Linux Containers: Why They're in Your Future and What Has to Happen First

What You Will Learn

Linux containers and Docker are poised to radically change the way applications are built, shipped, deployed, and instantiated. They accelerate application delivery by making it easy to package applications along with their dependencies. As a result, the same containerized application can operate in different development, test, and production environments. The platform can be a physical server, virtual server, public cloud, or network device.

This white paper is intended for IT leaders and industry analysts. It explains:

- Why Linux containers and virtual machines are optimized for different types of workloads
- What needs to happen before enterprises and service providers will use Linux containers in production
- How Cisco and Red Hat are investing to make containers ready for production

This document assumes that you know the basics of Linux containers. The **Linux Journal** provides a good overview.

Application Delivery: Today's Challenges

Today's applications are more complex, and yet they must be developed more quickly. These trends increase demands on infrastructure, IT teams, and processes. IT departments are struggling to find ways to:

- Lower costs by helping teams do more with the same staff size
- Respond more quickly to new business requirements
- Keep systems and data secure
- Adopt innovative development and hosting methods using existing infrastructure

Where Virtual Machines Fit

Containers and virtual machines both allow multiple applications to run on the same physical systems. They differ in the degree to which they meet different kinds of business and IT requirements.

Virtual machines do a great job at what they were designed to do: abstract from the underlying hardware. This lowers costs and makes it possible to automate provisioning of a complete software stack, including the operating system, the application, and all application dependencies. At Cisco, for example, developers visit an online catalog to self-provision a complete application infrastructure - computing, networking, and storage - in 15 minutes. By automating infrastructure as a service (IaaS) and platform as a service (PaaS) solutions, the Cisco IT team reduced overall data center total cost of ownership (TCO) by more than 65 percent. Some of these savings came from server consolidation. Other savings resulted from simplified system administration, because different operating systems now can run on the same hardware.

Virtual machines aren’t ideal for every use case:

- Virtual machines need minutes to spin up, which can degrade the user experience and give hackers time to exploit known vulnerabilities during bootup.
- Patching and lifecycle management for virtual machines requires a significant effort. That’s because every virtualized application has at least two operating systems for operators to manage and secure: the hypervisor and the guest OS that is inside the virtual machine.
- Even the simplest OS process needs its own virtual machine. This requirement increases flexibility, but it also makes virtual machines impractical to use for microservices architectures with hundreds or thousands of processes.
- When each physical server is replaced by one virtual machine, physical resource utilization tends to remain low. Server sprawl is simply replaced with virtual machine sprawl.

Businesses, academia, and government can gain from a more efficient way to build, ship, deploy, and run applications. That’s where Linux containers come in.

Where Linux Containers Fit

Briefly, a Linux container is a set of processes that are isolated from the rest of the machine. A container can encapsulate any application dependency. For example, if a website relies on a particular version of the PHP scripting language, the container can encapsulate that version. As a result, multiple versions of the same scripting language can co-exist in the same environment - without the administrative overhead of a complete software stack, including the OS kernel. Containerized applications perform about as well as applications deployed on bare metal.

Where hypervisors provide a logical abstraction at the hardware level, containers run in isolation, sharing an operating system instance. This approach can improve application delivery in several ways:

- Lowering costs
- Speeding up application development
- Simplifying security
- Making it easier to adopt new IT models such as hybrid clouds and microservices architecture

Lower Costs and Greater Agility

Containers simplify IT operations:

