# Linux containers

Bruno Barcarol Guimarães

*bbgstb@gmail.com*

2015-07-10

# Resumo

# Visão geral - Container

container != vm != lxc != docker

# Visão geral - Container

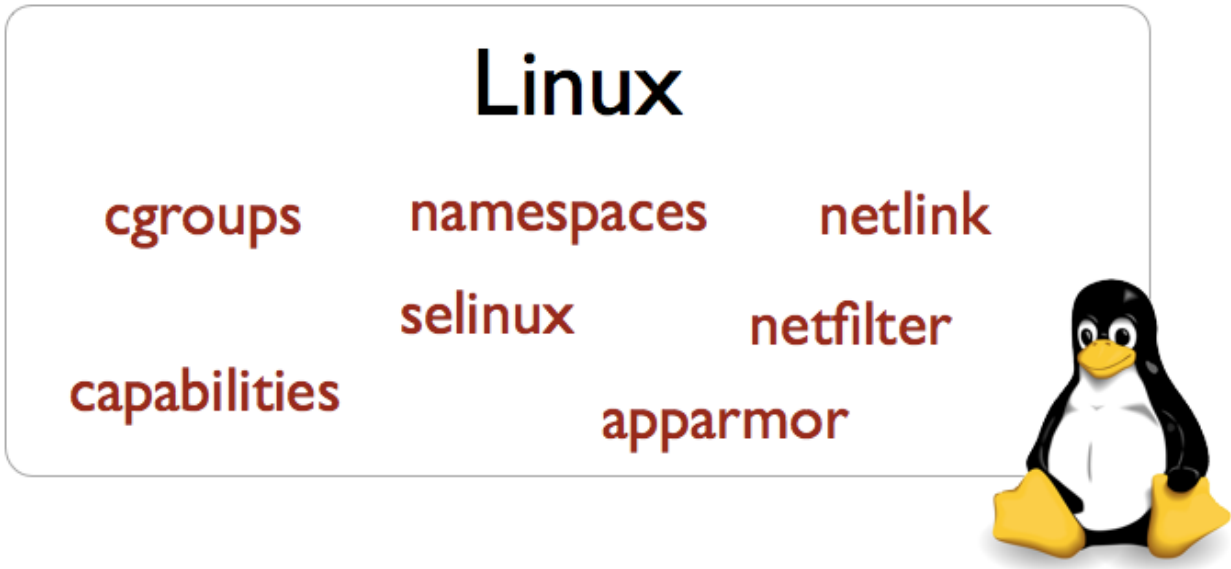
container != vm != lxc != docker != **solução de todos os problemas**

