

Linux containers

Bruno Barcarol Guimarães

bbgstb@gmail.com

2014-12-06

- 1 Visão geral
 - Tecnologias
 - Vantagens e desvantagens
- 2 Implementação
 - Kernel
 - Namespaces
- 3 Segurança
 - uid 0
 - Aplicações
- 4 Ferramentas
 - systemd-nspawn
 - lxc
 - docker
- 5 Referências

Container

- um grupo de processos
- sobre um mesmo *kernel*
- isolado dos outros processos

Linux

cgroups

namespaces

netlink

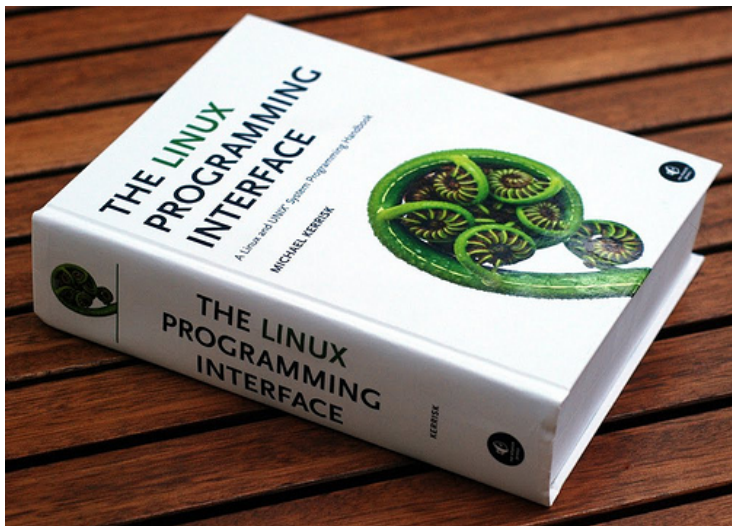
selinux

netfilter

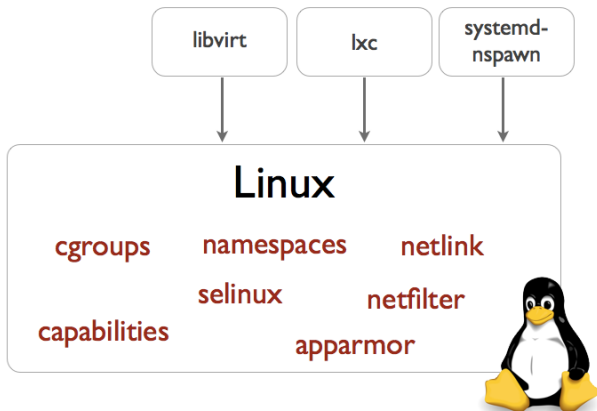
capabilities

apparmor

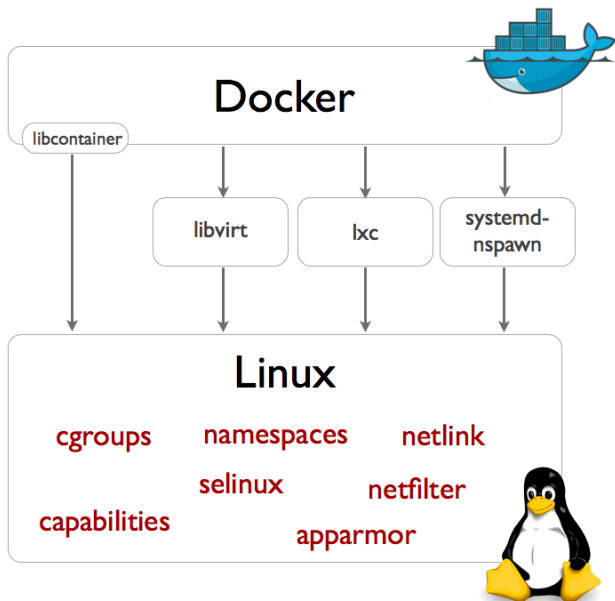




Visão geral - Tecnologias



Visão geral - Tecnologias



Visão geral - Tecnologias



Visão geral - Tecnologias

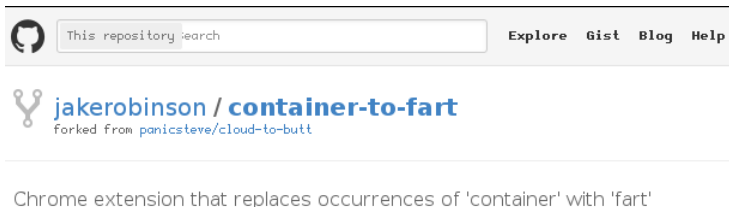


Visão geral - Tecnologias



- *kernel* compartilhado
 - *boot* mais rápido (*ms*)
 - menor *overhead*
 - menor utilização de recursos
 - uma única atualização (*kpatch/ksplce/live patching?*)

