Containers for Highly Scalable Applications in the Cloud

Dr. Ann L. Chervenak

Computer Systems Research Department

Talk Outline

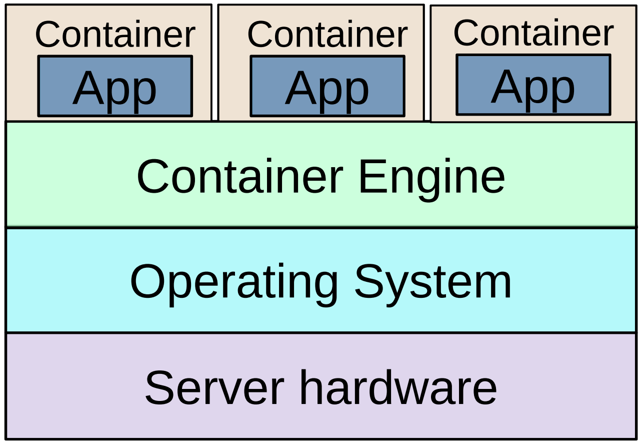
- **Virtualization in clouds**
- **Containers**
  - *A lightweight virtualization mechanism*
  - *Comparison of Containers and Virtual Machines*
- **Minimal Operating Systems to facilitate container scaling**
- **Containers as an enabling technology for highly scalable internet applications**
- **Key technologies for containers**
- **Docker containers**
- **Container Management (Kubernetes)**
- **Security Issues for Containers**
- **Summary**

Virtualization

- **Virtualization techniques allow sharing of physical resources by multiple applications**
  - *Widely used in cloud computing*
  - *Each application sees a model of computation, storage and networking: appears to run on its own machine*
  - *Multiple applications share underlying hardware resources*

- **To enable different workloads to be co-located on a node, virtualization technologies must support:**
  - ***Isolation of virtualized workloads***
    - Workloads run securely in separate software environments
    - Any faults (bugs, crashes, viruses) are contained within virtualization
    - Performance of each workload is independent of others running on server
  - ***Resource management:***
    - Control the resources consumed by each workload
    - Don’t allow any workload to consume all the resources

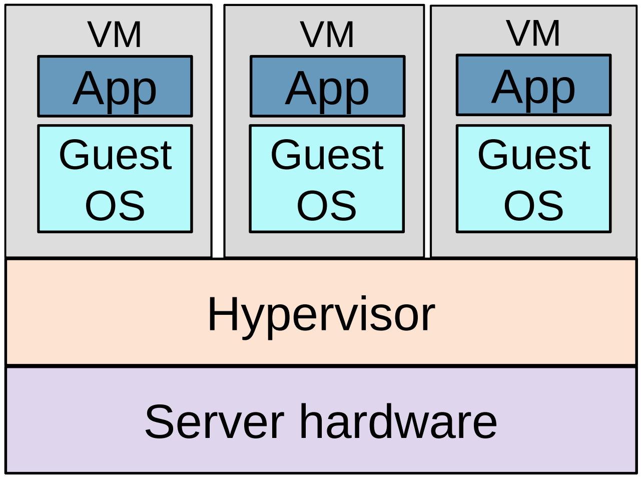
Containers



- **Mature virtualization technology (e.g., Docker, Rocket, LXC)**
- **Can be thought of as virtualizing an Operating System (OS)**
  - *Each container effectively receives a slice of an operating system kernel*
- **Container engine sits above the OS; manages and isolates containers on the server**
- **Each container includes an application and its dependent libraries and binaries**
  - *Packaged for fast, easy deployment*
- **Portable: can run on a wide range of hardware and cloud platforms**
  - ***Easily deploy in development, test and production environments***

Ann.L.Chervenak@aero.org
Computer Systems Research Department

Comparing Containers and Virtual Machines



Virtual machines (VMs)

