DaoCloud

洞悉DOCKER安全史

孙宏亮

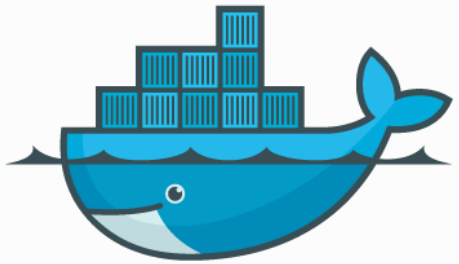
DaoCloud

allen.sun@daocloud.io

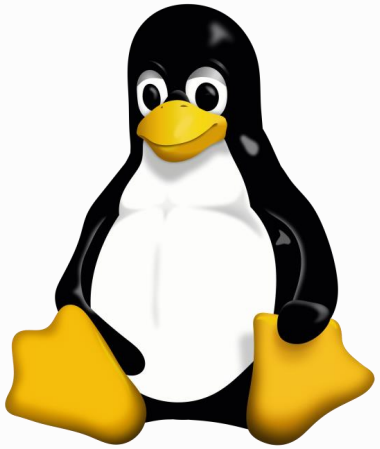
ABOUT ME

- DaoCloud
- Docker
- Linux
- GitHub: allencloud

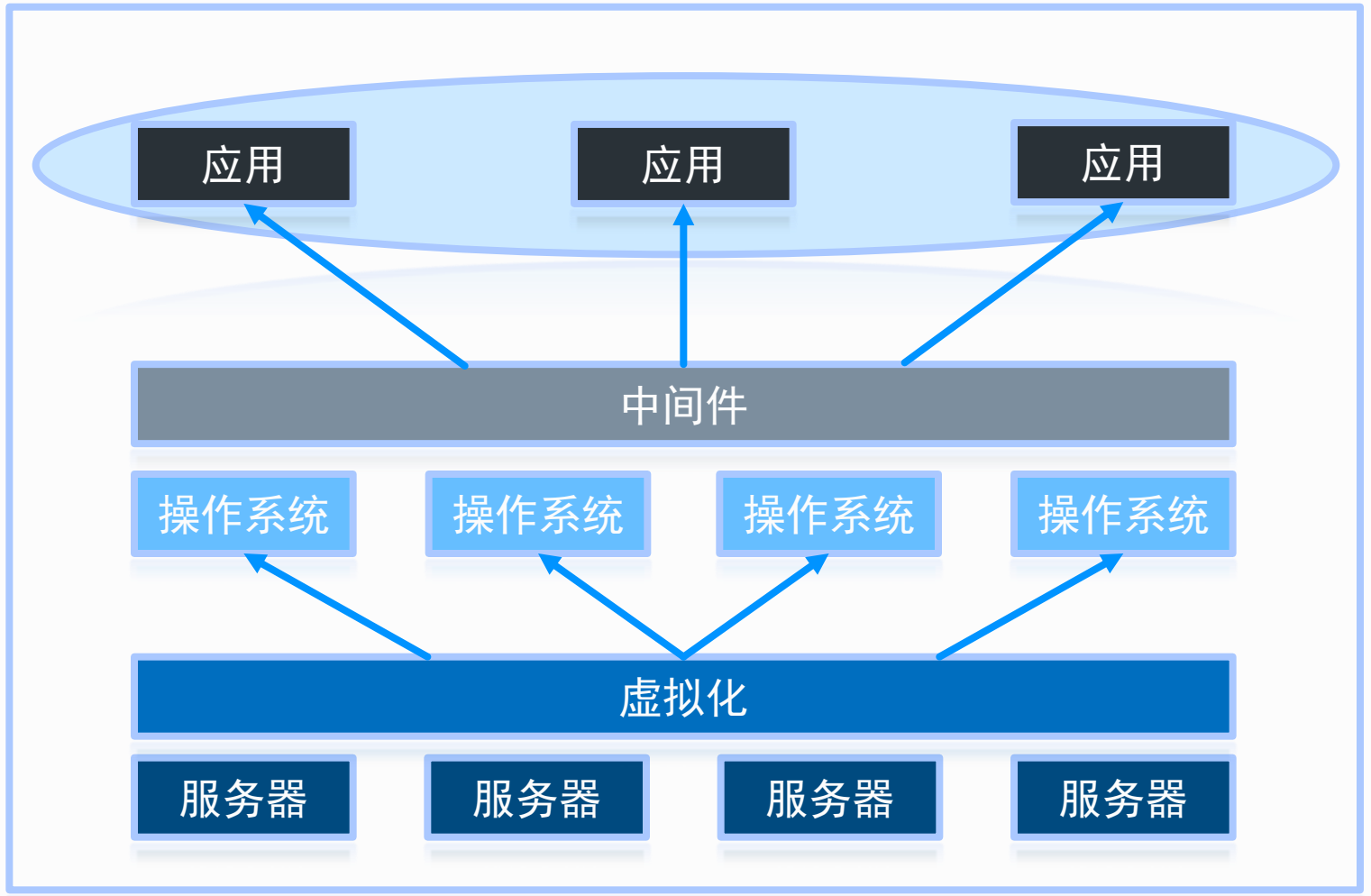
DaoCloud



docker



企业级IT架构



Dev

