker registry and runs them on a fleet of hosts with the correct environment variables, host volume mappings, and port mappings. Centurion separates builds from deployments to simplify the two tasks. New Relic also created Shipright, for rapid deployment within hours. RAKE, the build utility for Ruby, is the foundation.
- **Clocker.** Clocker is a Docker orchestration engine based on Apache Brooklyn, an open-source framework for modeling, monitoring, and managing applications through autonomic blueprints. Clocker differentiates itself by creating a SSH server via Dockerfile. The server is treated like a VM, allowing portability and fault tolerance. Apache Brooklyn uses jclouds a robust Java toolkit.
- **Crane.** Crane orchestrates Docker containers by reading declarative configuration files in the form of YAML or JSON. It makes it easy to repetitively setup identical environments. It also allows for the segmentation of containers into groups and orchestration by group.
- **io.** Consul is a tool for discovering and configuring services. Service Discovery uses DNS or HTTP to find all service providers with availability. In addition, it offers granular health check at the service provider level or by specific node. This allows developers and network operations to make dynamic provisioning decisions based on service provider performance.
- **Decking.** Based on js**,** Decking aims to simplify the creation, organization, and management of clusters of Docker containers. Decking associates the metadata with images or containers that Docker has abstracted. By defining parameters for each container, it simplifies orchestration.
- **Deimos.** Deimos is a Docker plugin for Apache Mesos that provides external containerization of clusters. Combined with the Marathon meta framework and a REST API, applications can be orchestrated.
- **Docker Swarm.** A native tool from Docker, Inc, Swarm makes it easy to deploy containers on a cluster. It ships with a simple scheduler but has a pluggable architecture that allows swapping in other backends.

- **Dockerize.it**. Based on an agent, Dockerize.it manages and orchestrates Docker containers from anywhere. It can be managed from hosts running in an enterprise datacenter behind the firewall.
- **Flocker**. Flocker from ClusterHQ is a data volume manager and multihost Docker cluster-management tool. It makes the data portable along with the container.
- **Geard**. Borrowing the terminology from OpenShift, Red Hat created an orchestration engine to run Docker containers in production. Combined with systemd, a service manager for Linux, Geard is a service agent for cluster orchestration.
- **Kubernetes**. Google, which claims to launch over 2 billion containers every day with Kubernetes, maintains the attention of the industry. Kubernetes has become the preferred orchestration engine to manage Docker containers on Google Cloud Platform, Rackspace, and Azure.
- **Maestro**. MaestroNG is a command-line utility developed by Signalfuse. It automatically manages and orchestrates deployments by bringing up of a set of service instance containers on a set of target host machines.
- **Marathon**. Marathon is a cluster-wide init and control system for services in cgroups or Docker containers. It integrates Docker with Apache Mesos. Marathon is written in SCALA.
- **Shipper**. Built by the Mailgun team at Rackspace, Shipper is a fabric for Docker and a tool for orchestrating Docker containers. It supports parallel execution and can generate command line interface.
- **Shipyard**. Shipyard enables multihost, Docker cluster management. It uses the Citadel toolkit for cluster resourcing and scheduling.

Application services

Application services provide an abstraction layer for developers to interact with the container layer. They enable developers to treat multiple containers as a single unit of deployment by linking the containers that belong to the same workload.

- **AWS Beanstalk**. AWS Elastic Beanstalk provides a way for developers to deploy and manage Docker containers on AWS. Applications can be packaged as Docker images that are stored in a public or private Docker repository.
- **Amazon EC2 Container Service**. Amazon EC2 Container Service (ECS) is a cluster-management service for containers deployed on Amazon EC2. It integrates with AWS services like Identity Access Management (IAM), Elastic Block Storage (EBS), and Security

Groups. It enables testing of high availability of clusters prior to deployment. This service is in preview as of December 2014.

- **Cloud Foundry.** Pivotal is rewriting Droplet Execution Environment (DEA) in Go language called Diego that supports Docker containers. This will enable Warden, the existing container technology of Cloud Foundry to accept Docker containers.
- **Deis.** Deis is an open-source PaaS offering that makes it easy to deploy and manage applications. Deis builds upon Docker and CoreOS to provide a lightweight PaaS with a Heroku-inspired workflow.
- **Dokku.** Dokku is positioned as mini-Heroku powered by Docker. It is written with less than 100 lines of Bash. Once it's set up on a host, developers can push Heroku-compatible applications to it via Git. They can build using Heroku build packs and then run in isolated containers.
- **Google App Engine.** Google built a set of extensions that allow App Engine developers to build and deploy Docker images in Managed VMs. Developers can use these extensions to access the library of Docker images to deploy containers into a completely managed environment with access to other Google Cloud Platform services.
- **Google Container Engine.** Google Container Engine (GCE) is a frontend for Kubernetes. It abstracts provisioning and scheduling containers on Google Compute Engine. Announced at Google Cloud Platform Live event in November, this service is still in Alpha.
- **Octohost.** Octohost is a simple web focused, Docker mini-PaaS server that can be run in a variety of IaaS platforms including Amazon EC2, Digital Ocean, and Rackspace.
- **OpenShift.** Developed by Red Hat, OpenShift PaaS is integrated with Docker, Kubernetes, Geard, and Project Atomic. It is one of the first Private PaaS platforms to support containers.
- **Stackato.** Stackato from ActiveState is one the first enterprise-grade cloud application platforms to incorporate Docker. Stackato provides the necessary layer of orchestration and application services, allowing developers to quickly shrink or grow the pool of Docker containers within which an application operates.