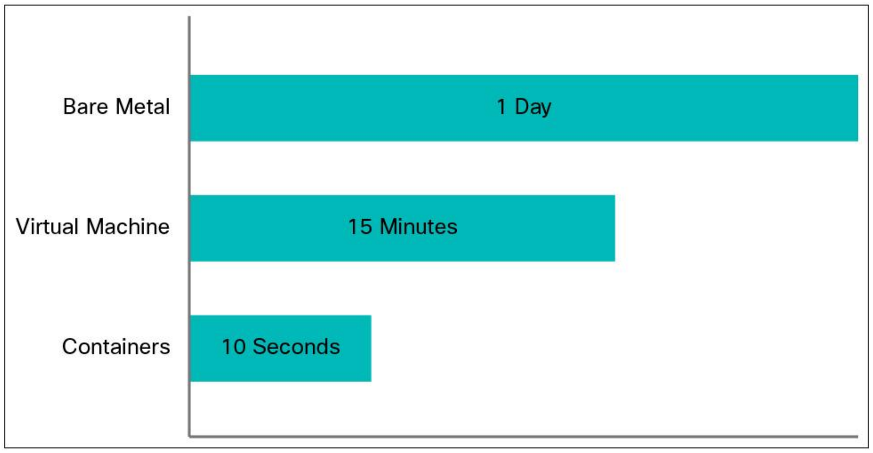
- **Fewer operating systems to manage:** Each virtual machine can be carved into multiple containers, all sharing the same operating system kernel.
- **Greater application mobility:** You can move workload between private and public clouds more quickly, by orders of magnitude. Instead of moving gigabytes between clouds, you can move megabytes.
- **Easier OS patching:** A virtualized application with 10 virtual machines has 11 operating systems - the hypervisor and each guest operating system - and each needs patching. In contrast, a containerized server with 10 applications has only one operating system.
- **Easier application patching:** Docker container images are composed of layers, and you can patch by just adding a layer. The new layer doesn’t affect the others. The layers of a web application image, for example, might include an Apache web server, a PHP runtime system, and Redis for caching.

- **Better workload visibility:** Operators can't see the workload inside a virtual machine, but they can look inside a container from the container host environment. One benefit is the ability to detect unused container instances, helping to avoid virtual machine sprawl. Another benefit is simplified dependency mapping and security maintenance.

- **Improved resource utilization:** Linux containers run on a single computing instance, making it easier to detect their activity and retire unused containers. Idle containers don't take up computing, memory, and I/O resources.

- **Faster provisioning:** Containerized applications can boot and restart in seconds, compared to minutes for virtual machines (Figure 1). Booting and restarting are faster because containers have smaller payloads and don't carry the overhead of a hypervisor and guest OS. Faster boot time correlates with less downtime. This benefit is especially appealing for organizations that want to use a public cloud service to handle higher-than-usual transaction volumes. In this case, faster boot and restart times can directly reduce costs.

Figure 1: Containerized Applications: Faster Provisioning



Faster Response to New Workload Requirements

Less time is needed to transport, copy, and instantiate containers because they are smaller than virtual machines. They are often just a few dozen megabytes, where a typical virtual machine might be hundreds of megabytes, or even gigabytes.

Linux containers also tend to speed up application development, because you can build once and run on any infrastructure. Infrastructure can include bare-metal servers, virtual machines, public clouds, and network devices. Encapsulating dependencies inside a container helps shorten the testing cycle. That's because containers make it easier to write single-function applications, which simplifies testing and can accelerate the development and operations (DevOps) cycle. Developers can add new application features more quickly by taking advantage of automated building, testing, integration, and packaging - at the speed of containers.

Containers also help organizations adopt the DevOps model, especially when it is used for PaaS. Each team can focus on what matters to them. Developers are concerned only with what's inside the container: the code. Operations and quality assurance (QA) teams are only concerned with placing and maintaining the containerized code in production.

Keeping Systems and Data Secure

Containers that are secured with Linux namespaces, control groups (cGroups), and Security Enhanced Linux (SELinux) provide almost as much isolation as a virtual machine. They also provide much greater flexibility and efficiency.

Some security issues need to be addressed before enterprises and service providers will use containers in production (see "Paving the Way for Widespread Adoption" later in this document). Containerization will eventually make it easier to secure applications, for three reasons:

- A smaller payload reduces the surface area for security flaws.
- Instead of incrementally patching the operating system, you can update it.
- By allowing a clear separation of concerns, containers help IT and application teams collaborate (see the "Hybrid Cloud" sidebar).