MQ/Cache

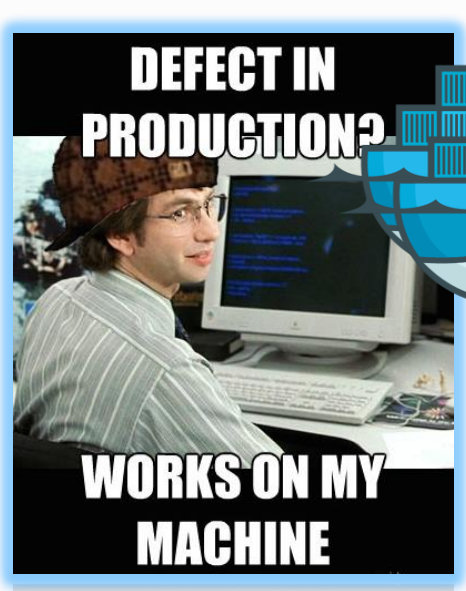
RHEL/Windows

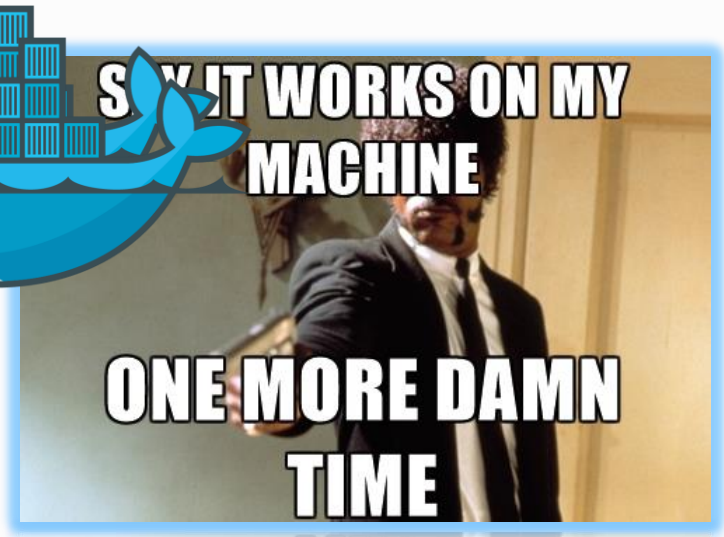
Ops

VMWare/KVM

X86/Power

企业应用交付流程





开发 → 测试 → 运维

DOCKER不为安全而生

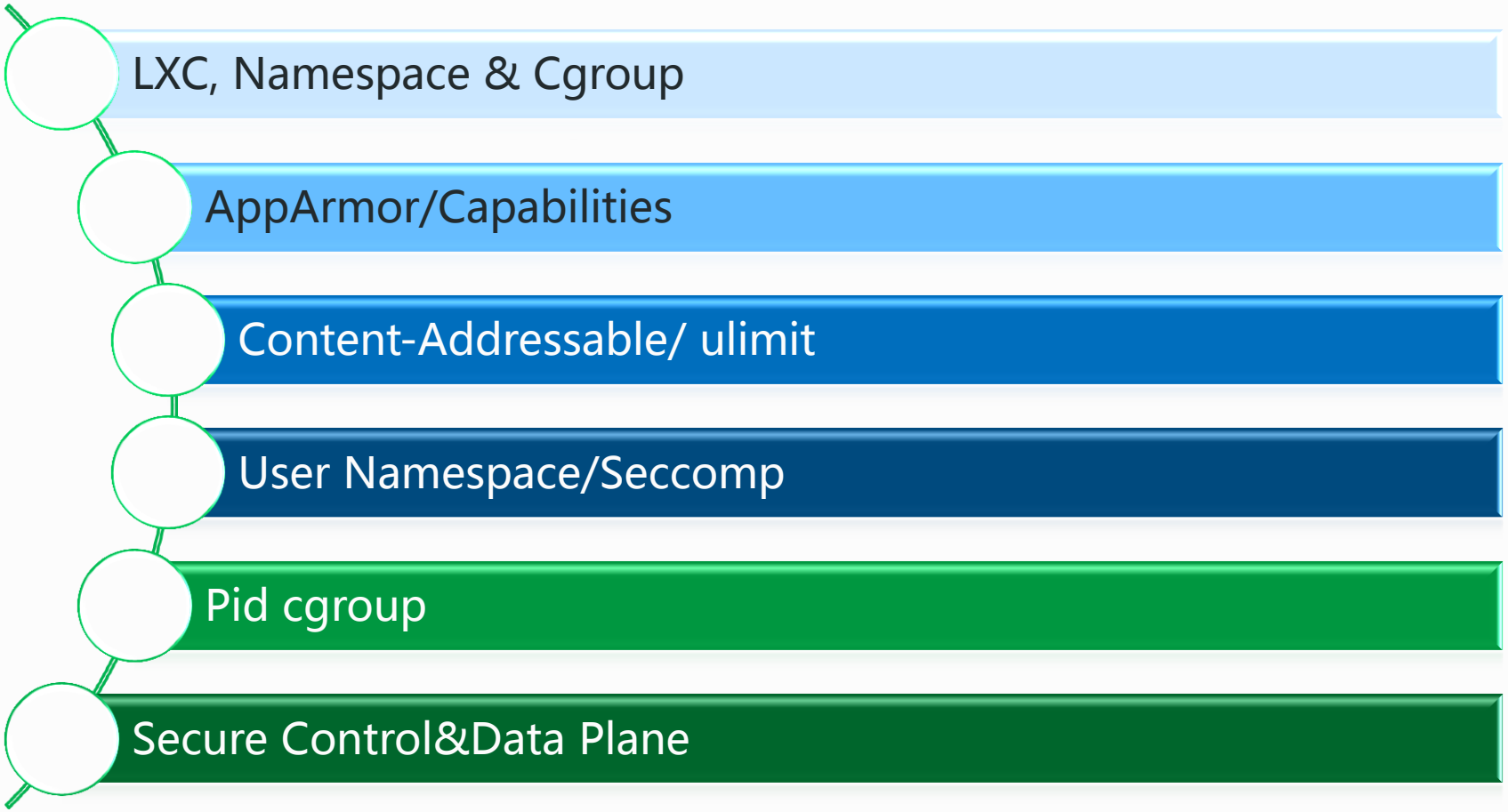
不可回避！

易用性　　标准化　　性能

外界对DOCKER的安全担忧

- 没有 VMs 安全
- 降低安全性获得性能
