Container Security

Daniel J Walsh
Consulting Engineer
Twitter: @rhatdan
Blog: danwalsh.livejournal.com
Email: dwalsh@redhat.com

Container Security

Container Security

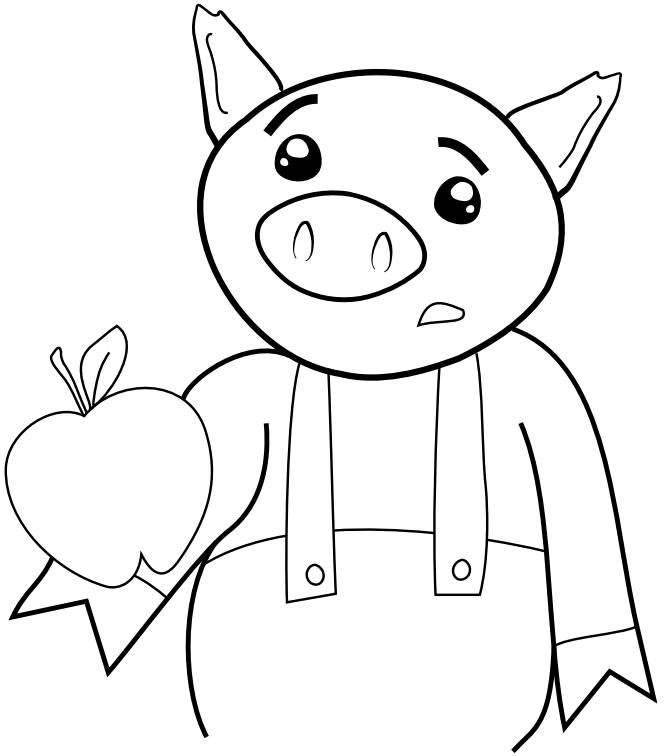
As explained by the three pigs

Chapter 1: When should I use containers versus VMs?

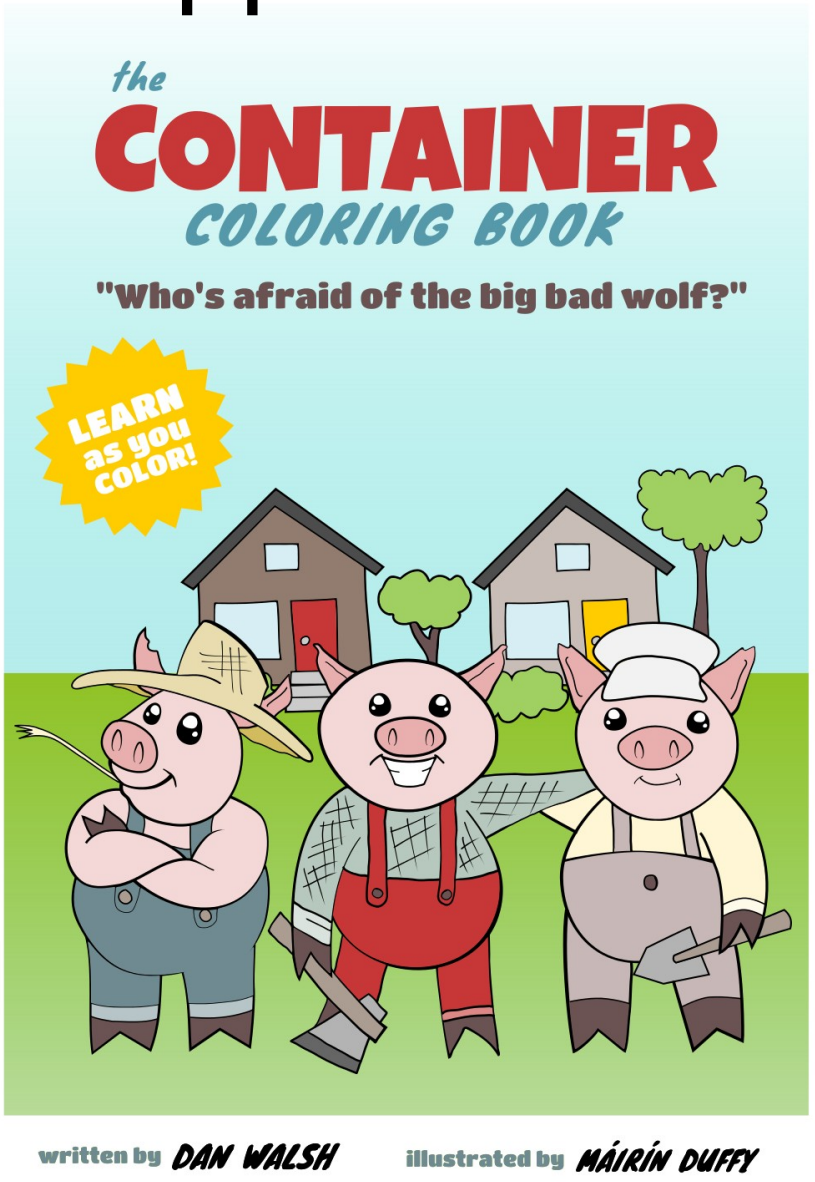
Chapter 2: What platform should host my containers?

Chapter 3: How do I ensure container separation?

Chapter 4: How do I secure content inside container?