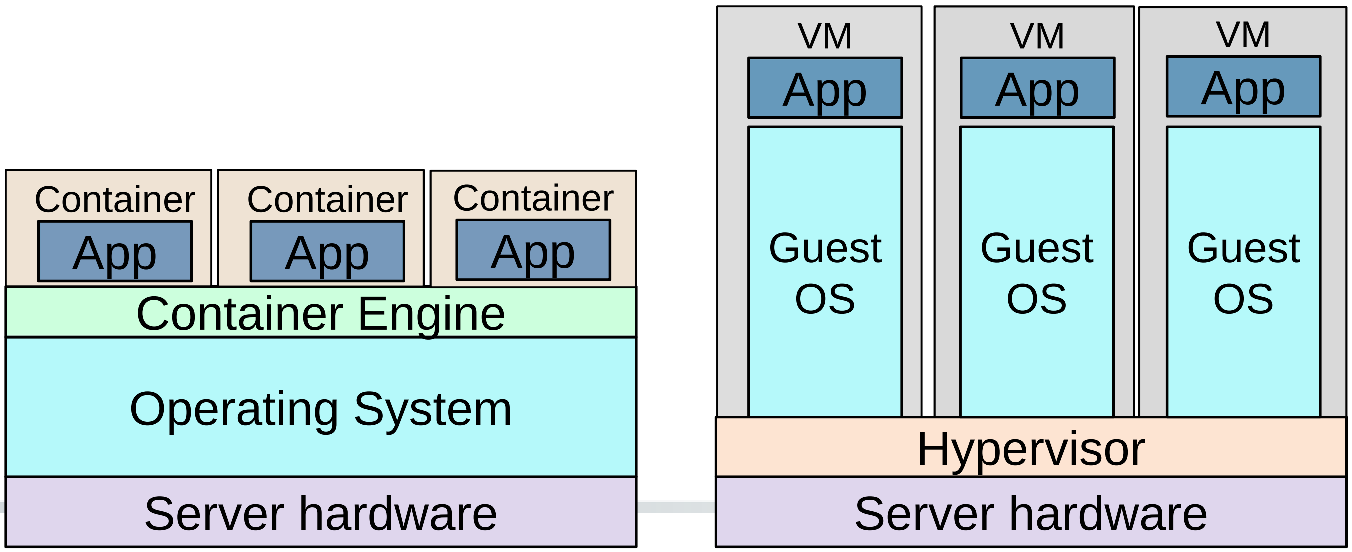
- **Earlier, mature virtualization technology**
- **Can be thought of as providing each VM a slice of the underlying server hardware**
  - *Hypervisor software runs above server, manages one or more VMs*
  - *Portable: Each VM contains a full OS version (the "Guest OS"), necessary libraries and dependencies, and the application*
- **Problem: Modern Operating Systems are very large**
  - *Linux: Over 25 million lines of code; Windows: Over 50 million lines of code; MacOS: Over 85 million lines of code*
- **Having a full Guest OS makes VMs very large**
  - ***Slow to deploy (several minutes)***
  - ***Take up space on server: limits number of VMs that can be deployed***

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE

Containers vs. Virtual Machines

- **Containers provide a lightweight alternative to VMs**
  - *Do not contain a full guest Operating System for every application container*
- **Containers are smaller (*if applications are small compared to OS*)**
- **Can be deployed more quickly** (in seconds rather than minutes)
- **Can be deployed more densely** on cloud resources
- Results in **improved resource utilization and lower power usage**
  - *Important considerations for data centers and clouds*



Ann.L.Chervenak@aero.org
Computer Systems Research Department

Minimal OS or Cloud OS

- **Containers share a single operating system**
- **However, further improvements are needed**
  – ***Potential limits to scalability***
    - Linux is not really optimized for 1000s of processes
  – ***Security is a concern***
    - Still large unused portion of shared OS with potential vulnerabilities
- **Cloud OS or Minimal OS**
  – *E.g., Red Hat Atomic Host, CoreOS, Ubuntu Snappy, RancherOS*
  – *An operating system designed and optimized for use in a cloud environment*
  – ***Goal: include minimal OS capabilities needed to host container-based cloud applications***
    - Containers running on a host share a minimal OS kernel
    - No need for the majority of OS utilities
    - Select the OS utilities normally used by cloud applications
- **Note: VMware also reducing size of OS for VMs**

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE

Containers are a Key Enabling Technology for Modern, Massive Scale Internet Applications