- 容器内root就是宿主机root
- Cgroup/Namespace不安全
- 网络隔离弱
- ……

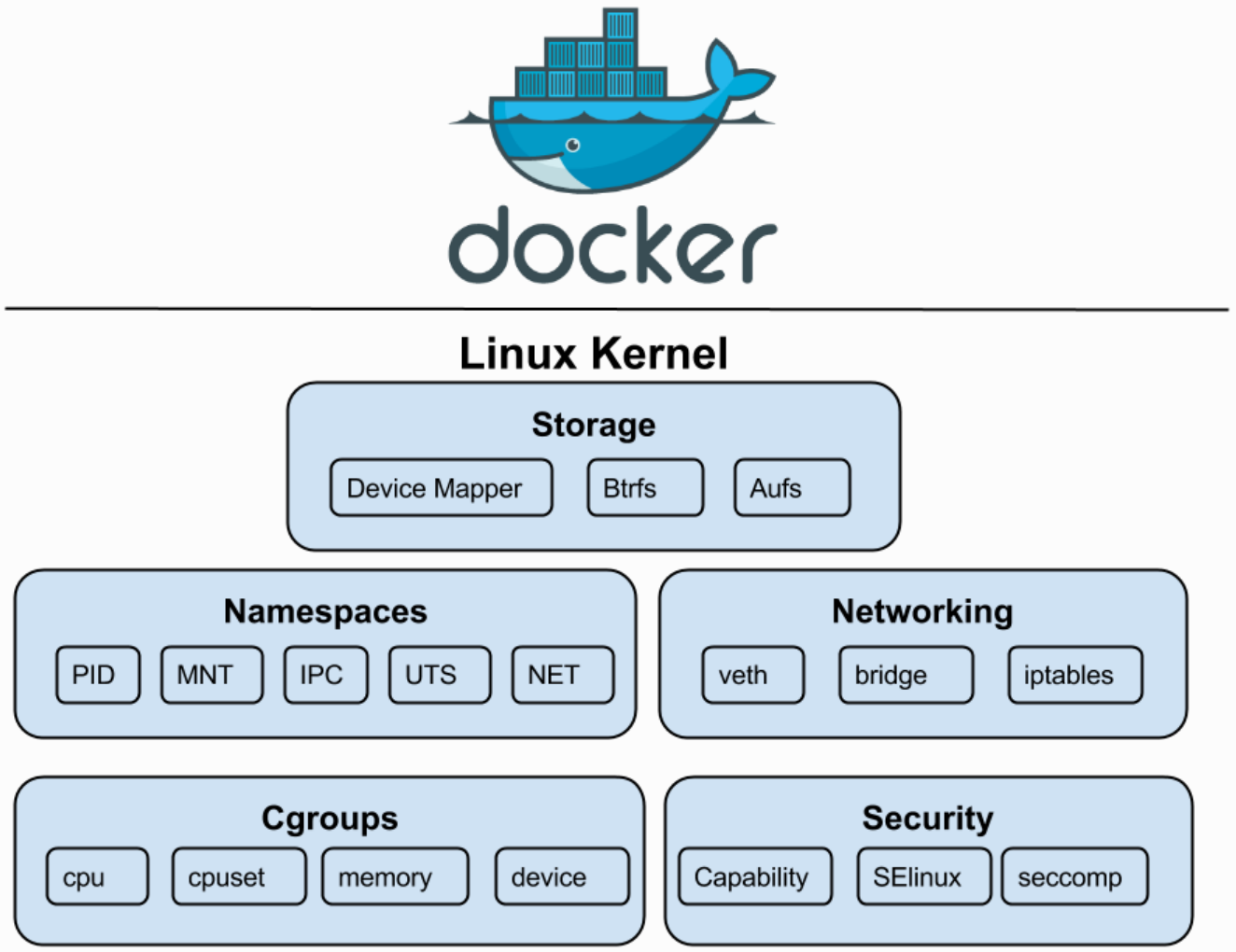
安全发展史

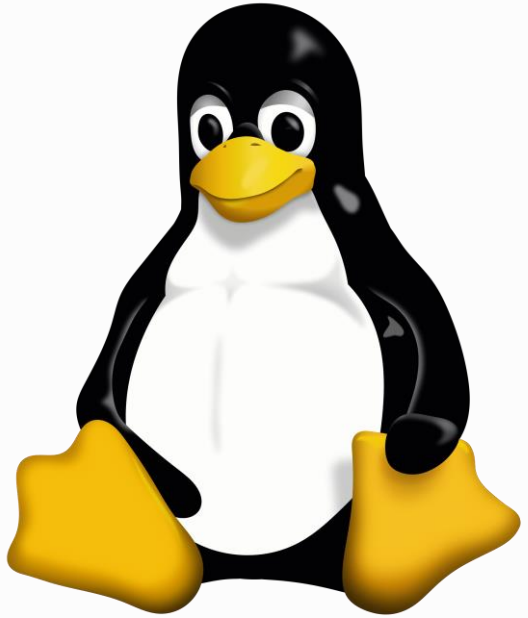

LXC, Namespace & Cgroup
AppArmor/Capabilities
Content-Addressable/ ulimit
User Namespace/Seccomp
Pid cgroup
Secure Control&Data Plane