The IT department is responsible for security flaws associated with the infrastructure. The application team fixes flaws inside the container and is also responsible for run-time dependencies. Easing the tension between IT and applications teams helps smooth the transition to a hybrid cloud model.

Delivering New Kinds of Services

Containers provide new ways to speed up application development and deployment. Two notable examples are microservices and open application containers for network devices.

Hybrid Cloud: Helping IT and Application Developers Navigate the Transition

Consider a retail or financial services application that uses SSL to encrypt sensitive information traveling over the Internet. In a virtualized architecture, SSL is part of the application image. So if an SSL security flaw emerges, the developer has to change the application image. This process results in downtime and time-consuming regression testing.

With a container-based architecture, in contrast, SSL can be isolated in its own container, separate from the application code. The flaw takes less time to correct because the developer does not need to touch the application logic.

Microservices

Instead of building one application (monolithic architecture), developers build a suite of components, called microservices, which come together over the network. Each component is written in the best programming language for the task, and each component can be deployed and scaled independently of the others.

Containers are better suited for microservices than virtual machines are because microservices can start up and shut down more quickly. In addition, computing, memory, and other resources can scale independently. Red Hat and other vendors are currently working on orchestration software that connects microservices.

Support for microservices is one reason that containers are an efficient way to deliver PaaS. Red Hat uses containers in OpenShift, which provides a self-service provisioning platform for the full application stack. Red Hat OpenShift also includes development tools and middleware.

Applications that benefit most from a microservices architecture tend to be horizontally scalable. For example, consider a financial services firm that sells options and futures contracts. Its main application might have three components:

- Trader interface
- Trade settlement code that interfaces with order management system and exchanges
- Pricing system based on proprietary algorithms

The pricing system creates the firm's competitive advantage. If the pricing system is in its own container, the firm's developers can quickly swap out old algorithms for new ones. They don't need to touch the code for the user interface or back-end interfaces, and they don't need to retest those components.

Open Application Containers

Cisco uses open application containers for network devices such as routers and switches. The containers add new capabilities to Cisco® network operating systems. One example is performance measurement. Another is integration of the network operating system with third-party tool chains, such as Puppet agent and Chef agent.

The Vision for Linux Containers

Linux containers have the potential to transform application delivery - in the data center, network core, and network edge (Figure 2). Table 1 lists use cases.