- **Virtual Machines are too large and deploy too slowly to enable fast scaling of interactive, massive internet applications**
  - *E.g., Google search, Gmail, Netflix*
- **Containers are small and deploy quickly (*if applications are small relative to the OS*)**
  - *Can be **deployed more densely on cloud resources***
- **In 2014, Google announced that they launch more than 2 billion container instances per week across their global data centers**
- **Note: Containers don't provide as much advantage if large, monolithic legacy applications are just wrapped in a container and deployed on the cloud**
  - *Get portability but do not achieve density or performance improvements*
  - *Mitigation: refactor legacy applications*

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE

Containers are a Key Enabling Technology for Modern, Massive Scale Internet Applications (cont.)

- **Containers enable a Microservices Architecture approach**
- **Microservices replace large, monolithic applications with a distributed system of lightweight, narrowly focused, independent services that communicate with other parts of the system**
  - *Each microservice is a small application that can be* ***deployed, scaled and tested independently*** *and that* ***has a single responsibility***
  - *In practice, microservices typically range from a few hundred to a few thousand lines of code* ***(* Small compared to size of OS *)***
- **Containers are a good fit to deploy microservices in the cloud**
  - *Can quickly create and destroy containers*
  - *Facilitates quick scaling of applications, continuous delivery of new functionality*
  - *Portable across a range of platforms (development, test, operations environments)*

Ann.L.Chervenak@aero.org
Computer Systems Research Department

Key Technologies for Containers

- **The concept of containers is not new**
  - *Similar technology has been deployed in operating systems starting in 1979*

The current enthusiasm around containers is based on:

- **Recent technology developments to improve the security and isolation of Linux containers:**
  - ***Namespaces provide process isolation***
    - Processes in one container can't see or affect processes running outside the container
  - ***Control groups (cgroups) are used to allocate and manage resources***
    - Cgroups control how much memory, CPU, network and other resources are allocated to each container
- **An emerging ecosystem of products and services for easily creating, deploying and managing containers:**
    - Docker containers from Docker, Inc.
    - Higher-level container management software (Kubernetes, Swarm, etc.)