- *kernel* compartilhado
 - apenas um tipo (arquitetura, sistema operacional)
 - apenas uma versão
 - *kernel exploits*
 - segurança não era o objetivo principal



The screenshot shows the top section of a GitHub repository page. At the top left is the GitHub logo. Next to it is a search bar with the placeholder text "This repository search". To the right of the search bar are links for "Explore", "Gist", "Blog", and "Help". Below the search bar, the repository name "jakerobinson / container-to-fart" is displayed in a large blue font. Underneath the repository name, it says "forked from panicsteve/cloud-to-butt". A horizontal line separates this header from the main content area, which contains the text "Chrome extension that replaces occurrences of 'container' with 'fart'".

This repository search

Explore Gist Blog Help

 **jakerobinson / container-to-fart**
forked from panicsteve/cloud-to-butt

Chrome extension that replaces occurrences of 'container' with 'fart'

- cgroups
 - limite/reserva de recursos
 - *accounting*
- namespaces
 - clone(2)
 - unshare(2)
 - setns(2)

Namespaces

- mnt (CLONE_NEWNS)
- uts (CLONE_NEWUTS)
- ipc (CLONE_NEWIPC)
- pid (CLONE_NEWPID)
- net (CLONE_NEWNET)
- uid (CLONE_NEWUSER)

Namespaces - mount

- CLONE_NEWNS
- linux 2.4.19
- mount(2)/umount(2)
- processos diferentes têm visões diferentes do sistema de arquivos
- “chroot(2) *on steroids*”
- compartilhamento de *mount points*

Namespaces - uts

- CLONE_NEWUTS
- linux 2.6.19
- `uname(2)`/`sethostname(2)`/`setdomainname(2)`

Namespaces - ipc

- CLONE_NEWIPC
- linux 2.6.19 / linux 2.6.30
- `svipc(7)/mq_overview(7)`

- CLONE_NEWPID
- linux 2.6.24
- processos em *containers* diferentes podem ter o mesmo pid
- processos só vêem outros processos do mesmo *namespace*
- migração entre *hosts*
- múltiplos pid 1
- mapeamento de pid
- podem ser aninhados

Namespaces - net

- CLONE_NEWNET
- linux 2.6.24
- cada *namespace* tem seus próprios
 - dispositivos de rede
 - endereços ip
 - tabelas de roteamento
 - /proc/net
 - portas
 - etc

- CLONE_NEWUSER
- linux 2.6.23
- iniciado no kernel 2.6.23
- finalizado no kernel 3.8
- ~ cinco anos

- isolamento de uid e gid
- mapeamento de uid e gid
- uid 0

Recursivos

- um processo sem privilégios pode criar um *namespace*
- uid 0 dentro do *namespace*
- ö

- não use

A maioria dos *containers* executa uma tarefa específica, ao invés de um sistema completo, e a maioria dessas aplicações (apache, postgresql, mongodb, redis, ...) não precisa de privilégios de root. Os riscos são os mesmo que existiram até hoje: assumir que a aplicação pode fazer qualquer coisa para escapar do isolamento.

syscalls

- e.g. `vmsplice(2)`
- limitar as *syscalls* disponíveis
- `seccomp/seccomp-bpf`
- *capabilities*
- `grsec`
- atualizações frequentes

= reduzir a área de exposição do kernel

Outros *containers*

- *bug* no código dos *namespaces*
- *filesystem leak*

- uid *container* — > uid diferente no *host*
- selinux

"[...] there's maybe marginal increases in practical security for certain kinds of deployment, and perhaps marginal decreases for others. We end up coming back to the attack surface, and it seems inevitable that that's always going to be larger in container environments. The question is, does it matter? If the larger attack surface still only results in one more vulnerability per thousand years, you probably don't care. The aim isn't to get containers to the same level of security as hypervisors, it's to get them close enough that the difference doesn't matter."

Matthew Garrett

- Namespaces in operation
- Linux Container Security
- Docker, Linux Containers (LXC), and security
- vmsplice(): the making of a local root exploit
- seccomp
- grsec

Referências (imagens)

- <http://www.servicioswebgratis.com/wp-content/uploads/2012/08/the-linux-programming-interface.jpg>
- <http://www.mccarthyboas.com/BetsyAcyrilcSnakeCage02.JPG>
- <http://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/>
- <http://tchelinux.org/media/tchelinux.svg>