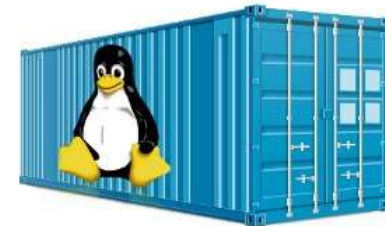




Secure Application Cloudification with Docker

Silvio Bologna, Filippo Millonzi, Diego Terrana,
Gianluca Zangara, Pietro Paolo Corso

Dipartimento di Fisica e Chimica
Università degli Studi di Palermo

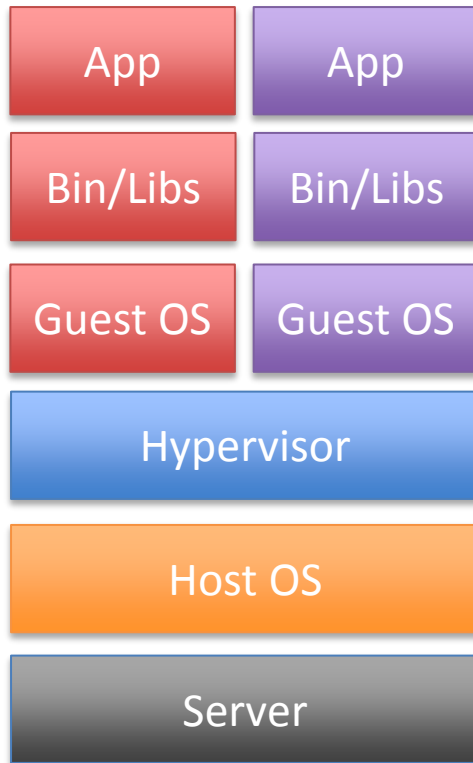


What are Containers

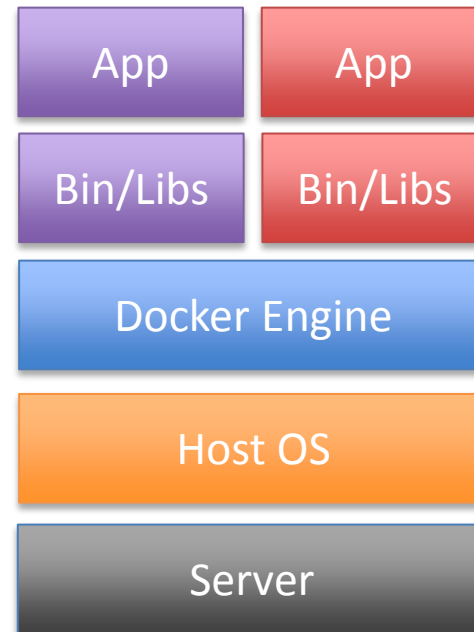
- Containers are based on a technology that allows the execution of processes in an isolated user space, sharing the same kernel.
- Created in late 90s, several Linux implementations have been used, but the advent of orchestration tools has made possible the adoption of Container technology in place of Virtual Machines.



Container VS Virtualization



Virtual Machines



Containers



Benefits of using Containers

- Portability of environments across machines
- Use for prototyping and in production environments
- Rapid application deployment – Containers include the minimal runtime requirements of the application
- Environment reuse
- Central Container repository



Container: Linux alternatives (1)

- **Linux V-Server:** uses a patched Linux kernel to emulate several Containers (also with different kernels) within *Contexts*. To the contexts there are associated resource quotas. V-Server also manages network limiting
- **OpenVZ:** patches the original Linux Server, runs Virtual Environments (VEs) with resource limiting and namespaces. Has three command line interfaces

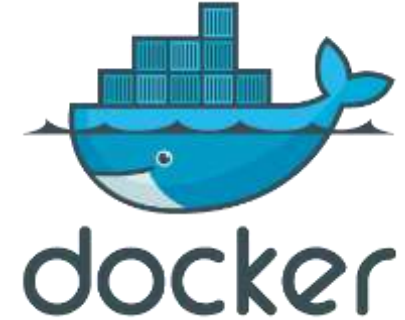


Container: Linux alternatives (2)