Figure 2: Where Containers Will Appear

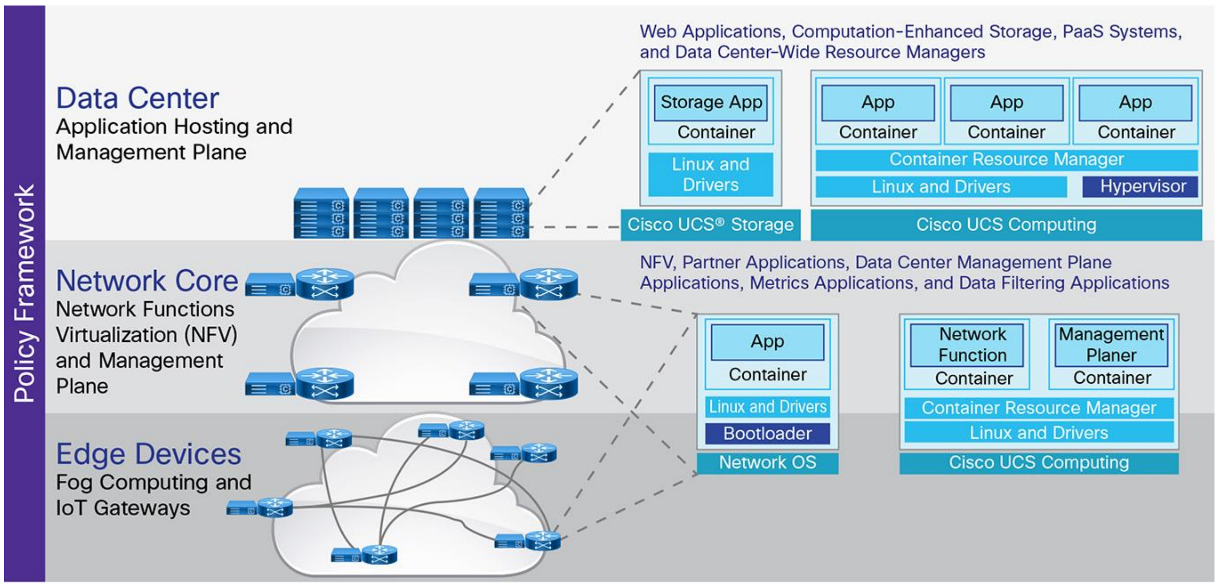

Policy Framework
Data Center
Application Hosting and Management Plane
Web Applications, Computation-Enhanced Storage, PaaS Systems, and Data Center-Wide Resource Managers
Storage App
Container
Linux and Drivers
App
Container
App
Container
App
Container
Container Resource Manager
Linux and Drivers
Hypervisor
Cisco UCS® Storage
Cisco UCS Computing
Network Core
Network Functions Virtualization (NFV) and Management Plane
NFV, Partner Applications, Data Center Management Plane Applications, Metrics Applications, and Data Filtering Applications
App
Container
Linux and Drivers
Bootloader
Network OS
Network Function
Container
Management Planer
Container
Container Resource Manager
Linux and Drivers
Cisco UCS Computing
Edge Devices
Fog Computing and IoT Gateways

Table 1: Use Cases for Linux Containers

Use Case	What's New	Benefit
Development and testing	Replicate self-contained application images on any certified host platform and infrastructure.	Reduce dependency problems. Choose where to deploy.
Low-overhead Linux distributions	Encapsulate libraries and services in the container image.	Reduce OS overhead. Enable the use of small-footprint operating system.
Cloud-native applications	Abstract physical resources and create stateless web and application tiers.	Avoid being locked into a cloud application framework.
Scaling up and down	Spin up containers when workload is heavy. Retire them when they are no longer needed.	Use resources more efficiently.
PaaS	Transport application binaries and configurations in containers. Most PaaS frameworks are already container-based.	Allow PaaS frameworks to interoperate.
Intercloud portability	Natively support Docker containers in public and private clouds.	Exchange application components between clouds.
Microservices	Design applications as suites of services, each written in the best language for the task.	Scale just the microservices that need more resources, not the entire application. Allow different teams to manage different microservices.
Compute-enabled storage	Bring computing resources to data instead of bringing data to computing resources.	Speed up computing operations by not having to move large volumes of big data. Use any language or runtime system.
Shared network functions and services	Virtualize network services in containers instead of on virtual machines or network devices.	Accelerate startup and shutdown, improving user experience.
Containerized control and management planes	Transport control-plane and management-plane software.	Reduce overhead and maintain image integrity.
Edge computing and Internet of Everything solutions	Package, distribute, and run applications and services at the edge to process data closer to the origin.	Conserve bandwidth from the edge to the core. Enable the right kind of services for the particular type of data and type of analysis.
Policy-aware networks	Grant or block access to containers based on policy. For example, apply Quality of Service (QoS) to containers.	Improve the application experience by giving priority to containers with critical services. Improve security.
Network application containers	Run third-party applications in containers on the network operating system.	Add new capabilities to the base network operating system.

Why Now?

Containers aren't new. Linux containers were introduced in 2004. Even earlier examples include Solaris Zones and FreeBSD Jails.

What's new is the use of containers to encapsulate all application components, such as dependencies and services. When all dependencies are encapsulated, applications become portable.