安全发展史

- 通信安全
  - Client－Engine
  - Engine－Registry
- 镜像安全
- 容器安全
- 网络安全

DOCKER & LINUX



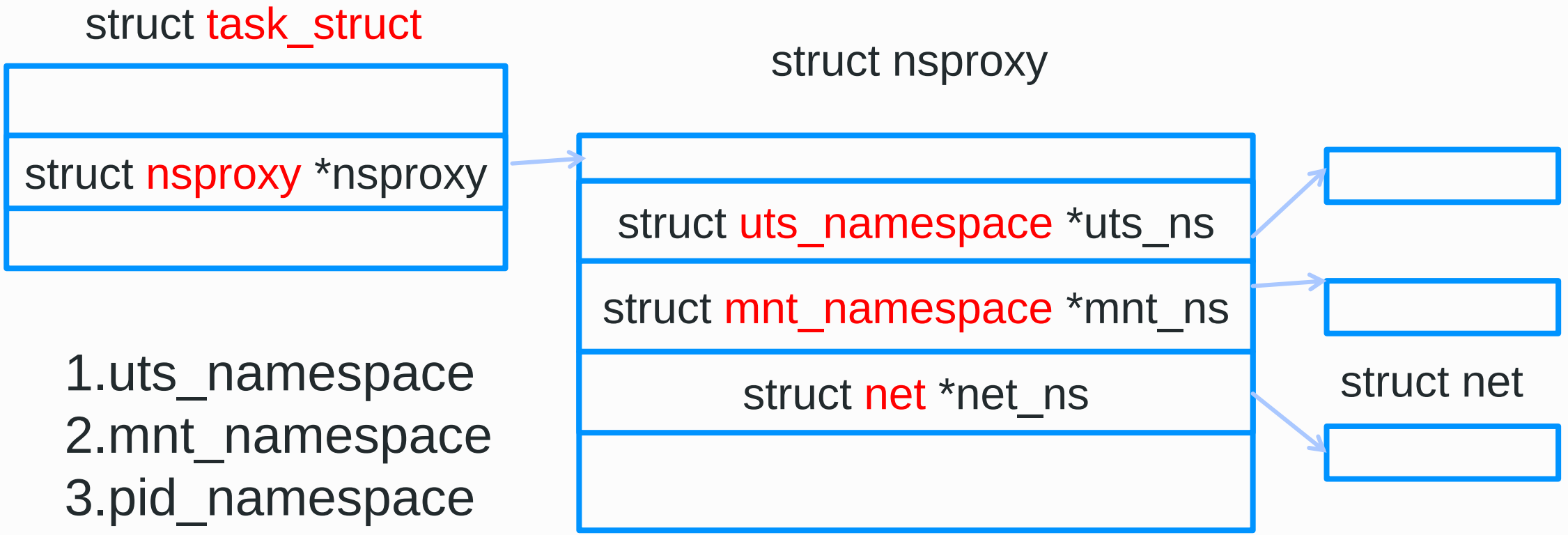


NAMESPACE

隔离系统资源

mnt (mount points, filesystems)	*CLONE_NEWNS*
pid (processes)	*CLONE_NEWPID*
net (network stack)	*CLONE_NEWNET*
ipc (System V IPC)	*CLONE_NEWIPC*
uts (unix timesharing - domain name, etc)	*CLONE_NEWUTS*
user (UIDs)	*CLONE_NEWUSER*

TASK_STRUCT AND NAMESPACES



1.uts_namespace
2.mnt_namespace
3.pid_namespace
4.ipc_namespace
5.net

WHAT IS IN NAMESPACES?