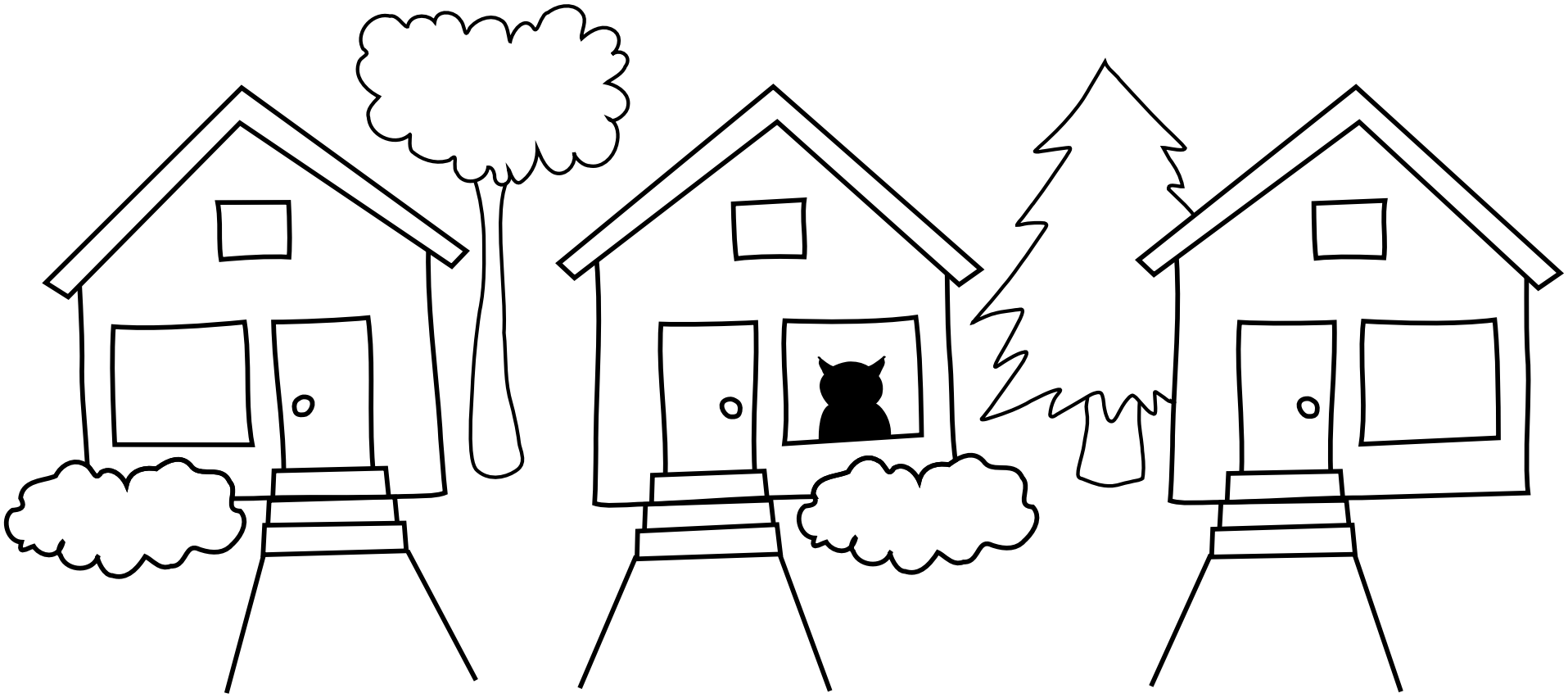
GLOSSARY
PIG==Application Service



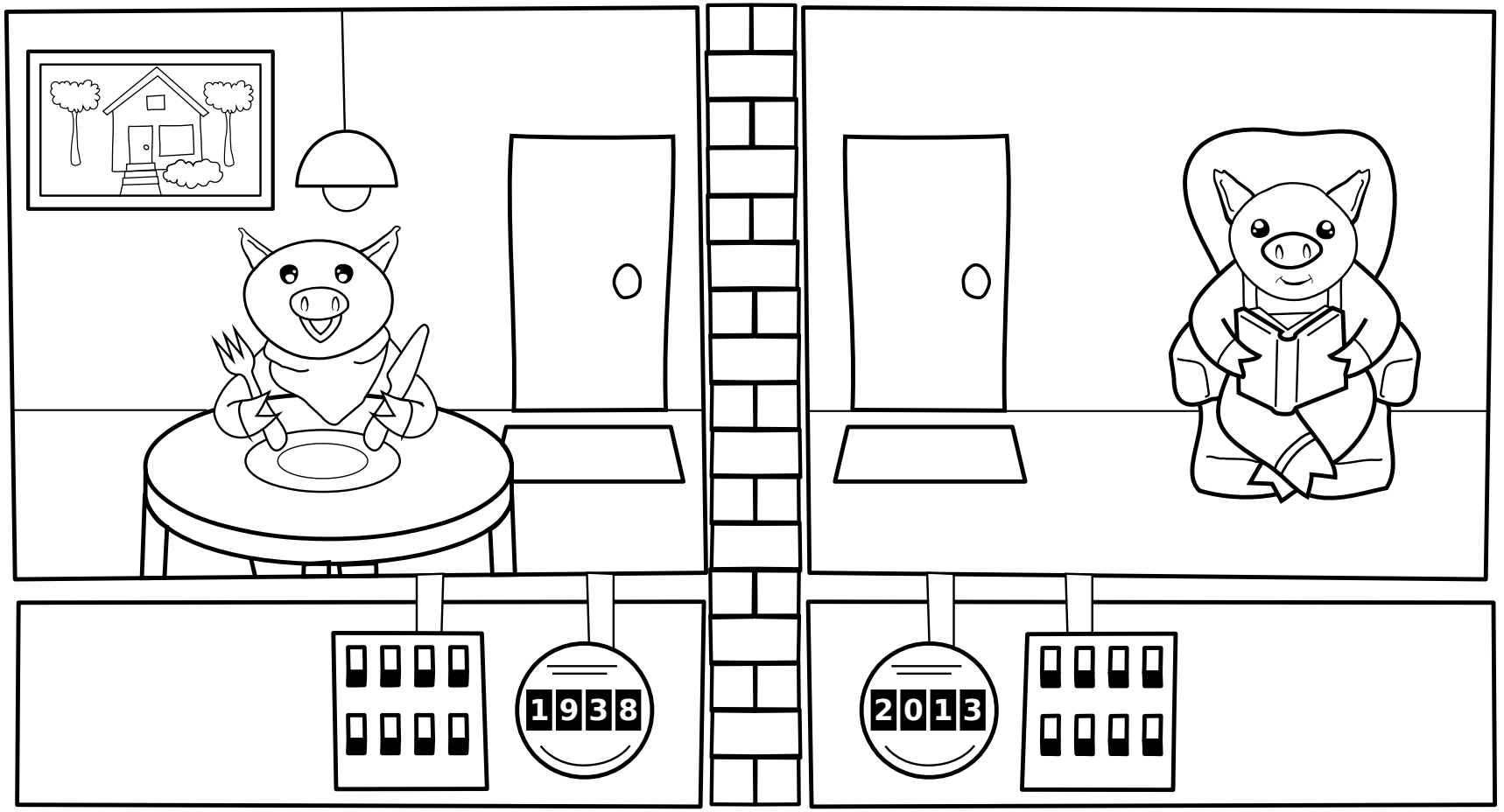
Chapter 1
Where should the pigs live?

When should I use containers versus virtual machines?