# Visão geral - Container

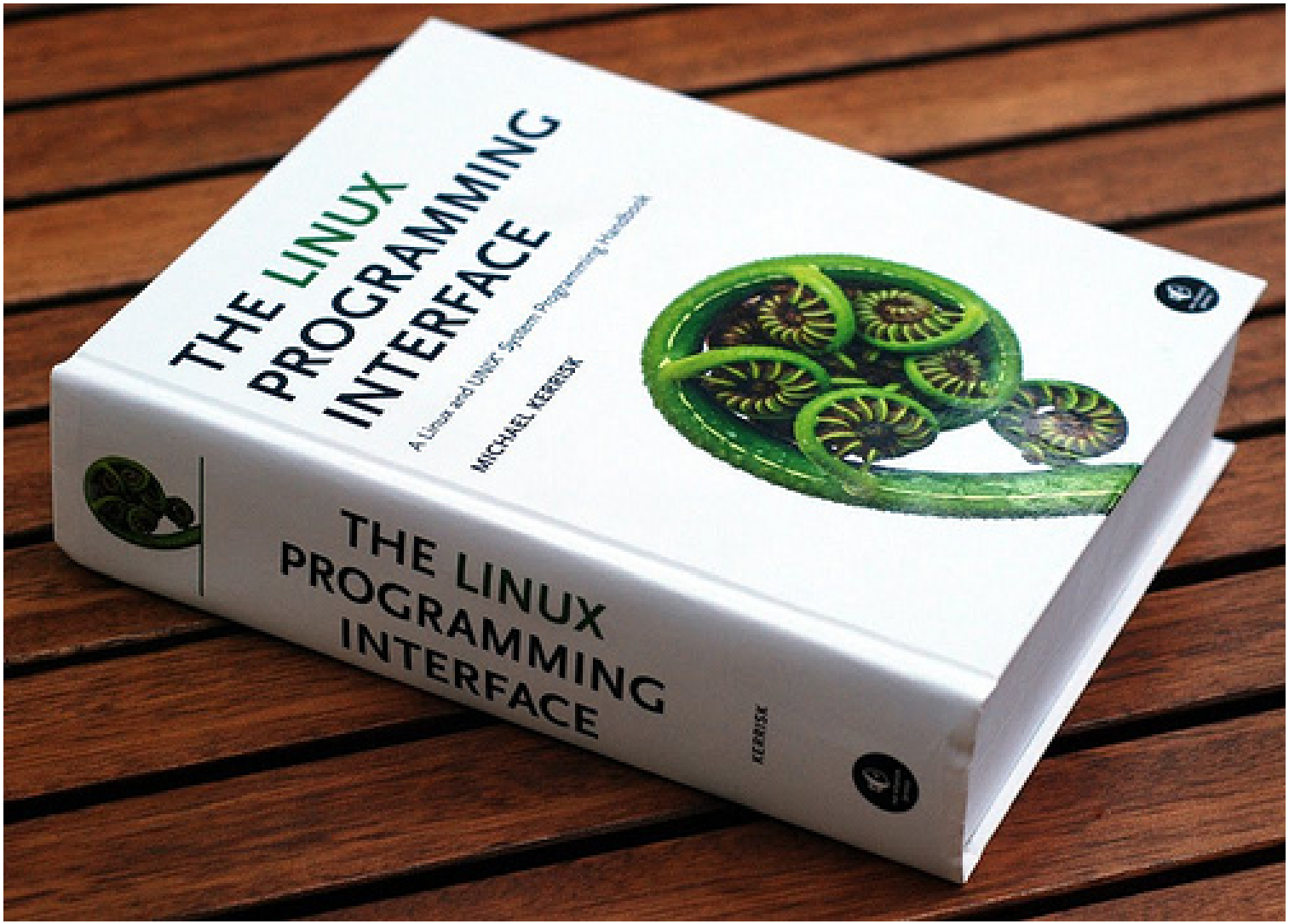
Container

- um processo (ou grupo de processos)
- executando em um mesmo *kernel* (*i.e.* o kernel não é virtualizado)
- com diferentes níveis de isolamento entre os outros processos e recursos desse *kernel*