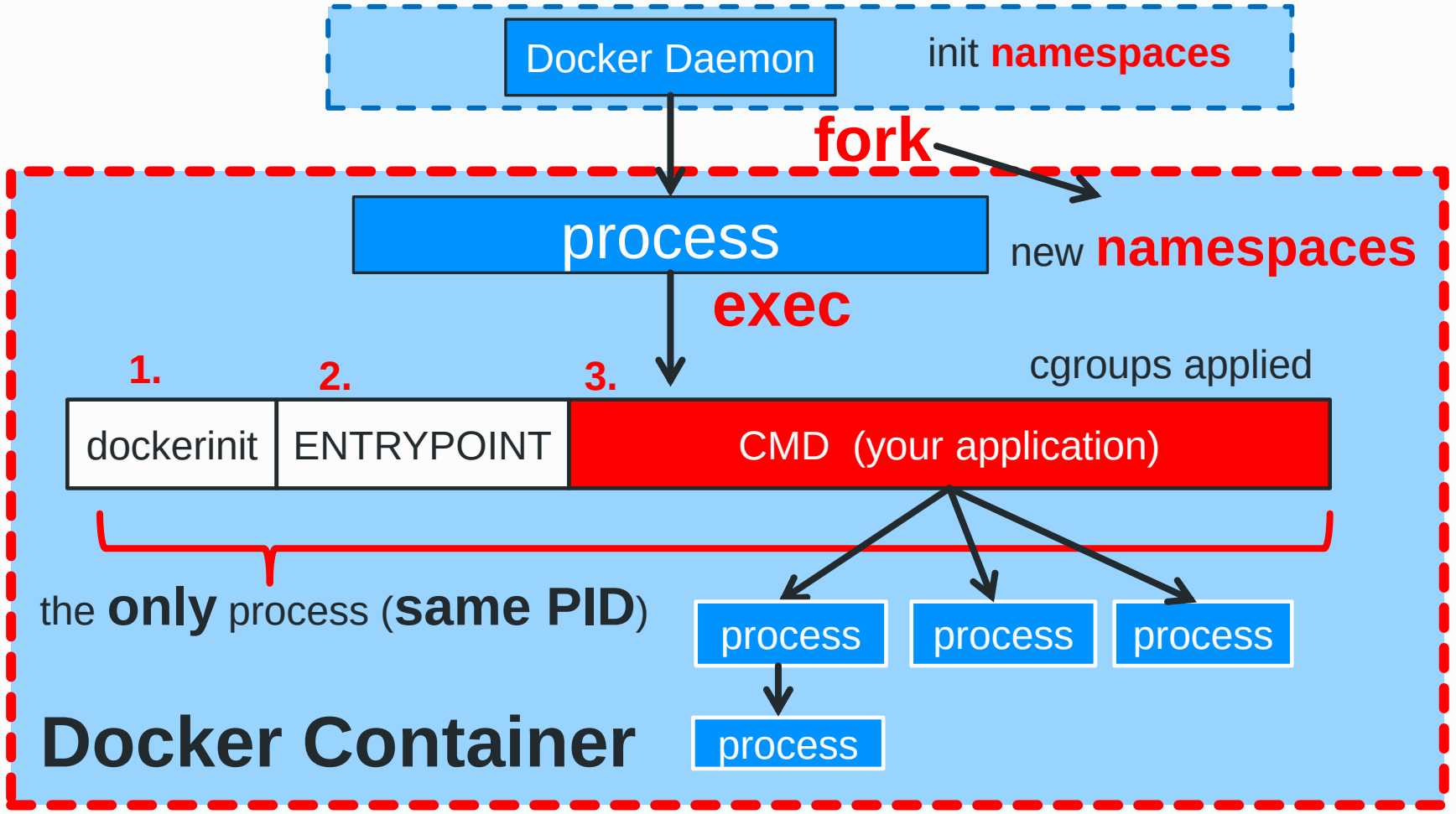
```
struct pid_namespace {
        …
        struct task_struct *
child_reaper;
        …
        int level;
        struct pid_namespace
*parent;
};
```

```
struct mnt_namespace {
        atomic_t count;
        struct mount *root;
        struct list_head list;
        ……
};
```

```
struct uts_namespace {
        struct kref kref;
        struct new_utsname
name;
        struct user_namespace
*user_ns;
}
```

```
struct new_utsname {
        char sysname[..];
        char nodename[..];
        char release[..];
        char version[..];
        char machine[..];
        char domainname[..];
};
```

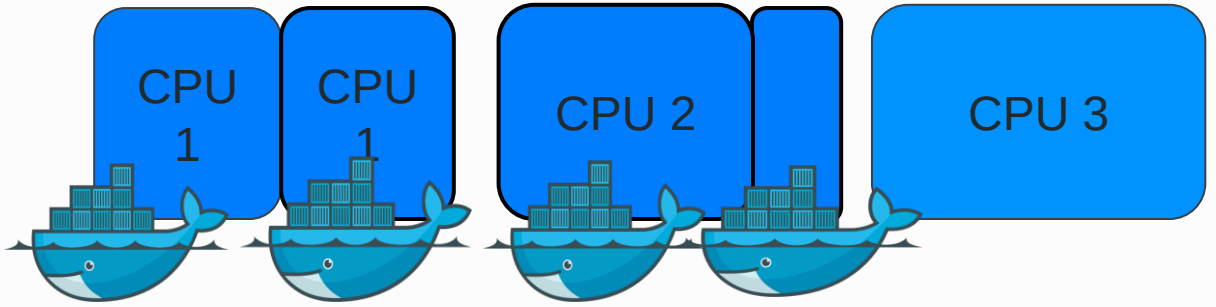
Based on Linux kernel 3.13



文件系统 / Pid / 网络 / IPC …

CGROUP(CONTROL GROUP)