- **LXC:** fully runs on unmodified Linux Kernel, consists of several tools. This technology manages process, network and file system isolation (namespaces), isolated directory tree (chroots) and resource quotas (CGroups)
- **LXD:** evolution of LXC with enhanced capabilities for running Containers in several machines, allowing live migration and virtualizing different kernel (i.e. RHEL on Ubuntu)



Docker

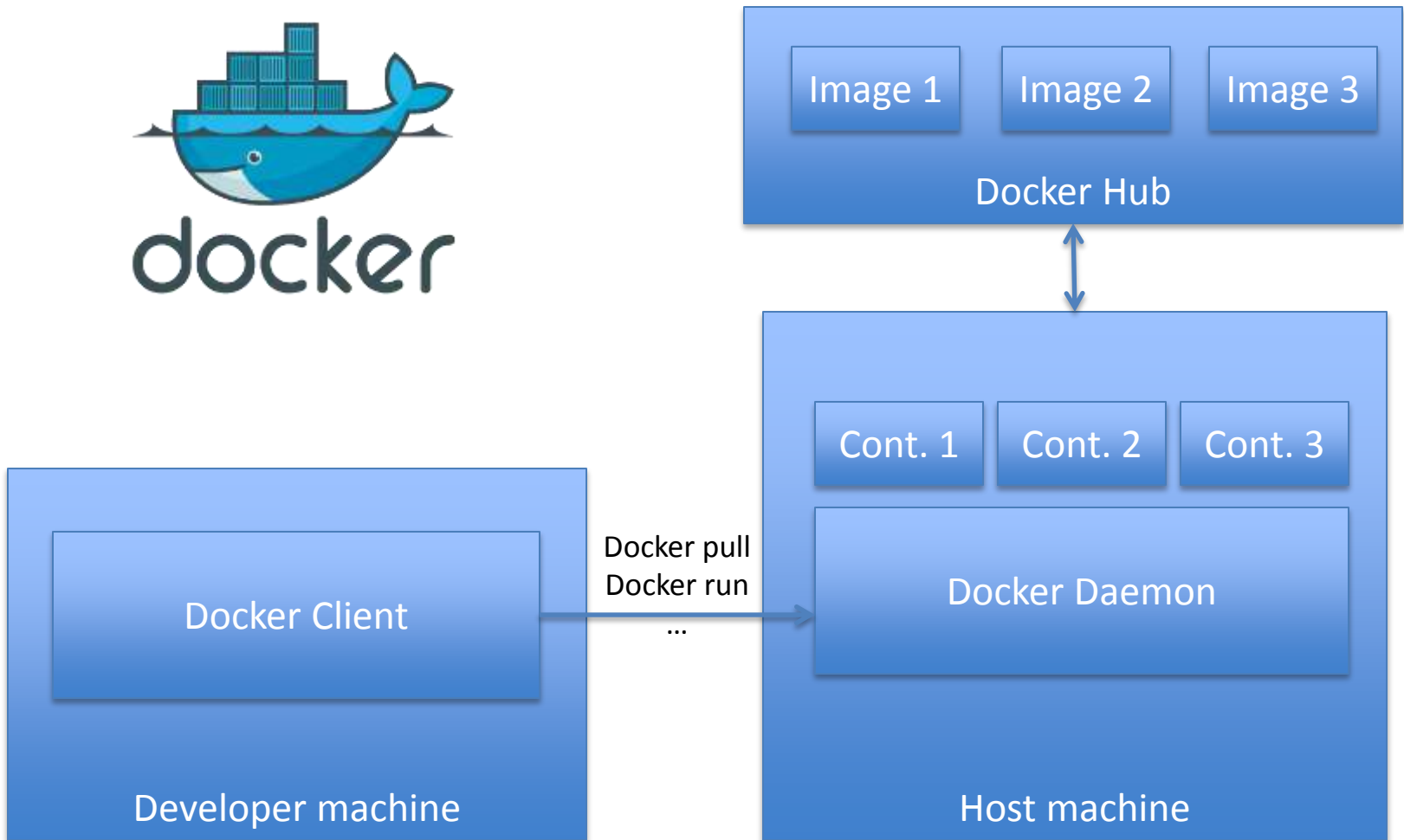


- Orchestration of deployment and execution of Containers
- Docker Engine
- Docker Hub
- LXD (Integration with OpenStack) or LXC
- Client tools