Ann.L.Chervenak@aero.org
Computer Systems Research Department

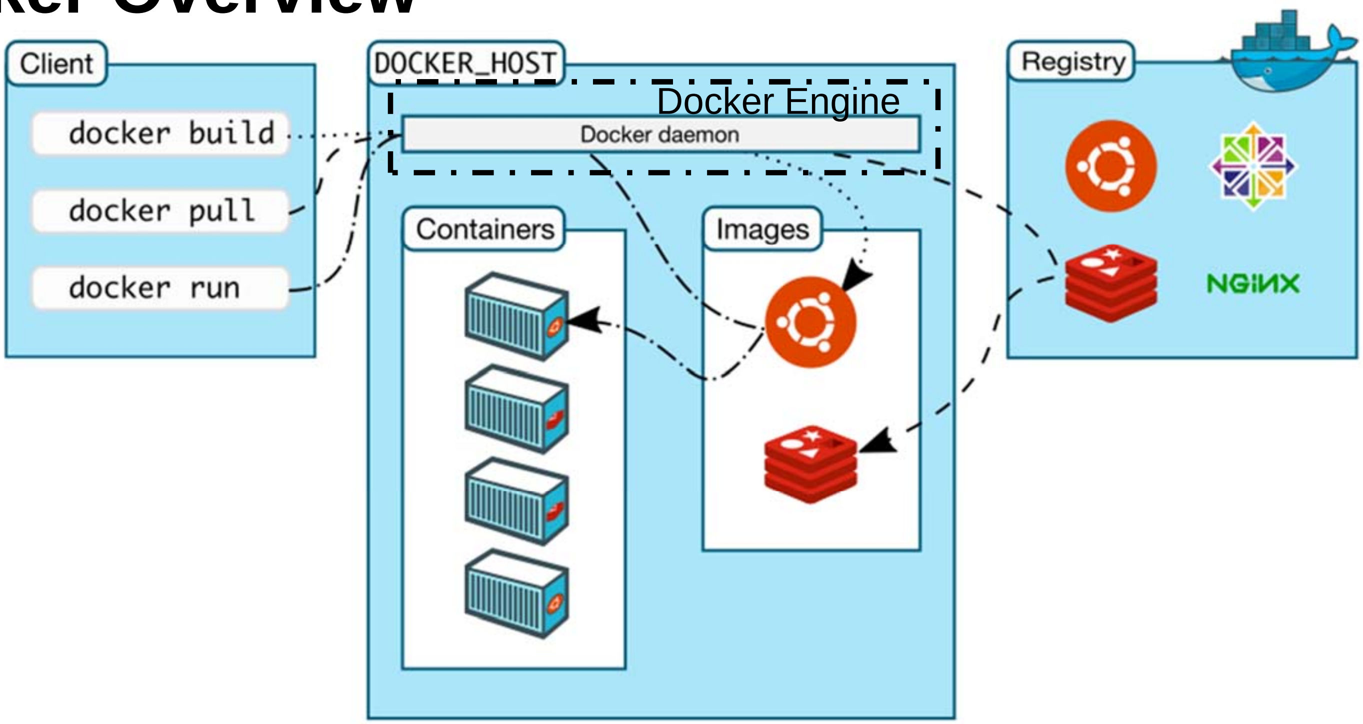
AEROSPACE

Multiple Container Implementations

- **LinuX Containers (LXC)**
  - *A set of APIs and tools that allow Linux users to create and manage containers*
- **Docker**
  - *Builds on Linux Containers (LXC), namespaces, cgroups, and other technologies*
  - *Has quickly become the **de facto industry standard for containers***
- **Rocket**
  - *Implementation of the AppContainer from the CoreOS project*
  - *Specification of a container image format, runtime, and discovery*
- **Imctfy (or "let me contain that for you")**
  - *An open source version of Google's container stack*
- **Singularity**
  - *Addresses security concerns: blocks privilege escalation within the container*

Ann.L.Chervenak@aero.org
Computer Systems Research Department

Docker Overview



- **Docker Engine** (a **container engine**) is a client-server application that consists of the long-running Docker daemon software, a REST API interface for services and a command line interface for interactive commands
- **A Docker *client*** talks to the **Docker engine**, which builds, runs, and distributes **Docker containers**
- A **Docker image** is a read-only template used to create Docker containers
- **Docker registries** are public or private repositories that hold images

Figure from: https://docs.docker.com/engine/understanding-docker/

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE

Container Management

(Also called cluster management)

Higher level software that makes using containers across a cluster of nodes easier by:

- **Scheduling containers** on multiple cloud nodes
- **Replicating** containers on multiple nodes
- **Automatically scaling** containers based on load
- **Monitoring** containers, nodes, racks, clusters
- **Providing automated recovery from container or node failures**
- **Providing security:** who is allowed to launch containers

Three players:

- ***Kubernetes*** *from Google*
- ***Swarm*** *from Docker, Inc. (now incorporated into Docker)*
- ***Mesos*** *family of products from Mesos project, Mesosphere (company)*

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE

Kubernetes Container Management

- **Developed by Google based on 15 years of experience operating their production workloads at large scale in Google data centers**
  - *Based on lessons learned from Google's Borg cluster management system*
- **Donated to community as open source**
- **Kubernetes provides capabilities to deploy, schedule, update, maintain and scale containers**
- **Monitors and manages containers to ensure that the state of the cluster meets user requirements**
- **Supports Docker and Rocket containers and will support other container image formats and runtimes as they are developed**

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE

Docker Security

- **An important security consideration is that "running containers (and applications) with Docker implies running the Docker daemon. This daemon currently requires root privileges", which creates potential security risks.**

- **Mitigation techniques:**
  - *Only trusted users should be allowed to control the Docker daemon*
  - *Use Linux kernel capabilities: containers run with a reduced capability set*
  - *Security Enhanced Linux (SELinux): supports access control policies; protects the host file system from attacks from inside the container*
  - *AppArmor: Linux kernel security module that supplements standard Linux user and group based permissions to confine programs to limited set of resources*

- **Other container formats (e.g., Rocket, Singularity) don't require root privileges for running the container**