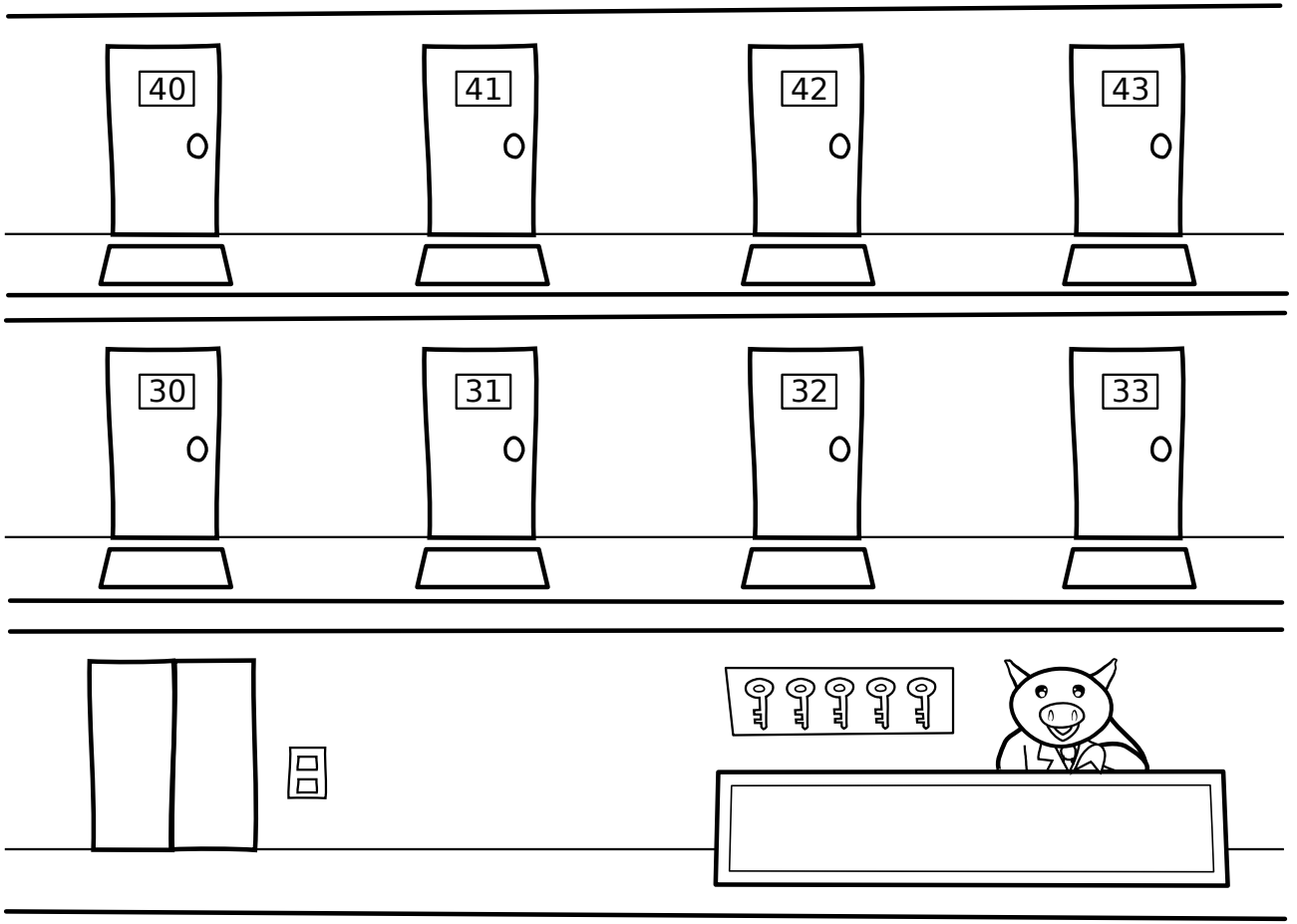
Standalone Homes
(Separate Physical Machines)



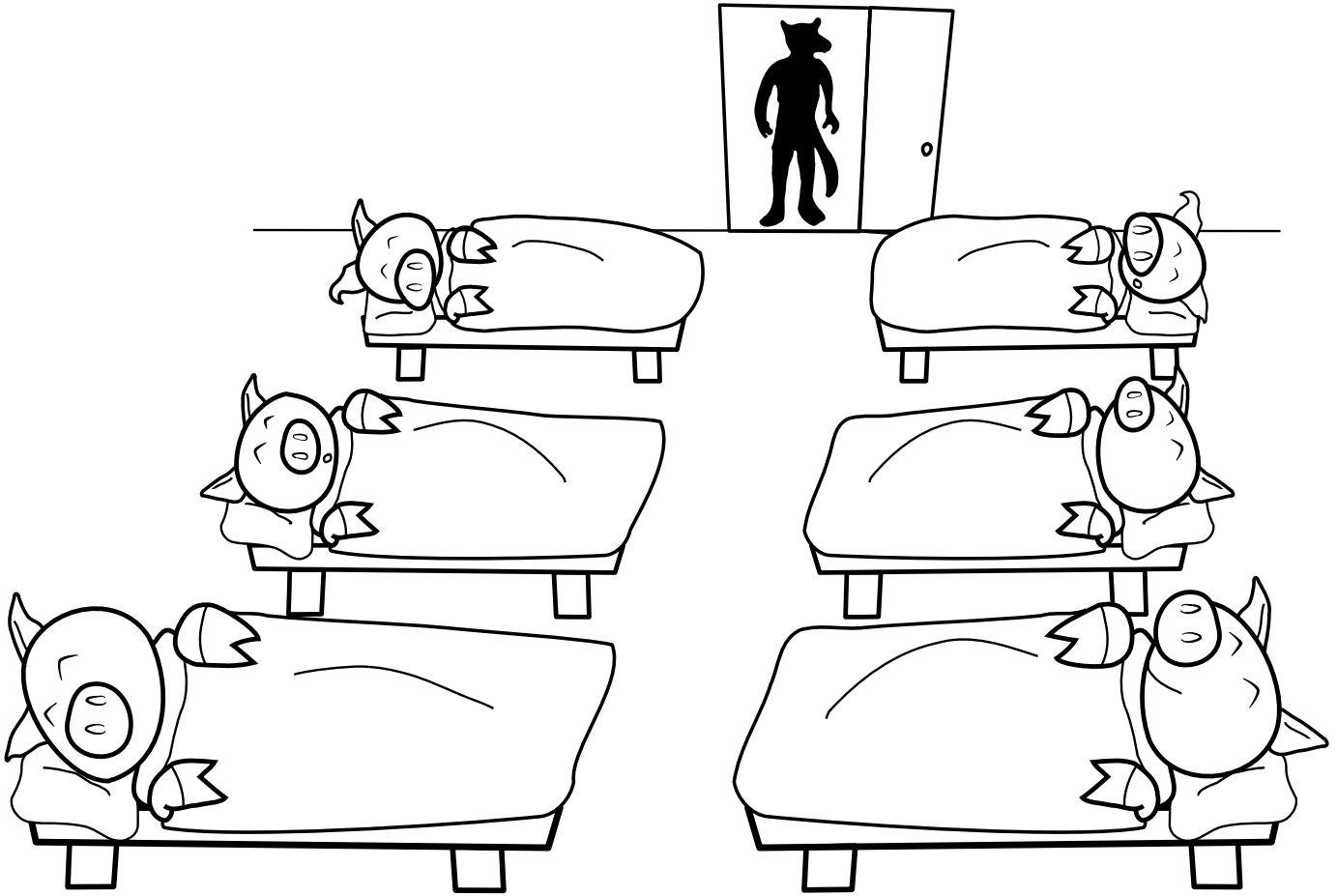
Duplex Home
(Virtual Machines)