Docker architecture





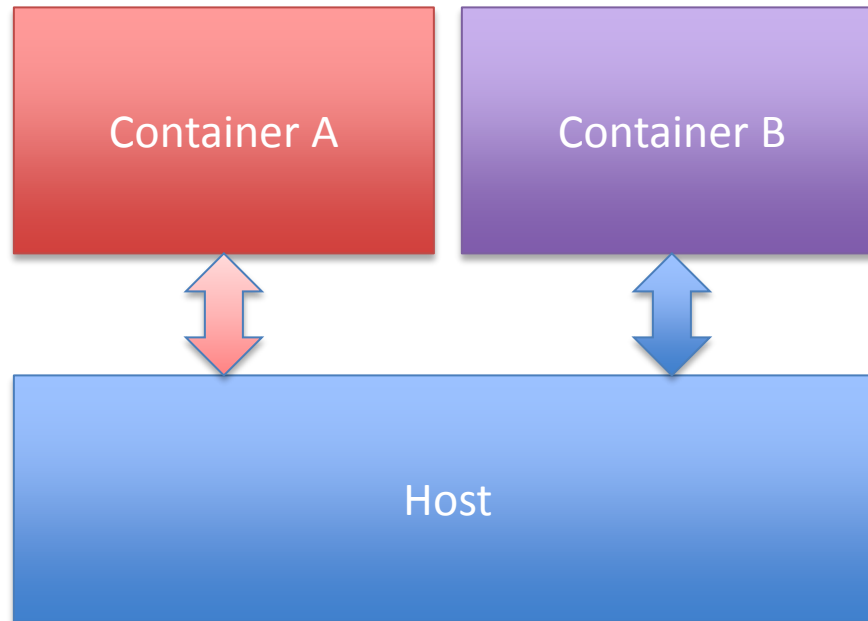
Running application over the Cloud: legal concerns

- Lack of precise legislation, all included in the contract between provider and customer
- Cloud provider must make the “best effort” not to loose customer’s data
- Configuration of software modules are critical for ensuring security of the runtime environment



Securing Container execution (1)

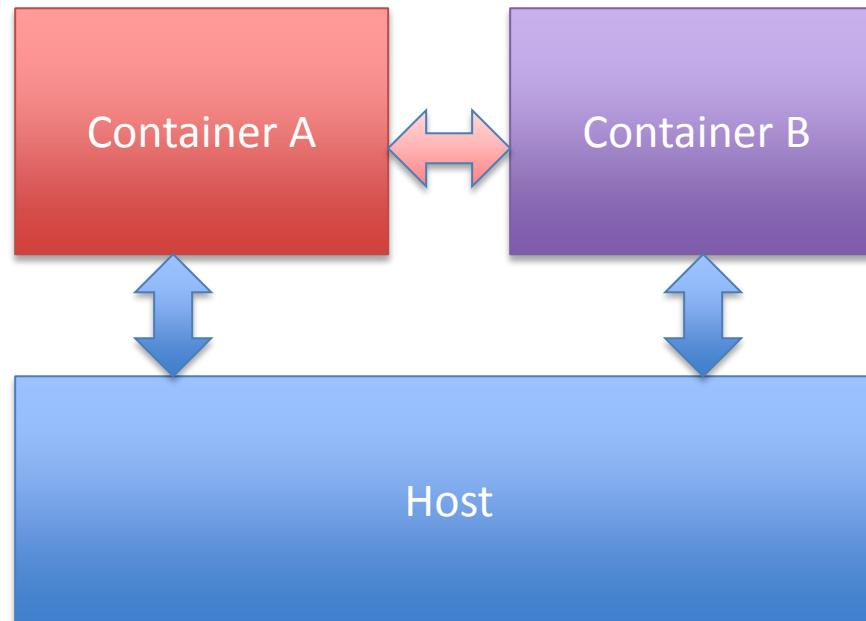
- Container to host attack





Securing Container execution (2)