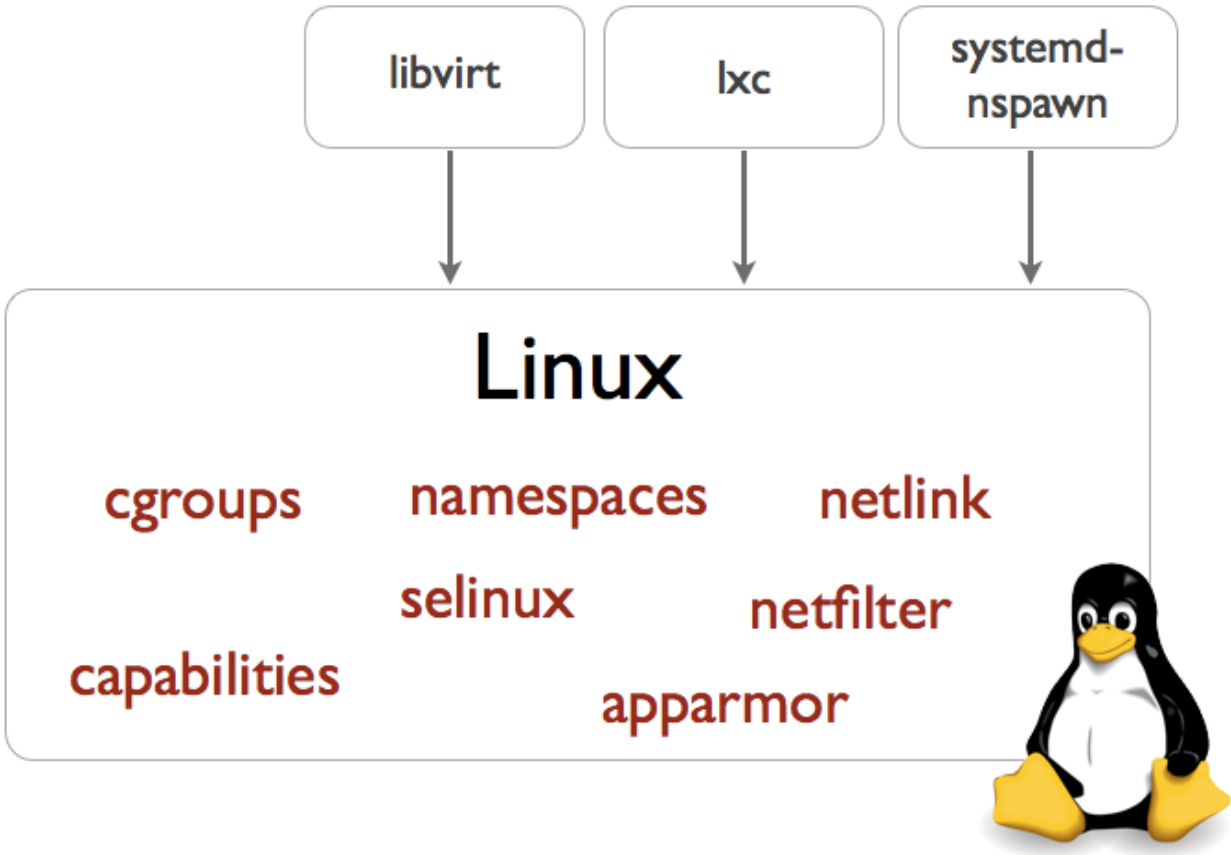
# Visão geral - Tecnologias



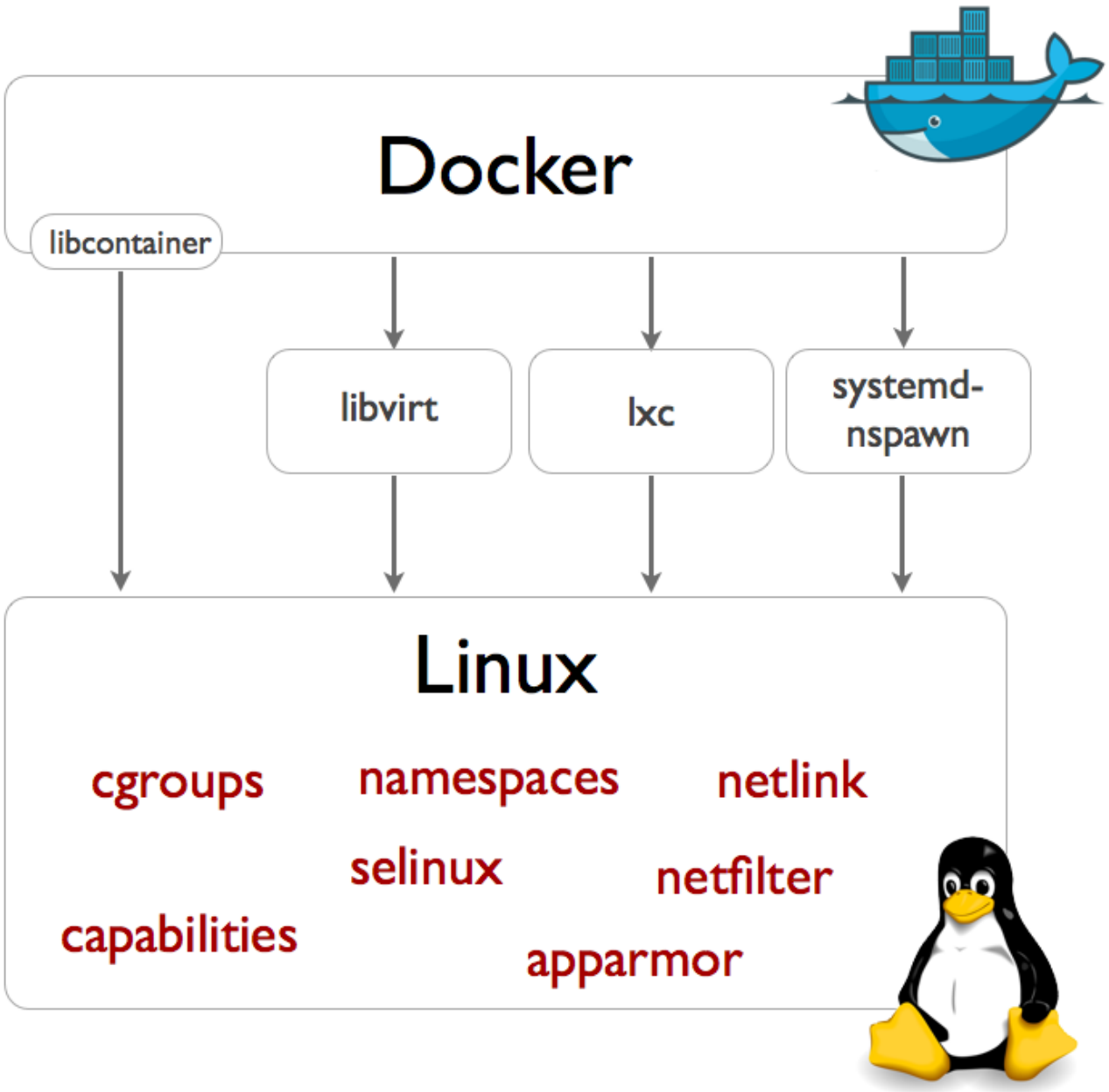
# Visão geral - Tecnologias



# Visão geral - Tecnologias



# Visão geral - Tecnologias



# Visão geral - Vantagens

- tecnologia nova e em grande expansão
- *kernel* compartilhado
  - *hardware* não é virtualizado
  - inicialização mais rápida (*ms*)