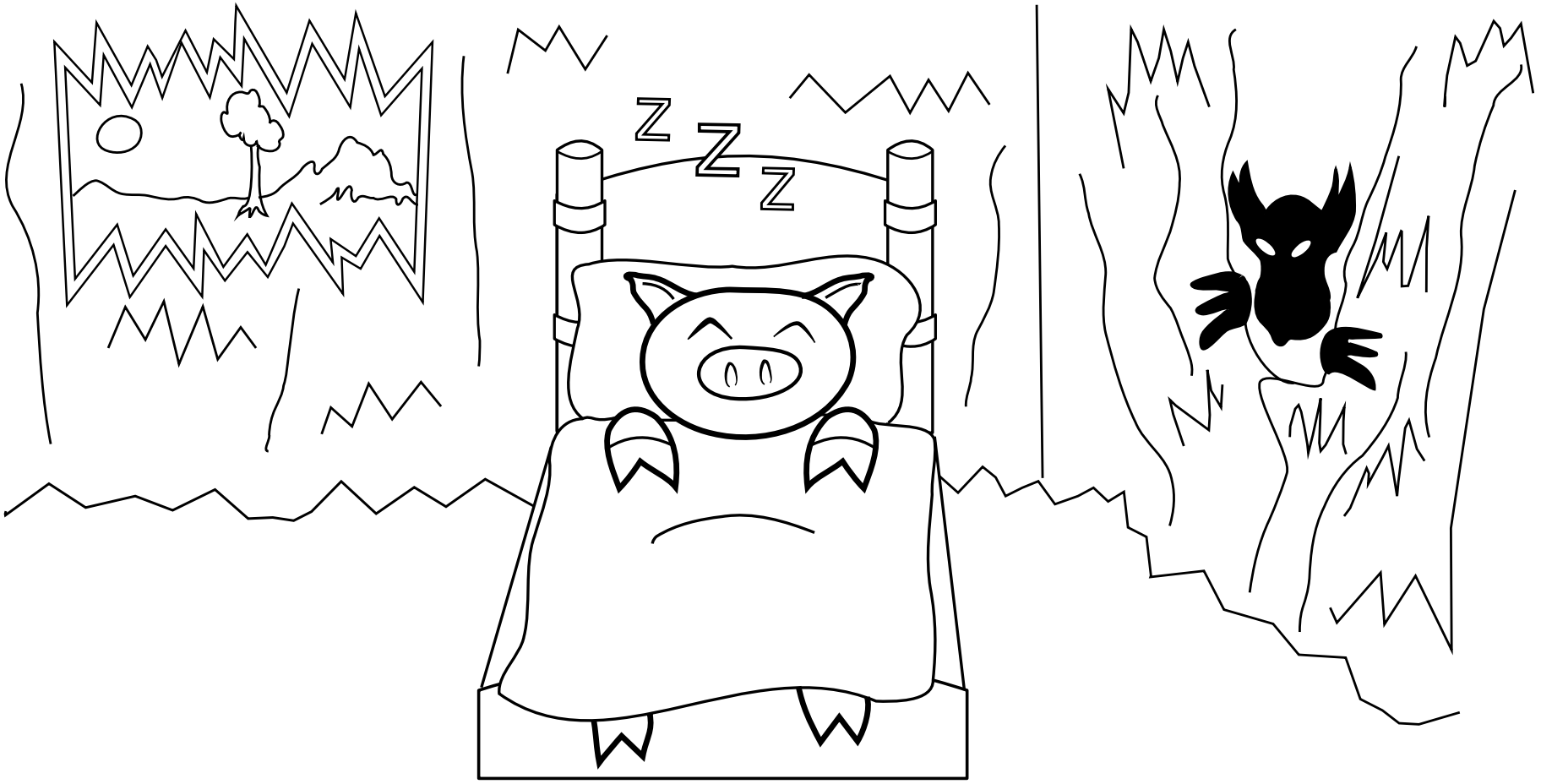
Apartment Building (Containers)



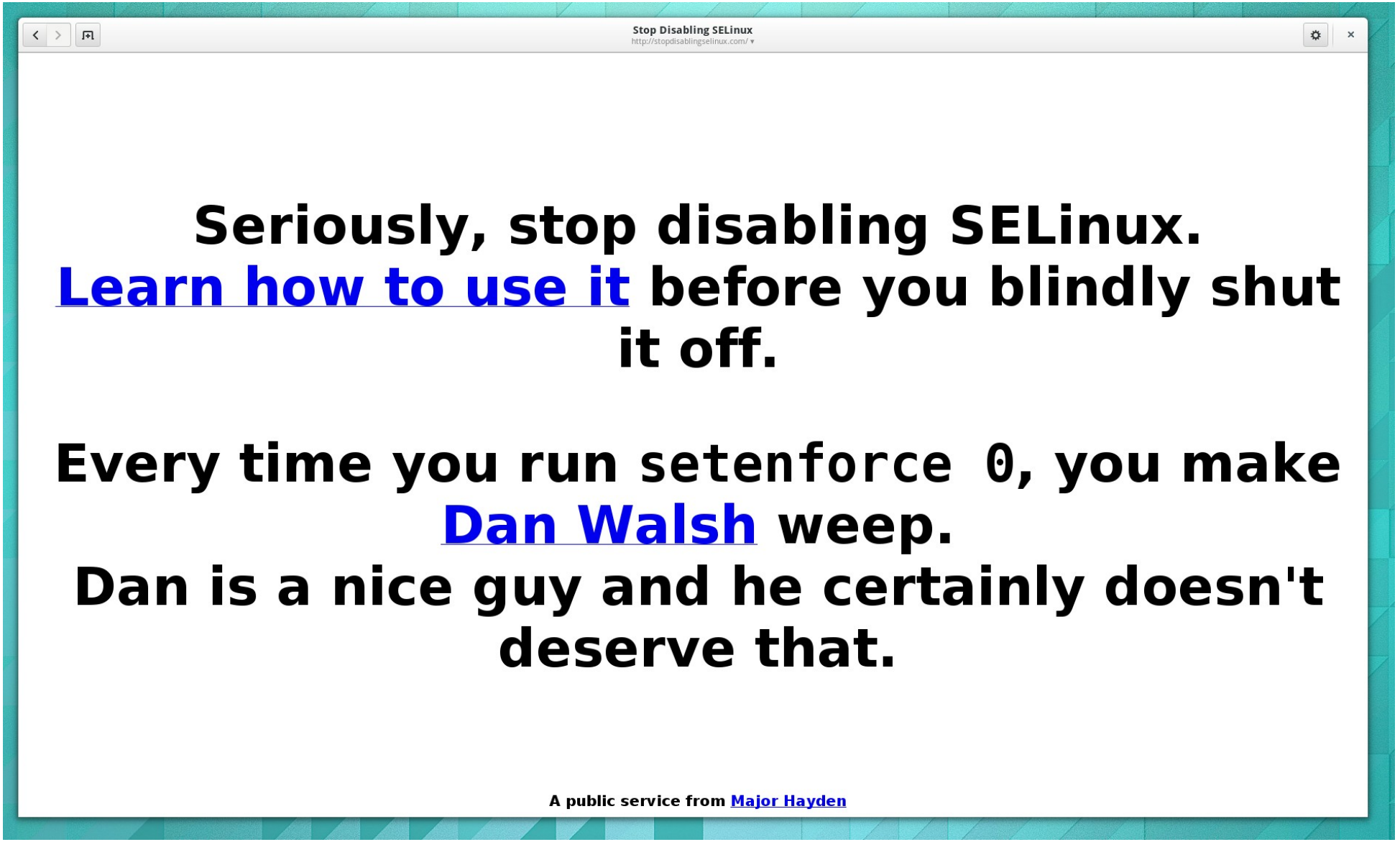
Hostel
(Services Same Machine)