Source: Docker Security, https://docs.docker.com/engine/security/security/

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE

Pros of Containers

- **Mature technology: standards & industry leaders emerging**
- **Lighter weight than Virtual Machines (assuming application is small relative to OS size)**
  - *Smaller, faster to deploy, more scalable*
  - *Can be deployed more densely on cloud resources, improving resource utilization and reducing power usage*
- **Containers are a key enabling technology for modern, highly scalable internet applications and microservices architecture**
- **Undergoing rapid development**
  - *Extensive industry and venture capital support*
- **Ecosystem of useful tools has been developed for containers:**
  - *Creation and re-use of container images (e.g., Docker, registries)*
  - *Container management: Deployment, replication, management of containers on clouds (e.g., Mesos, Kubernetes, Swarm)*
- **Relatively easy to adopt container technology**

Cons of Containers

- **Not yet as mature as Virtual Machines**
- **Security of containers still being improved**
  - *Docker containers require root privileges*
  - *Deploy mitigating technologies: Linux capabilities, SELinux, AppArmor*
  - *Other container implementations avoid this (Rocket, Singularity)*
- **Rapid development of container technologies: a moving target**
- **To achieve full benefits of containers, applications should be small relative to OS size**
  - *Large, monolithic legacy applications can be wrapped in containers, but won't see as much benefit*
- **Microservices architecture is a good match for containers but increases complexity**

References

- Ying Lu, "Virtual Machines," Univ. of Nebraska-Lincoln, CSCE 351: Operating System Kernels (Fall 2011), http://*cse.unl.edu/~ylu/csce351/notes/VirtualMachines.ppt*
- E. Brewer, "An update on container support on Google Cloud Platform," in *Google Cloud Platform Blog*, ed: http://googlecloudplatform.blogspot.com/2014/06/an-update-on-container-support-on-google-cloud-platform.html, June 10, 2014.
- S. Newman, *Building Microservices*: O'Reilly Media, Inc., 2015.
- J. Thones, "Microservices," *IEEE Software,* vol. 32, pp. 116-116, 2015.
- *Linux Containers*. Available: https://linuxcontainers.org/
- *Docker*. Available: https://www.docker.com/
- *Kubernetes*. Available: http://kubernetes.io/
- M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski*, et al.*, "A view of cloud computing," *Communications of the ACM,* vol. 53, pp. 50-58, 2010.
- W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," ed: IBM Research Report RC25482 (AUS1407-001), IBM Research, Austin, TX, 2014.
- D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Computing,* pp. 81-84, 2014.
- E. Brewer, "Robust Containers," DockerCon Keynote, http://www.slideshare.net/Docker/brewerdockerconkeynote-140613153819phpapp02-37588923, June 10, 2014.
- Amazon Web Services. *Amazon EC2 Container Service*. Available: https://aws.amazon.com/ecs/

Ann.L.Chervenak@aero.org
Computer Systems Research Department

References

- Red Hat. (2015). *Introduction to Linux Containers*. Available: https://access.redhat.com/articles/1353593
- *Let Me Contain That For You (lcmtfy)*. Available: http://lmctfy.io/
- N. Khare. (2014). *Linux Containers and Future of Software Delivery*. Available: Rootconf 2014 Workshop, http://neependra.net/docker/rootconfWorkshop.html
- D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal,* vol. 2014, 2014.
- Docker. *Docker Registry*. Available: http://docs.docker.com/registry/
- Docker. (2015). *Docker Security*. Available: https://docs.docker.com/articles/security/
- Intel. (2014). *Linux Containers Streamline Virtualization and Complement Hypervisor-Based Virtual Machines*. Available: http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/linux-containers-hypervisor-based-vms-paper.pdf
- A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the Linux virtual machine monitor," in *Proceedings of the Linux Symposium*, 2007, pp. 225-230.
- A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *Proceedings of the Tenth European Conference on Computer Systems*, 2015, p. 18.
- C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach*, et al.*, "Live migration of virtual machines," in *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, Boston, MA, 2005, pp. 273-286.

Ann.L.Chervenak@aero.org
Computer Systems Research Department

AEROSPACE