Docker made a big contribution by providing easy-to-use tools and a repository for container images. More recently, other vendors, including Red Hat, have contributed small-footprint operating systems and frameworks for management and orchestration.

Paving the Way for Widespread Adoption

A few large cloud service providers have begun using Linux containers at scale. Some use Red Hat OpenShift as the basis for a PaaS offering. Widespread adoption by enterprises will require improvements in the following areas:

Security

Currently, kernel exploits at the host operating-system level affect all containers on the host. To address this issue, vendors are improving techniques such as mandatory access control. SELinux is a foundational element in Red Hat's security strategy for virtualization; its mandatory access controls protect the host and containers against untrusted container processes. A related project, libseccomp, allows you to eliminate syscalls, preventing a hacked container from compromising the kernel.

Management and Orchestration

Vendors are working to create frameworks for managing container images and orchestrating the container lifecycle. Existing OpenStack and DevOps tools are evolving to support containers, and new frameworks are being developed. Narrowing the frameworks to just a few, and ideally just one, will encourage adoption.

Tools for Managing, Creating, Deploying, and Retiring Containers

Solutions are already available for container management and orchestration. Docker has made it easier to create and delete containers. The container community is currently developing new tools for continuous building and testing of images.

Containers also let you retire unused resources more easily. Containers live **inside** the Linux kernel. So if the security policy permits, you can see what's running in the container. Docker can limit the resources that a container consumes, such as disk space, memory, and I/O, while providing metrics on the use of these resources.

Vendors are also working to create an audit trail for containers. The idea is to add metadata to the container images to show when and where containers are delivered, and their content. Metadata might also include information about who produced the container, the container's products and components (for license management), and certifications.

Live Migration

Workloads requiring the highest availability must not lose their connection when they migrate to another physical or virtual host. An open-source project called CRIU (Checkpoint/Restore in Userspace) satisfies part of this requirement. CRIU continues to mature.

Intercontainer Communication

Linux containers need to talk to other containers on the same or a different host. Communication requires code for endpoint naming, discovery, and connectivity. Because Linux containers operate natively in the Linux kernel, developers tend to use Linux constructs such as iptables to create policies that enable containers to talk to each other.

Better methods for intercontainer communications are available through the open source community. Examples include virtual switching, hardware-enhanced switching and routing, Domain Name System (DNS), and distributed key-value stores. Standardizing on one or a few methods for naming, discovery, and connectivity will help accelerate adoption of Linux containers.

Container Innovations from Cisco and Red Hat

Cisco and Red Hat each have projects underway to bring Linux containers to market. Our strengths are complementary: application development and deployment for Red Hat, and networking and security for Cisco. The network is a critical part of containerized architecture because it carries containers and microservices between hosts and between clouds.

Red Hat, the leading Linux vendor and lead contributor to Linux kernel development, has been working on container technologies since the inception of containers. Red Hat has been instrumental in adding foundational technologies to the Linux kernel that make containers possible. Containers take advantage of major Linux kernel subsystems, including SELinux, cgroups, and namespaces. Red Hat OpenShift Gears are Linux containers that are built from cgroups and namespaces and augmented with the SELinux mandatory access control security subsystem. By building on SELinux, Red Hat created a container model that is both efficient and very secure.

"Cisco Cloud Services is creating an Intercloud of container and micro-services in a cloud native and hybrid CI/CD model across OpenStack, VMware, and public clouds. Look for availability early next year."
- Ken Owens, Cisco Cloud Services CTO

Ongoing projects for Cisco and Red Hat include:

- Hybrid cloud: Both companies are working on allowing free movement of applications into different clouds, with enterprise-class controls. The Cisco initiative is called Cisco Intercloud, and Red Hat is focusing on the Open Hybrid Cloud.
- Application-optimized containers: Today, when a new application is placed in production, a networking team needs to select the appropriate VLAN, open ports, configure load balancing, set up port security through access control lists (ACLs), and apply network policies such as QoS. Cisco and Red Hat are collaborating to automate network configuration using software-defined networking (SDN) and an application-optimized infrastructure. When a new containerized application is placed in production, the network will recognize the application requirements and automatically apply them.
- Microservices integrated with Cisco Intercloud.
- Continuous integration and continuous deployment (CI/CD) in hybrid cloud environments.
- Small-footprint operating system: Red Hat is working on an open-source project called Project Atomic. It provides a space and the tools for the ecosystem to collaborate on small-footprint host operating systems. Red Hat has announced Red Hat Enterprise Linux Atomic Host to deliver an optimized, lightweight container host.

- Container certification: Enterprise applications need enterprise-grade reliability and security for all the technologies inside a container as well as the container host environment. To this end, Red Hat has announced a container certification program to support multivendor deployments.

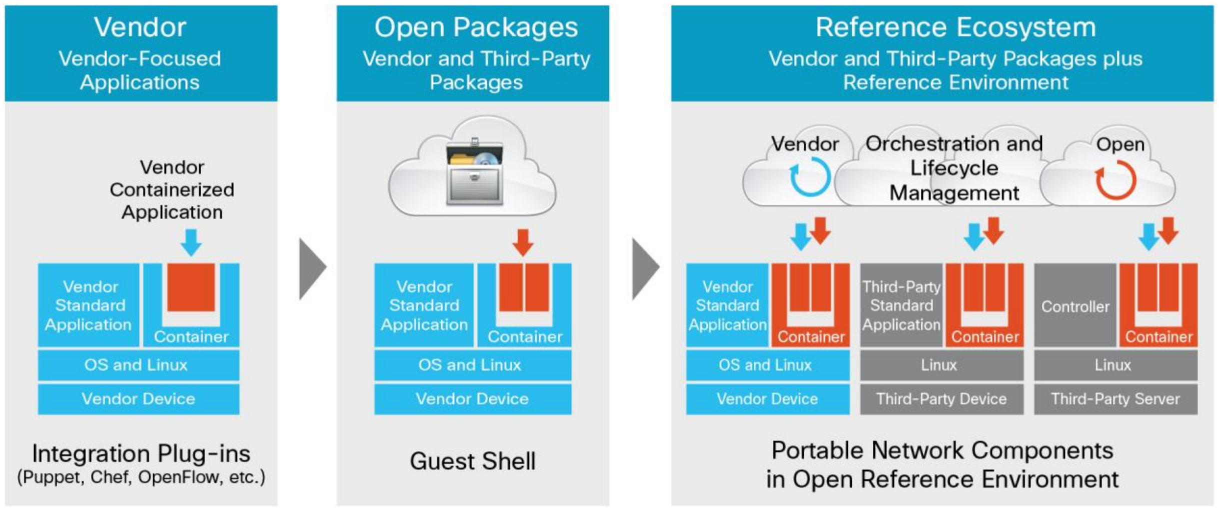
- OpenShift: Red Hat deployed OpenShift for PaaS in 2011. Hundreds of thousands of live containers are running on Red Hat OpenShift. Cisco IT also uses containers in the data center to deploy web applications built on Red Hat OpenShift.

- Containers on Cisco core and edge switches: Some Cisco switches and routers already support Linux containers, allowing Chef processes to run inside Linux containers. Cisco is exploring the use of linux containers in the area of Fog Computing in the context of the Internet of Everything.

- Exploration of opportunities to work with the open source community: Figure 3 shows one progression toward an open ecosystem.

Figure 3: How an Open Ecosystem Might Look

The diagram below portrays the journey how containers are leveraged in embedded devices. Initially, containers are used to host a single component, such as an agent, of a larger application which benefits from tight integration with the embedded device. "Guest Shell" starts to see the container as a subsystem, with the ability to host multiple applications in parallel and open access to Linux packages and tool chains as well as the network. Going forward we'll start to understand the container, pre-equipped with a set of functionality, as a portable reference environment which is an integral part of an eco-system combining network and application centric tools and functions.