Park
(setenforce 0)



http://stopdisablingselinux.com/



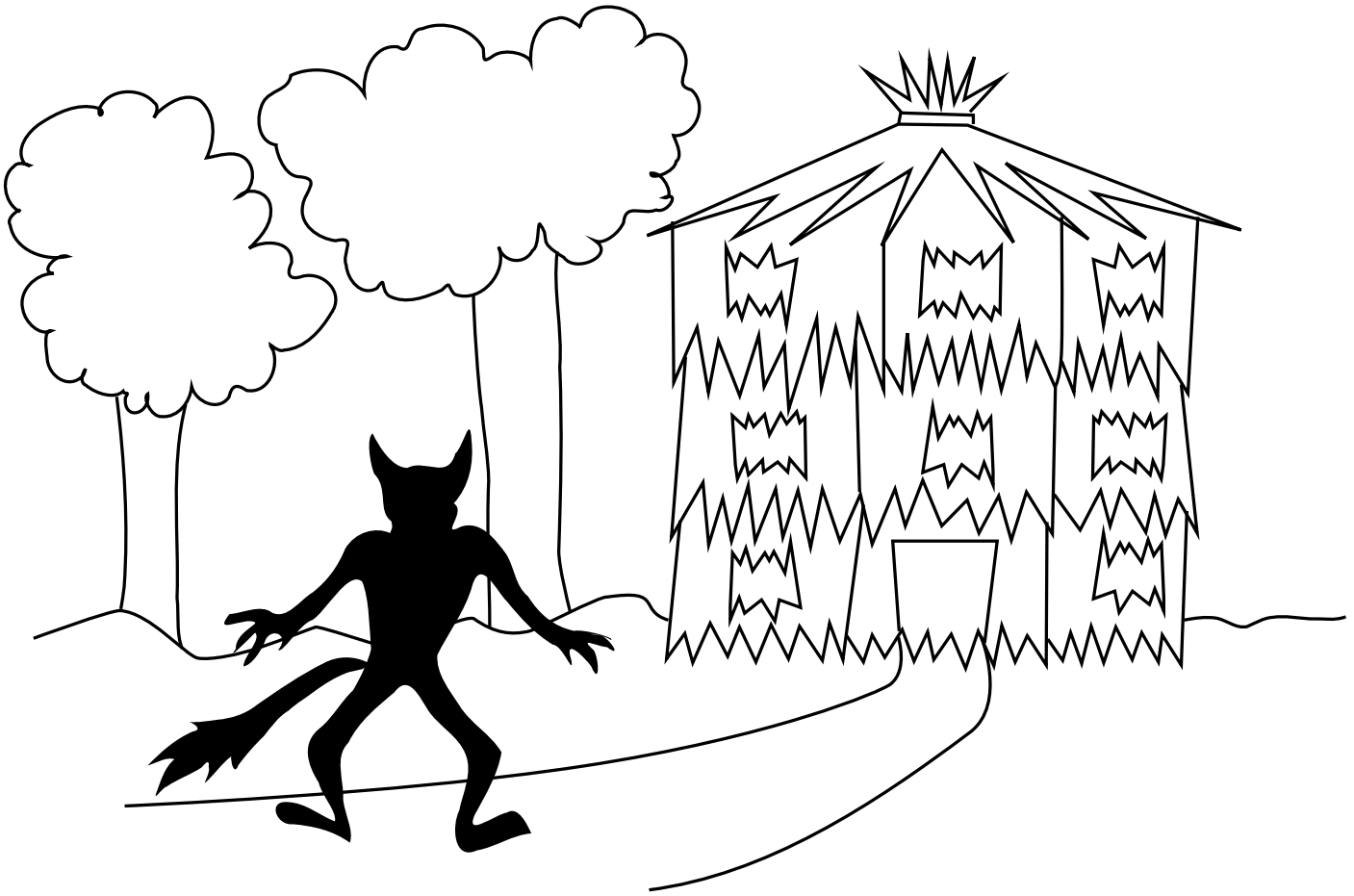
Pigs in Apartment Buildings

Best combination of resource sharing ease of maintainance & security

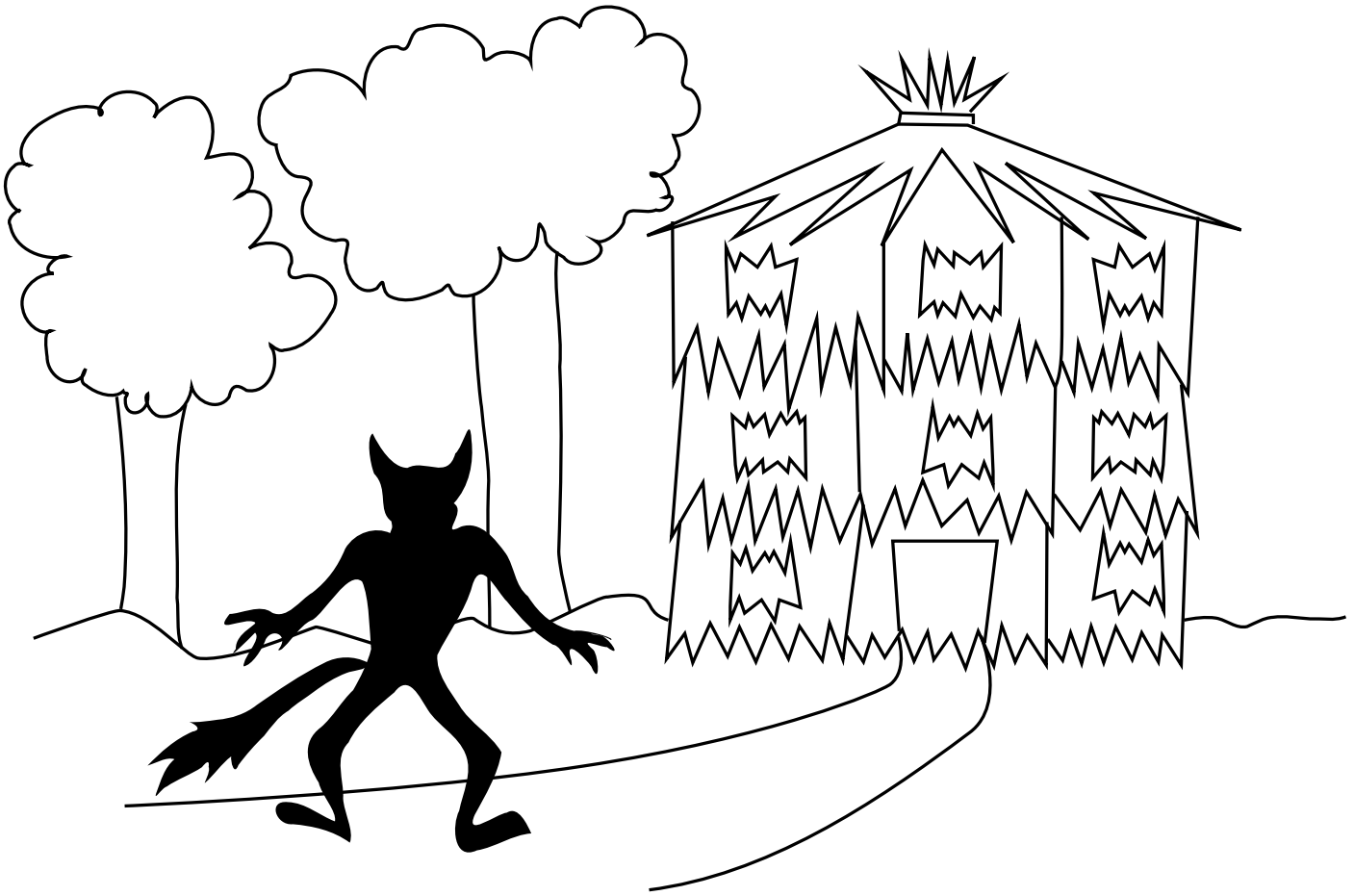
Chapter 2
What kind of apartment building?