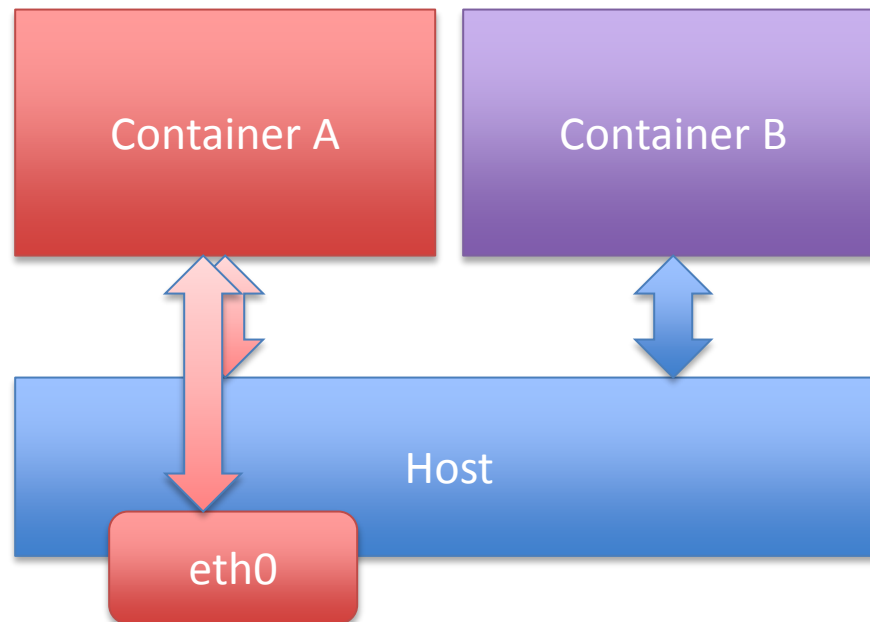
- Container to Container attack





Securing Container execution (3)

- Network attack: capability to perform Denial of Service, cause the physical network interface is shared









Cgroups, Namespaces, chroots

- **Cgroup:** resource management and accounting solution *per process group*
- **Namespace:** provides a view of a subset of system resources (files, ipc, processes, network, user groups)
- **Chroot:** change the root folder of a specific process, allowing to use other libraries and binaries than the host system



Enforcing security

- SELinux  
- AppArmor  
- Both systems ensure resource isolation via some kind of namespace
- Pros and cons: SELinux is more granular assigning policies, AppArmor is more manageable



SELinux

- Security-Enhanced Linux (SELinux) is a Linux component that provides a variety of security policies for Linux kernel
- Attaches labels to all files, processes and, in general, all Linux *resources*
- ✓ Well-defined policy interfaces
- ✓ Caching of access decisions for efficiency
- ✓ Separate measures for protecting system integrity (domain-type) and data confidentiality (multilevel security)
- ✓ Controls over use of "capabilities"
- ❖ Not all applications preserve labels
- ❖ It has a complex policy language, it is hard to manage rules also because there aren't integrated tools or GUIs



AppArmor

- AppArmor (Application Armor) is a security software for Linux which is maintained and released by Novell under GPL
- Was created as an alternative to SELinux
- Pathname based system does not require labeling or relabeling filesystem
- Auditable policies
- ✓ Integrated GUI/Console toolset
- ✓ Protects the operating system, custom and third-party applications from both external and internal threats by enforcing appropriate application behavior
- ✓ Reporting and alerting
- ❖ Not fine as SELinux



SELinux/AppArmor & Docker

- To enable SELinux/AppArmor support, use `–security-opt` command when launching.
- *`docker run –security-opt label:type:label_value -i -t centos \ bash`*
- Specific SELinux options are:
 - `–security-opt="label:user:USER"` (set label user)
 - `–security-opt="label:role:ROLE"` (set label role)
 - `–security-opt="label:type:TYPE"` (set label type)
 - `–security-opt="label:level:LEVEL"` (set label level)
- Specific AppArmor options are instead:
 - `–security-opt="apparmor:PROFILE"` (set AppArmor profile)



Best practices

- Don't enable SSH access in Containers
 - Connect via SSH to the host machine and connect to Container via Docker client
- Don't run privileged Containers
 - Rather prefer setting appropriate policies to grant access to devices / files
- Control Docker binaries access within host O.S.
 - A user with privileges granted to Docker binaries can allow him run/stop Containers
- Running smaller units of software, to control them is easier