Vendor
Vendor-Focused Applications
Vendor Containerized Application
Vendor Standard Application
Container
OS and Linux
Vendor Device
Integration Plug-ins
(Puppet, Chef, OpenFlow, etc.)
Open Packages
Vendor and Third-Party Packages
Vendor Standard Application
Container
OS and Linux
Vendor Device
Guest Shell
Reference Ecosystem
Vendor and Third-Party Packages plus Reference Environment
Vendor
Orchestration and Lifecycle Management
Open
Vendor Standard Application
Container
OS and Linux
Vendor Device
Third-Party Standard Application
Container
Linux
Third-Party Device
Controller
Container
Linux
Third-Party Server
Portable Network Components in Open Reference Environment

Conclusion

The time for Linux containers has arrived. Application packaging is becoming more important because of the Internet of Everything, fog computing, and a return to decentralized systems.

Vendors are responding with innovations to make containers ready for production environments. High-profile projects include Red Hat OpenShift, Apache Mesos, VMware CloudFoundry, Google Kubernetes, and Cisco application containers. These projects have led to notable advances:

- Distributed applications are being built with microservices architecture in mind.

- Apache Mesos is improving scheduling.

- Unique identifier (UID) namespace isolation is strengthening security.

- Docker has improved image management and ease of use.
- Virtualization is moving to embedded systems, allowing routers and switches to host network applications.

The adoption path for Linux containers will likely be similar to that for Linux - that is, slow at the beginning, and then picking up speed. Service providers will take the lead. Enterprises will follow as they become more confident in the technology and see the return on investment (ROI) from service providers.

Cisco and Red Hat are committed to meeting enterprise requirements through our work with hybrid cloud and network-based security.

For More Information

To learn more about Cisco work with Linux containers, email linux-containers-whitepaper@cisco.com.

To learn more about Red Hat work with Linux containers, visit http://www.redhat.com.

Acknowledgements

Cisco: Hicham Tout, Howie Xu, Jenny Koerv, Sandeep Bharda, Rich Gore, Frank Brockners, Roque Gagliano, Luca Relandini, Jari Koivisto, Pamela Lee

Red Hat: Daniel Riek, Bhavna Svarthy, Elisabeth Strenger, Michael Ferris, Dan Walsh

Additional Resources

- Linux kernel namespaces: http://lwn.net/Articles/531114/
- Cgroups: https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt
- LXC: https://linuxcontainers.org/
- SELinux: http://selinuxproject.org/page/Main_Page
- AppArmor: https://wiki.ubuntu.com/AppArmor
- Docker: https://docs.docker.com/userguide/
- Docker libcontainer: https://github.com/docker/libcontainer
- Google Kubernetes container management: https://github.com/GoogleCloudPlatform/kubernetes
- Mesosphere container management: http://mesosphere.io/learn/run-docker-on-mesosphere/
- Coreos JEOS Linux OS: https://coreos.com/docs/
- Project Atomic Linux OS: http://www.projectatomic.io/docs/
- OpenVZ/CRIU Container Checkpoint Restore: http://criu.org/Main_Page
- OpenVZ Live Migration: https://openvz.org/Checkpointing_and_live_migration

Alternative container technologies in other prevalent *Nix operating systems:

- FreeBSD Jails: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/jails.html
- SmartOS Zones: http://wiki.smartos.org/display/DOC/Zones
- Oracle Solaris Zones: http://docs.oracle.com/cd/E18440_01/doc.111/e18415/chapter_zones.htm

Authors

Cisco: Ivan Melia, Sandeep Puri, Ken Owens, Kiran Thirumalai, Sailesh Yellumahanti

Red Hat: Lars Herrmann, Mark Coggin, Joe Fernandes, Kimberly Craven, Dan Juengst

C11-732571-00 09/14