What platform should host your containers?

Straw?

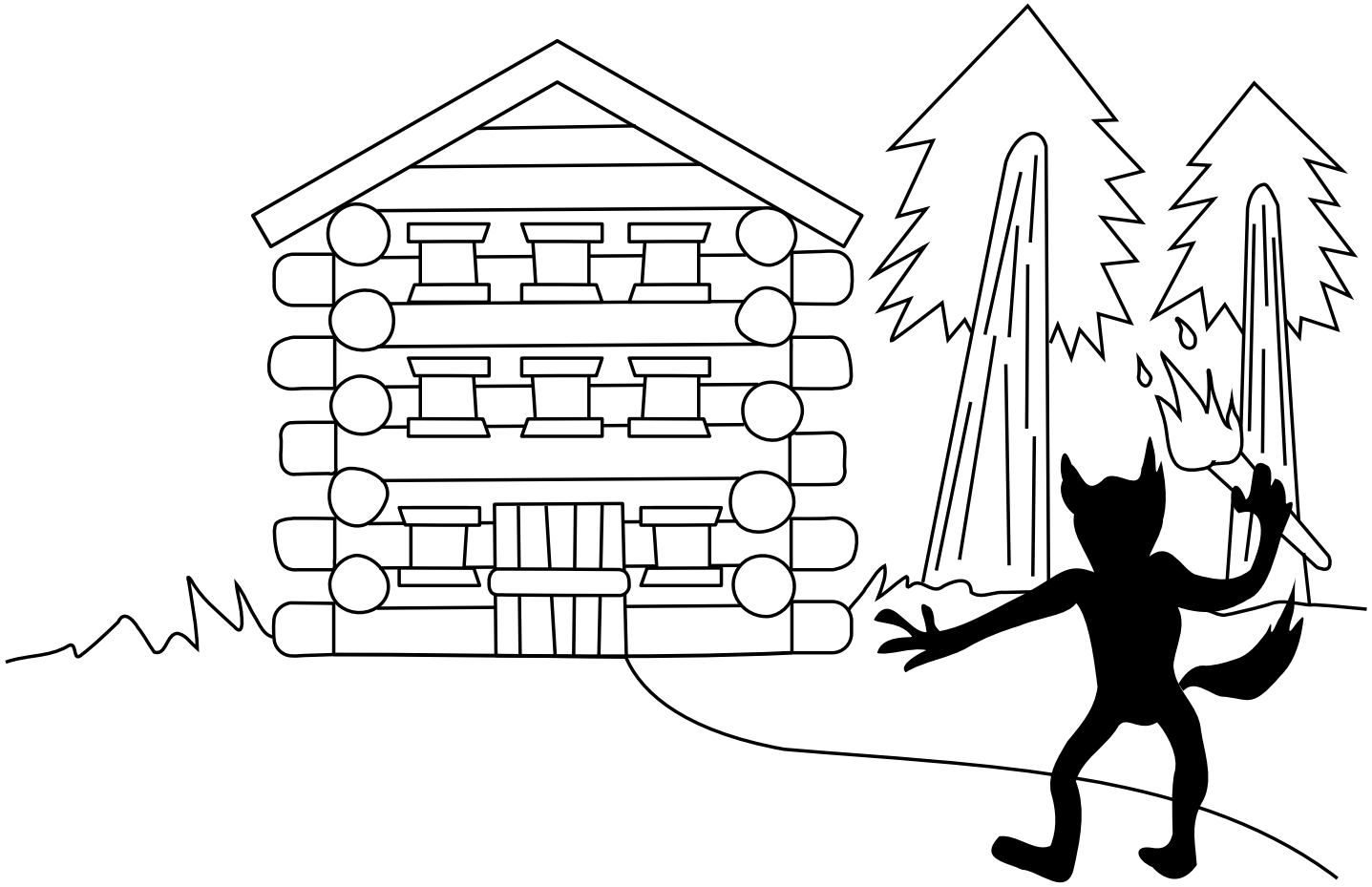


Straw?