    - menor *overhead*
    - menor utlização de recursos
  - uma única atualização (*kpatch*/*ksplice*/*live patching*?)

# Visão geral - Desvantagens

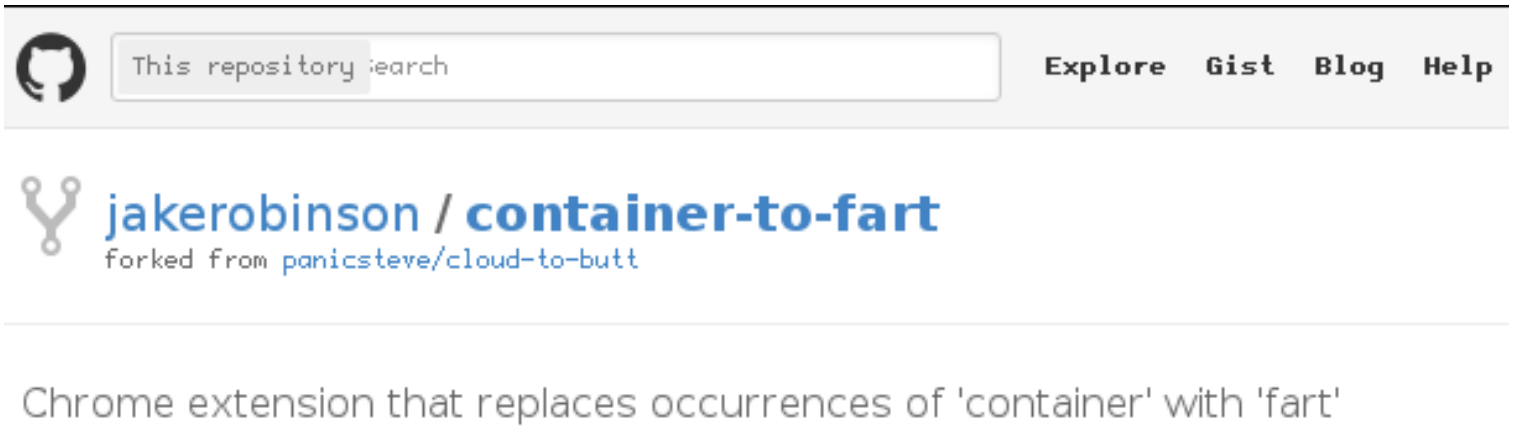
- tecnologia nova e em grande expansão
- *kernel* compartilhado
  - *hardware* não é virtualizado
  - apenas um tipo
    - versão
    - arquitetura
    - sistema operacional
  - *kernel exploits*