- 控制内存使用
- 控制CPU分配时间
- 控制磁盘I/O
- 控制设备访问权限
- ……



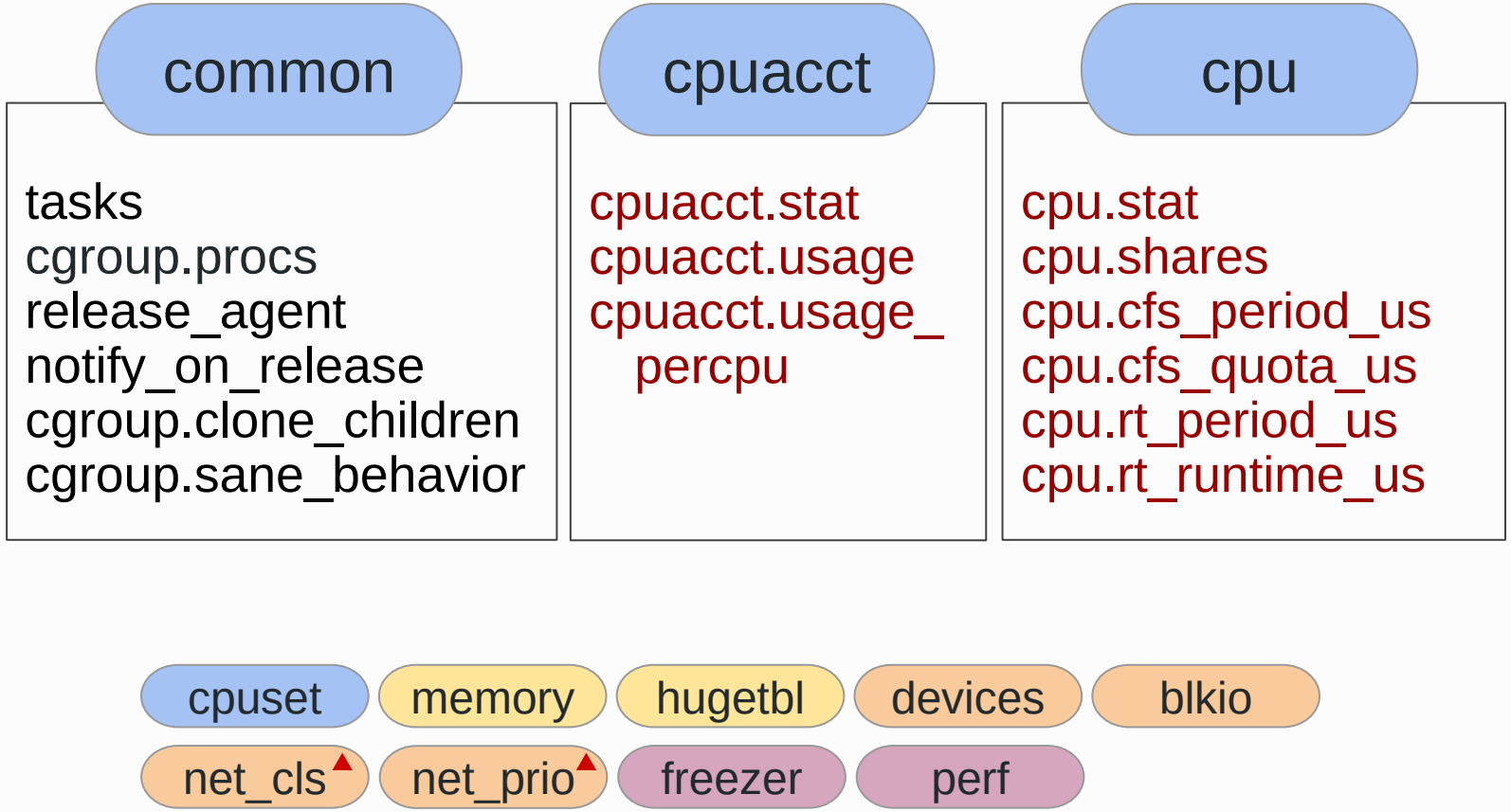
/sys/fs/cgroup/
…
↳blkio
↳cpu
↳memory
↳net_cls
↳systemd
…
↳user.slice
↳system.slice

.service
group.clone_children
group.procs
↳notify_on_release
↳tasks

CGROUP用户视角

cgroup hierarchies

```
/sys/fs/cgroup
  /cpu
    /high-priority
    /normal
    /experiment_1
  /mem
    /opus
    /normal
    /experiment_1
```



CAPABILITIES

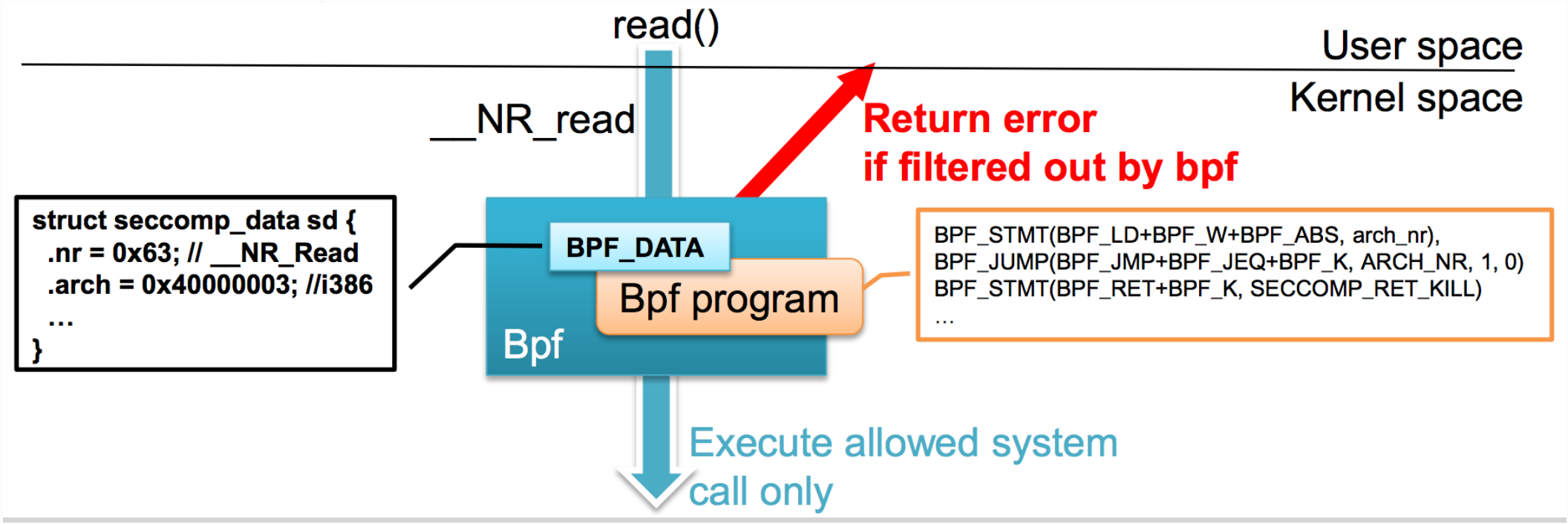
控制系统特权

- 超级用户的高级权限划分为Capabilities
- 独立使能单个Capability
- 禁用容器不需要的Capability

```
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# docker run -d --privileged ubuntu:14.04 sleep 1000
c2b9f7786a03e716e6be58b274fe1c669446b740df29ac53beaac6c22e821951
root@ubuntu:~#
root@ubuntu:~#
```