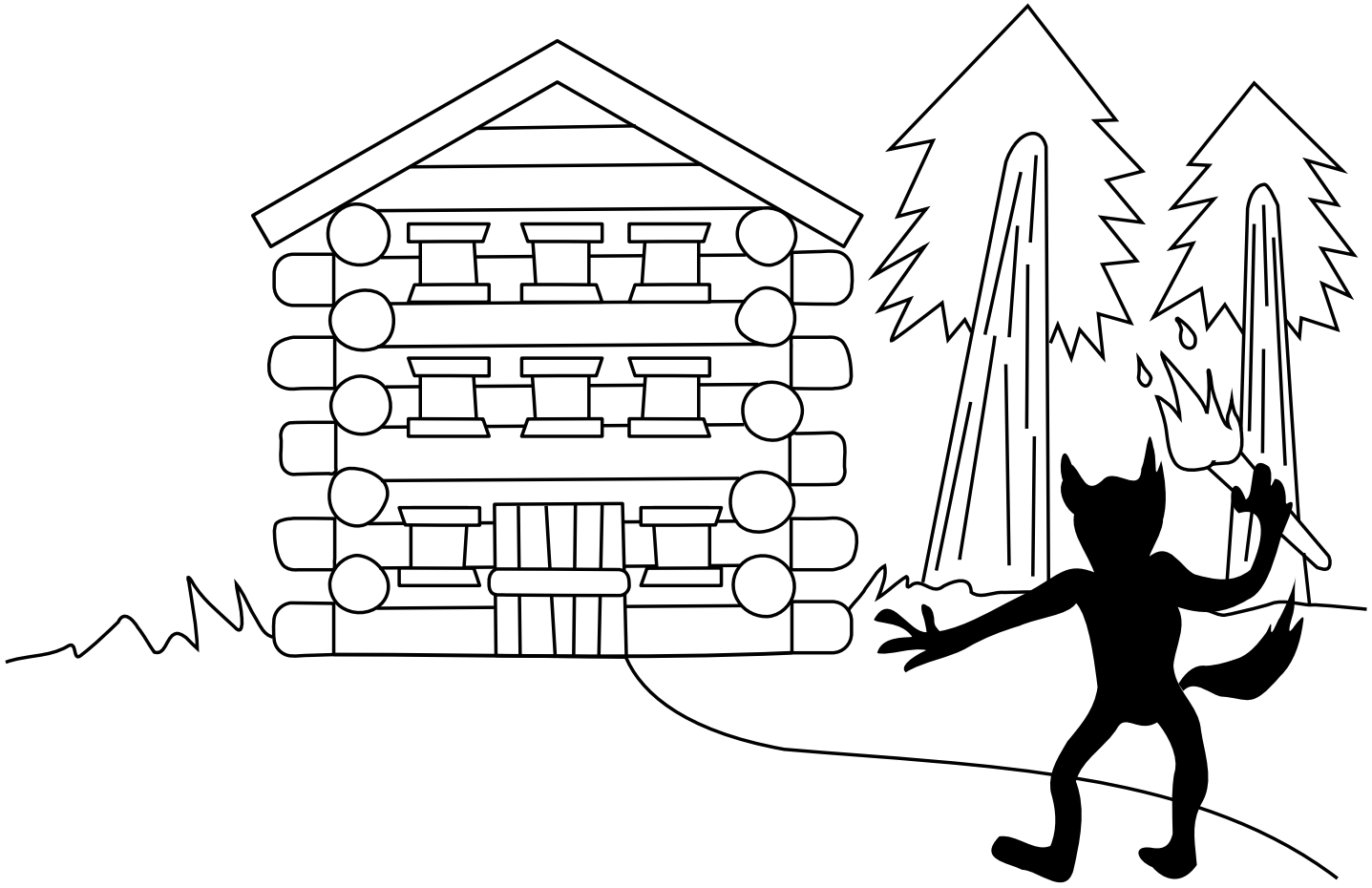


Running containers on do it yourself platform.

Sticks?



Sticks?



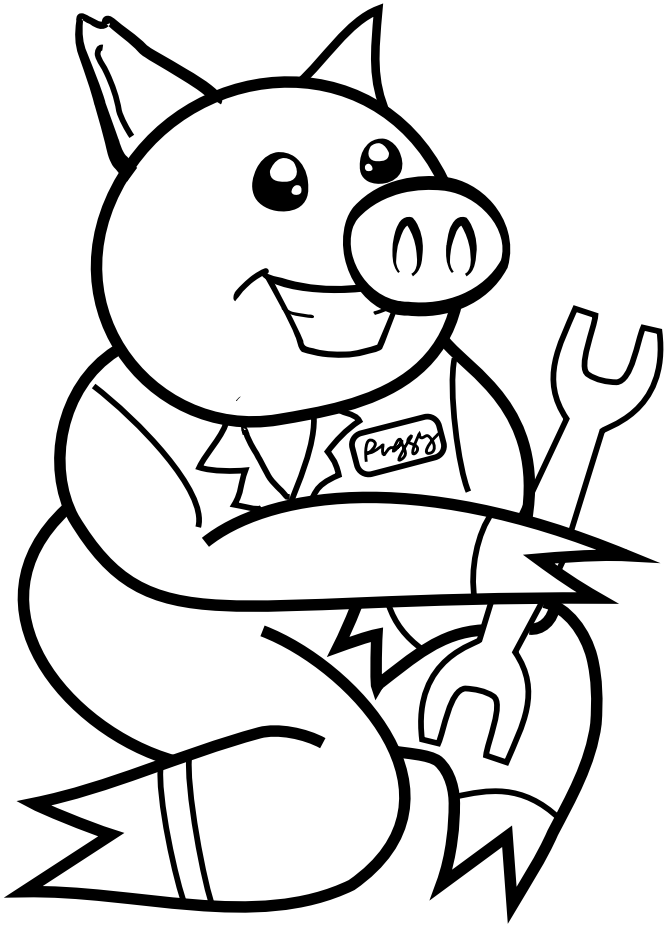
Running containers on community platform.

Brick?

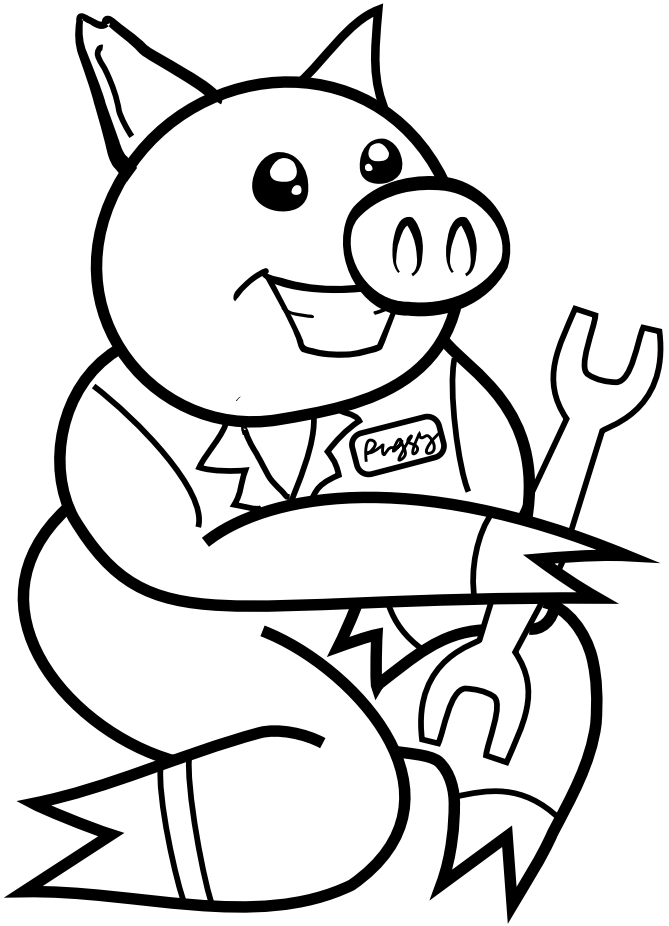
Brick?