Developer experience

Since Docker is implemented via API and CLI, developer experience is important to driving adoption in the developer community. Developer experience includes standalone web portals and plugins that integrate Docker with IDEs.

- **DockerUI.** DockerUI is a web interface that makes it possible to interact with the Docker remote API. It provides a pure client side implementation to effortlessly manage Docker.

- **Panamax.** CenturyLink built Panamax to provide a friendly interface for users of Docker, Fleet, and CoreOS. It incorporates Docker best practices and gives developers UX that does the heavy lifting of deploying complex containerized applications.
- **WaveMaker.** Pramati Technologies bought WaveMaker from VMware to turn it into a standalone PaaS. WaveMaker Cloud has a console to manage containerized applications running on Docker.

Deployment and configuration management

Deployment and configuration management tools handle the packaging, deployment, and configuration management of containers.

- **Ansible.** Ansible brings its traditional configuration management to Docker by making it easy to build, launch, and provision containers through Dockerfiles.
- **Chef.** The popular configuration-management tool supports containers by extending its cookbooks and recipes to include Docker configuration. Knife container is a plugin to manage Docker containers.
- **Docker Compose.** Docker Compose enables developers to assemble applications from autonomous and interoperable Docker containers independent of underlying infrastructure. This is based on Fig.
- **Fig.** Fig, which is now a part of Docker, Inc, wants to do to containers what Vagrant did to VirtualBox. It provides reusable configurations of containers that can be reproduced in any environment.
- **Packer.** Packer aims at packaging portable machine images that can run on any cloud. It supports creating images from Docker snapshots that can be used to start a pre-configured Docker instance on supported cloud platform.
- **Shippable.** Shippable delivers continuous integration and continuous delivery (CI/CD) for containerized applications.
- **StackEngine.** One of the new entrants of the Docker ecosystem, StackEngine aims to be the vCenter for automating, monitoring, and managing containerized applications.

Monitoring

Traditional monitoring tools are gearing up to support containers. Most of them support monitoring the host along with all the containers running on it.

- **cAdvisor.** Google created cAdvisor to monitor their lmctfy containers. Google has extended cAdvisor to support Docker containers. It is a running daemon that collects, aggregates, processes, and exports information about production containers.
- **DataDog.** DataDog uses an agent to monitor and report container metrics based on tags. It can be used to track specific metrics for many containers in aggregate, which can be visualized through a variety of charts.
- **Dataloop.** Dataloop claims to be one of the first monitoring platform for micro-services and containers.
- **Logspout.** Logspout attaches to all running containers on a host, then routes their logs to a predefined location.
- **Scout.** Scout is a Ruby-based monitoring platform that supports Docker. It can report metrics such as memory used, CPU utilization, and the number of running containers.

5 Key takeaways

The container ecosystem is growing at a rapid pace, and further consolidation will continue to change the landscape. Though each player is focusing on a specific area related to containers, they are all racing towards becoming the one-stop-shop for container-management tasks. Docker, Inc., which started as the core container management company, is continuing to grow, acquiring Fig and Orchard to support provisioning, automation, and management. Red Hat is heavily investing in containers through initiatives like Project Atomic Host, and certifications for containerized applications, Geard and OpenShift. Google is expected to tightly integrate containers with its IaaS and PaaS offerings.

Docker container technology is one of the most promising disruptive solutions to drive cloud adoption. The Docker community envisions the ability to leverage multiple IaaS service providers for portability, scale on demand, fault-tolerance, and performance, all while maintaining continuous application delivery.

6 About Janakiram MSV

Janakiram MSV is an Analyst for Gigaom Research and the Founder and Principal Analyst at Janakiram & Associates. He was Founder and CTO of Get Cloud Ready Consulting, a niche Cloud Migration and Cloud Operations firm that got acquired by Aditi Technologies. He is a guest faculty at the International Institute of Information Technology, Hyderabad (IIIT-H) where he teaches Big Data, DevOps and Cloud Computing to the students enrolled for the Masters course. Janakiram has worked at world-class product companies including Microsoft Corporation, Amazon Web Services and Alcatel-Lucent. His last role was with Amazon Web Services as the Technology Evangelist where he joined them as the first employee in India.

7 About Gigaom Research

Gigaom Research gives you insider access to expert industry insights on emerging markets. Focused on delivering highly relevant and timely research to the people who need it most, our analysis, reports, and original research come from the most respected voices in the industry. Whether you're beginning to learn about a new market or are an industry insider, Gigaom Research addresses the need for relevant, illuminating insights into the industry's most dynamic markets.

Visit us at: research.gigaom.com.

© Giga Omni Media 2014. *"Docker and the Linux container ecosystem"* is a trademark of Giga Omni Media. For permission to reproduce this report, please contact research-sales@gigaom.com.