SECCOMP

系统调用过滤



MAC
(MANDATORY ACCESS CONTROL)

- SELinux/AppArmor
- 强制访问控制
- Mount Options
  - /dev/opts
  - root fs
- 访问系统路径
  - /sys
  - /proc

Available Container Security Features, Requirements and Defaults			
Security Feature	**LXC 2.0**	**Docker 1.11**	**CoreOS Rkt 1.3**
User Namespaces	Default	Optional	Experimental
Root Capability Dropping	Weak Defaults	Strong Defaults	Weak Defaults
Procfs and Sysfs Limits	Default	Default	Weak Defaults
Cgroup Defaults	Default	Default	Weak Defaults
Seccomp Filtering	Weak Defaults	Strong Defaults	Optional
Custom Seccomp Filters	Optional	Optional	Optional
Bridge Networking	Default	Default	Default
Hypervisor Isolation	Coming Soon	Coming Soon	Optional
MAC: AppArmor	Strong Defaults	Strong Defaults	Not Possible
MAC: SELinux	Optional	Optional	Optional
No New Privileges	Not Possible	Optional	Not Possible
Container Image Signing	Default	Strong Defaults	Default
Root Interation Optional	True	False	Mostly False

Secure By Default

"In this modern age, I believe that there is little excuse for not running a Linux application in some form of a Linux container, MAC or lightweight sandbox."

– Aaron Grattafiori, author of NCC Group's white paper

一场本来是可以避免的
不必要战争

HUMAN FACTOR

```
// It's a bad idea to bind to TCP without tlsverify.
if proto == "tcp" && (serverConfig.TLSConfig == nil || serverConfig.T
    logrus.Warn("[!] DON'T BIND ON ANY IP ADDRESS WITHOUT setting "+
        "-tlsverify IF YOU DON'T KNOW WHAT YOU'RE DOING [!]")
}
```

一无所有!


DEMO

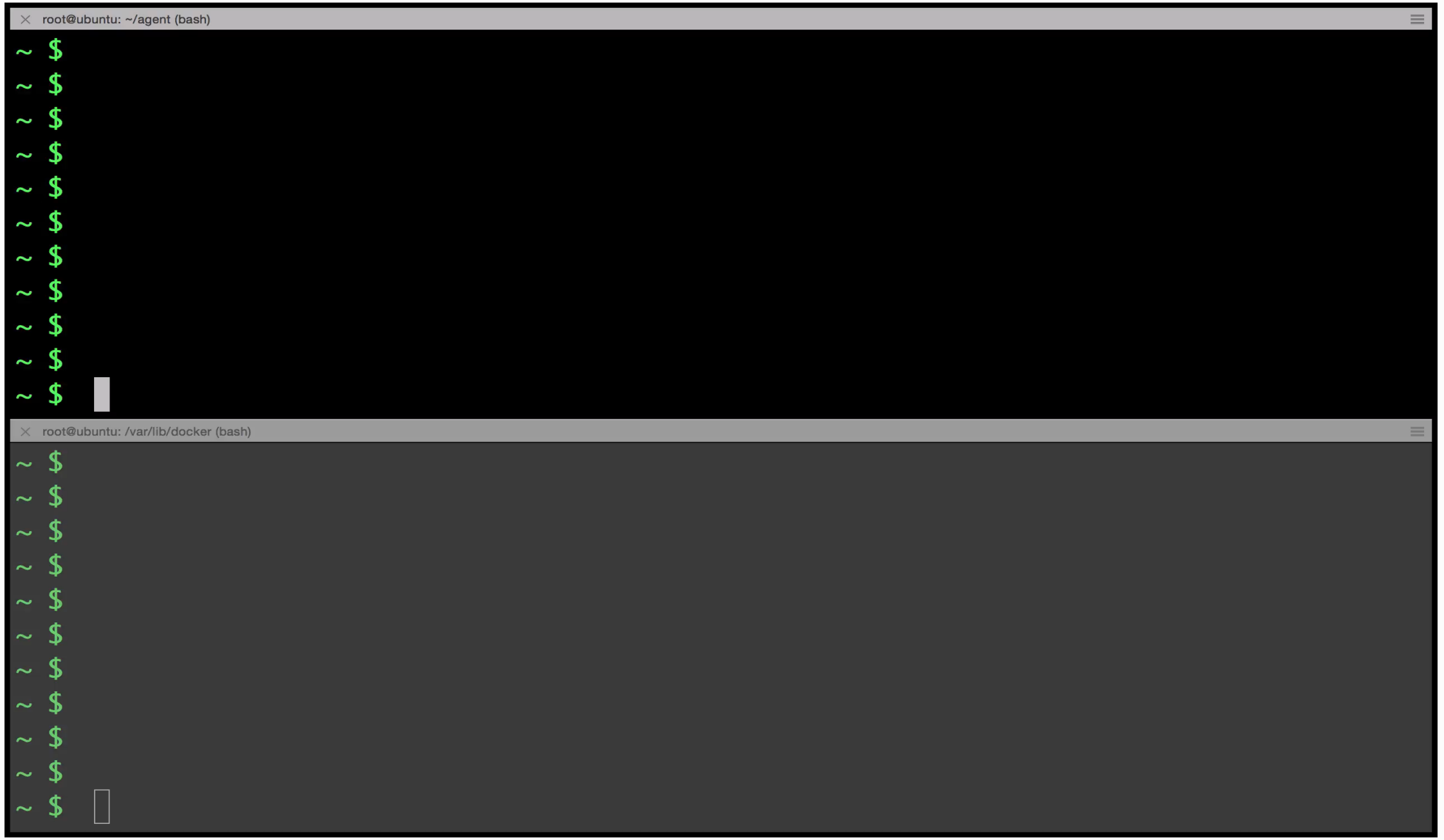
```
~ $
~ $
~ $  clear
~ $  ssh root@192.168.59.104
```