# Visão geral

*There's an additional advantage to containerizing your application. It forces you to think hard about configuration, limiting the amount of mutable state inside your environment and your ability to scale horizontally. An exercise in switching to container-based deploys is actually an exercise in good engineering practices and any extra work required by Docker pays off by making your codebase better factored and less brittle. Onwards to excellence, riding the blue container whale!*

Jan Urbański - New Relic

# Visão geral


This repository earch
Explore
Gist
Blog
Help
jakerobinson / container-to-fart
forked from panicsteve/cloud-to-butt
Chrome extension that replaces occurrences of 'container' with 'fart'

# Implementação - Kernel

- cgroups
- namespaces

# cgroups

- linux 2.6.24 (2007)
- limite/reserva de recursos
- *accounting*
- *audit*

# 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 (2002)
- `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 (2006)
- `uname(2)`/`sethotname(2)`/`setdomainname(2)`

# Namespaces - ipc

- `CLONE_NEWIPC`
- linux 2.6.19 (2006) / linux 2.6.30 (2009)
- `svipc(7)`/`mq_overview(7)`

# Namespaces - pid

- `CLONE_NEWPID`
- linux 2.6.24 (2008)
- 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 (2008)
- cada *namespace* tem seus próprios
  - dispositivos de rede
  - endereços ip
  - tabelas de roteamento
  - `/proc/net`
  - portas
  - etc

# Namespaces - user

- `CLONE_NEWUSER`
- linux 2.6.23 (2007)
- finalizado no kernel 3.8 (2013) $->$ $\sim$ cinco anos
- isolamento e mapeamento de uid e gid
- `uid 0`
- recursivos
  - um processo sem privilégios pode criar um *namespace*
  - `uid 0` dentro do *namespace*
  - ö

# Segurança - uid 0

- não use

# Segurança - uid 0

> *For repos we recognize on or after 2015-01-01, linux builds are sent to our container-based infrastructure.*

Travis CI

# Segurança - uid 0

*This job is running on container-based infrastructure, which does not allow use of 'sudo', setuid and setguid executables. If you require sudo, add 'sudo: required' to your .travis.yml*

Travis CI

# Segurança - Aplicações

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.

# Segurança - Aplicações

syscalls

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

= reduzir a área de exposição do kernel

# Segurança - Aplicações

- uid *container* $->$ uid diferente no *host*
- selinux/apparmor
- docker (experimental)

# Segurança - Aplicações

*"[...] 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

# Ferramentas - Demos

- systemd-nspawn
- lxc
- docker

# Ferramentas - Futuro

- Software livre
- OCP (*Open Container Project*)
- FreeBSD docker port
- rkt/runc
- puppet/ansible ("Glorified Shell Script")
- live migration (CRIU)

# Referências


- Zero to docker
- cgroup docs
- Namespaces in operation
- Linux Container Security
- Docker, Linux Containers (LXC), and security
- vmsplice(): the making of a local root exploit
- seccomp
- grsec
- Open Container Project
- App Container Specification

# Referências (imagens)


- http://www.servicioswebgratis.com/wp-content/uploads/2012/08/the-linux-programming-interface.jpg
- http://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/