DOS

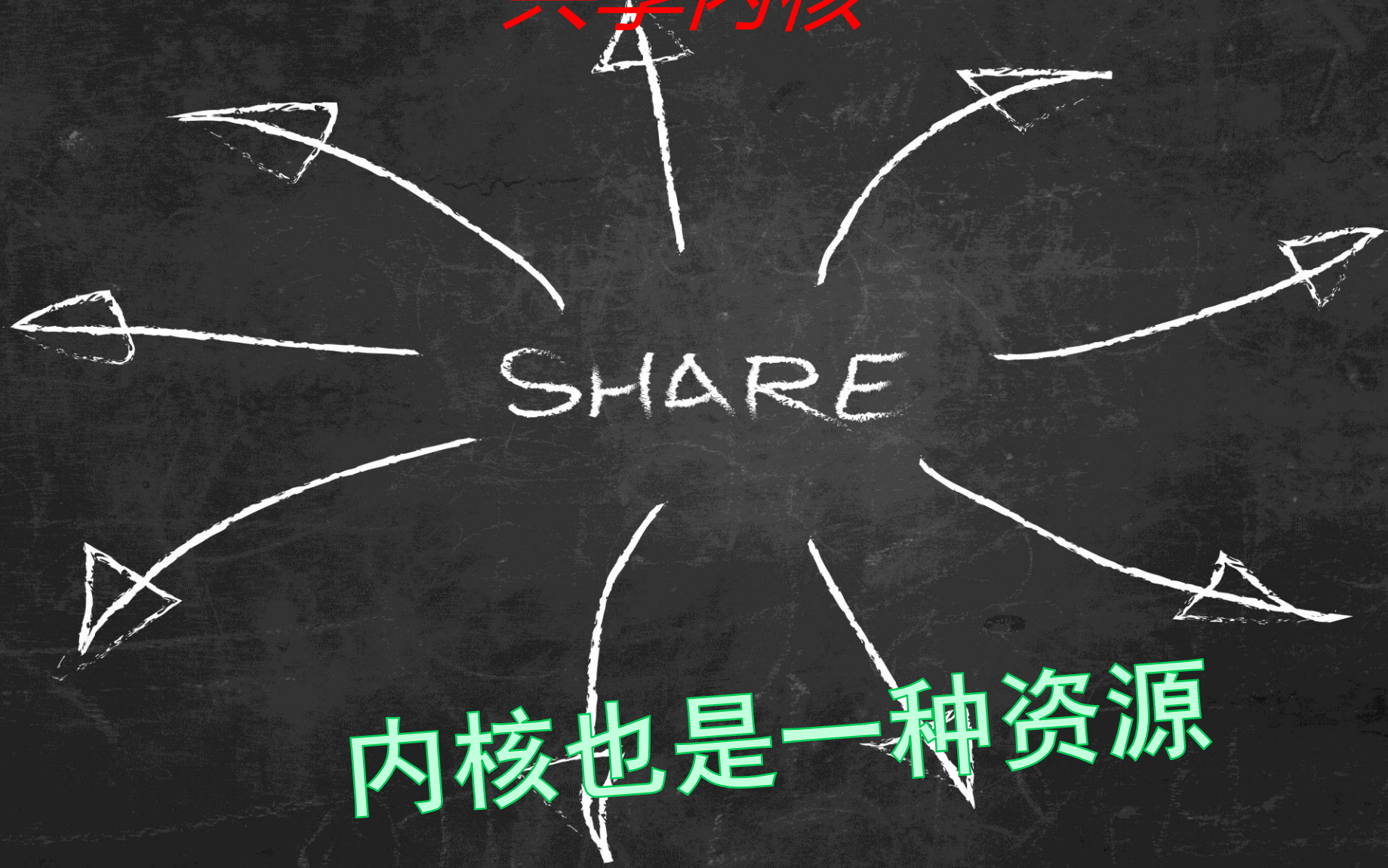
- Disk Space
- Processes
- Fd
- Signals

https://github.com/docker/docker/issues/15815 , docker 1.8.1


DEMO


root@ubuntu: ~/agent (bash)
root@ubuntu: /var/lib/docker (bash)

共享内核



内核也是一种资源


Chrome 文件 修改 视图 历史记录 书签 其他人 窗口 帮助
100% 周三上午1:53
DaoCloud Enterprise
Hongliang
demo.d.run/#/
总览
镜像仓库
admin
管理员
容器
9
容器个数
预计还可创建
20 个
预计容器总数
29 个
CPU 用量
7%
01:13
01:33
内存用量
30%
1.23 GB / 4.05 GB
01:13
01:33
系统状态
健康
您当前的系统运行正常
控制器
正常
Swarm
正常
etcd
正常
告警
目前没有任何告警
您可以悠闲享受下午茶了
应用
1 个运行中的应用
0 个停止中的应用
共 1 个应用
磁盘用量
24%
已使用 5.72 GB
剩余 13.84 GB
共 19.56 GB
主机状态
1 个健康主机
0 个异常主机
共 1 个主机
审计日志
voting 扩展成功
11 小时前
耗时 几秒
voting 部署成功
12 小时前
voting 卸载并删除成功
网络
8 个
存储卷
12 个
容器状态
共 9 个
全部
运行中
用量预测
444 天

REFERENCE

- http://blog.etsukata.com/2014/05/docker-linux-kernel.html
- https://blog.docker.com/2016/04/docker-security/

Thank you

DaoCloud