Running containers on
Red Hat Enterprise Linux

RHEL Maintenance



RHEL Maintenance



Security Response Team.

Chapter 3
How do I separate/secure pig apartments?

Chapter 3
How do I separate/secure pig apartments?

How do you ensure container separation?

CONTAINERS DO NOT CONTAIN

http://www.maritimenz.govt.nz/images/Incident-area/Rena7.jpg

Do you care?

Should you care?

Treat Container Services just like regular services

- Drop privileges as quickly as possible

Treat Container Services just like regular services

- Drop privileges as quickly as possible
- Run your services as non Root whenever possible

Treat Container Services just like regular services

- Drop privileges as quickly as possible
- Run your services as non Root whenever possible
- Treat root within a container the same as root outside of the container

Treat Container Services just like regular services

- Drop privileges as quickly as possible
- Run your services as non Root whenever possible
- Treat root within a container the same as root outside of the container

"Docker is about running random crap from the internet as root on your host"

"Docker is about running random crap from the internet as root on your host"

Only run container images from trusted parties

See Chapter 4

Why don't containers contain?

Why don't containers contain?

- Everything in Linux is not namespaced

Why don't containers contain?

- Everything in Linux is not namespaced
- Containers are not comprehensive like virtual machines (kvm)

Why don't containers contain?

- Everything in Linux is not namespaced
- Containers are not comprehensive like virtual machines (kvm)
- Kernel file systems: /sys, /sys/fs, /proc/sys
- Cgroups, SELinux, /dev/mem, kernel modules

Chapter 3
Overview of Security within Docker containers

Read Only Mount Points

- /sys, /proc/sys, /proc/sysrq-trigger,/proc/irq, /proc/bus

Capabilities

man capabilities

DESCRIPTION

For the purpose of performing permission checks, traditional UNIX implementations distinguish two categories of processes: privileged processes (whose effective user ID is 0, referred to as superuser or root), and unprivileged processes (whose effective UID is nonzero). **Privileged processes bypass all kernel permission checks**, while unprivileged processes are subject to full permission checking based on the process's credentials (usually: effective UID, effective GID, and supplementary group list).

Starting with kernel 2.2, **Linux divides the privileges traditionally associated with superuser into distinct units, known as capabilities, which can be independently enabled and disabled**. Capabilities are a per-thread attribute.

Capabilities Removed

CAP_SETPCAP	Modify process capabilities
CAP_SYS_MODULE	Insert/Remove kernel modules
CAP_SYS_RAWIO	Modify Kernel Memory
CAP_SYS_PACCT	Configure process accounting
CAP_SYS_NICE	Modify Priotity of processes
CAP_SYS_RESOURCE	Override Resource Limits
CAP_SYS_TIME	Modify the system clock
CAP_SYS_TTY_CONFIG	Configure tty devices
CAP_AUDIT_WRITE	Write the audit log
CAP_AUDIT_CONTROL	Configure Audit Subsystem
CAP_MAC_OVERRIDE	Ignore Kernel MAC Policy
CAP_MAC_ADMIN	Configure MAC Configuration
CAP_SYSLOGModify	Kernel printk behavior

Capabilities Removed

CAP_NET_ADMIN Configure the network

Capabilities Removed

CAP_NET_ADMIN Configure the network

CAP_SYS_ADMIN Catch all

SYS_ADMIN

```
less /usr/include/linux/capability.h
...
/* Allow configuration of the secure attention key */
/* Allow administration of the random device */
/* Allow examination and configuration of disk quotas */
/* Allow setting the domainname */
/* Allow setting the hostname */
/* Allow calling bdflush() */
/* Allow mount() and umount(), setting up new smb connection */
/* Allow some autofs root ioctls */
/* Allow nfsservctl */
/* Allow VM86_REQUEST_IRQ */
/* Allow to read/write pci config on alpha */
/* Allow irix_prctl on mips (setstacksize) */
/* Allow flushing all cache on m68k (sys_cacheflush) */
/* Allow removing semaphores */
/* Used instead of CAP_CHOWN to "chown" IPC message queues, semaphores
   and shared memory */
/* Allow locking/unlocking of shared memory segment */
/* Allow turning swap on/off */
/* Allow forged pids on socket credentials passing */
/* Allow setting readahead and flushing buffers on block devices */
```

SYS_ADMIN

```
/* Allow setting geometry in floppy driver */
/* Allow turning DMA on/off in xd driver */
/* Allow administration of md devices (mostly the above, but some
   extra ioctls) */
/* Allow tuning the ide driver */
/* Allow access to the nvram device */
/* Allow administration of apm_bios, serial and bttv (TV) device */
/* Allow manufacturer commands in isdn CAPI support driver */
/* Allow reading non-standardized portions of pci configuration space */
/* Allow DDI debug ioctl on sbpcd driver */
/* Allow setting up serial ports */
/* Allow sending raw qic-117 commands */
/* Allow enabling/disabling tagged queuing on SCSI controllers and sending
   arbitrary SCSI commands */
/* Allow setting encryption key on loopback filesystem */
/* Allow setting zone reclaim policy */
```

Namespaces

- PID Namespace

Namespaces

- PID Namespace
- Network Namespace

Cgroups

Device Cgroup

Device nodes allow processes to configure kernel

Cgroups

Device Cgroup

Device nodes allow processes to configure kernel

Should have been a namespace

Cgroups

Device Cgroup

Device nodes allow processes to configure kernel

Should have been a namespace

Controls device nodes that can be created

Cgroups

Device Cgroup

Device nodes allow processes to configure kernel

Should have been a namespace

Controls device nodes that can be created

/dev/console/dev/zero /dev/null /dev/fuse

/dev/full /dev/tty* /dev/urandom /dev/random

Cgroups

Device Cgroup

Device nodes allow processes to configure kernel

Should have been a namespace

Controls device nodes that can be created

/dev/console/dev/zero /dev/null /dev/fuse

/dev/full /dev/tty* /dev/urandom /dev/random

Images also mounted with nodev

SELinux

Everyone Please standup and repeat after me.

SELinux

Everyone Please standup and repeat after me.

SELinux is a LABELING system

SELinux

Everyone Please standup and repeat after me.

SELinux is a LABELING system

Every Process has a LABEL

SELinux

Everyone Please standup and repeat after me.

SELinux is a LABELING system

Every Process has a LABEL

Every File, Directory, System object has a LABEL

SELinux

Everyone Please standup and repeat after me.

SELinux is a LABELING system

Every Process has a LABEL

Every File, Directory, System object has a LABEL

Policy rules control access between labeled processes and labeled objects

SELinux

Everyone Please standup and repeat after me.

SELinux is a LABELING system

Every Process has a LABEL

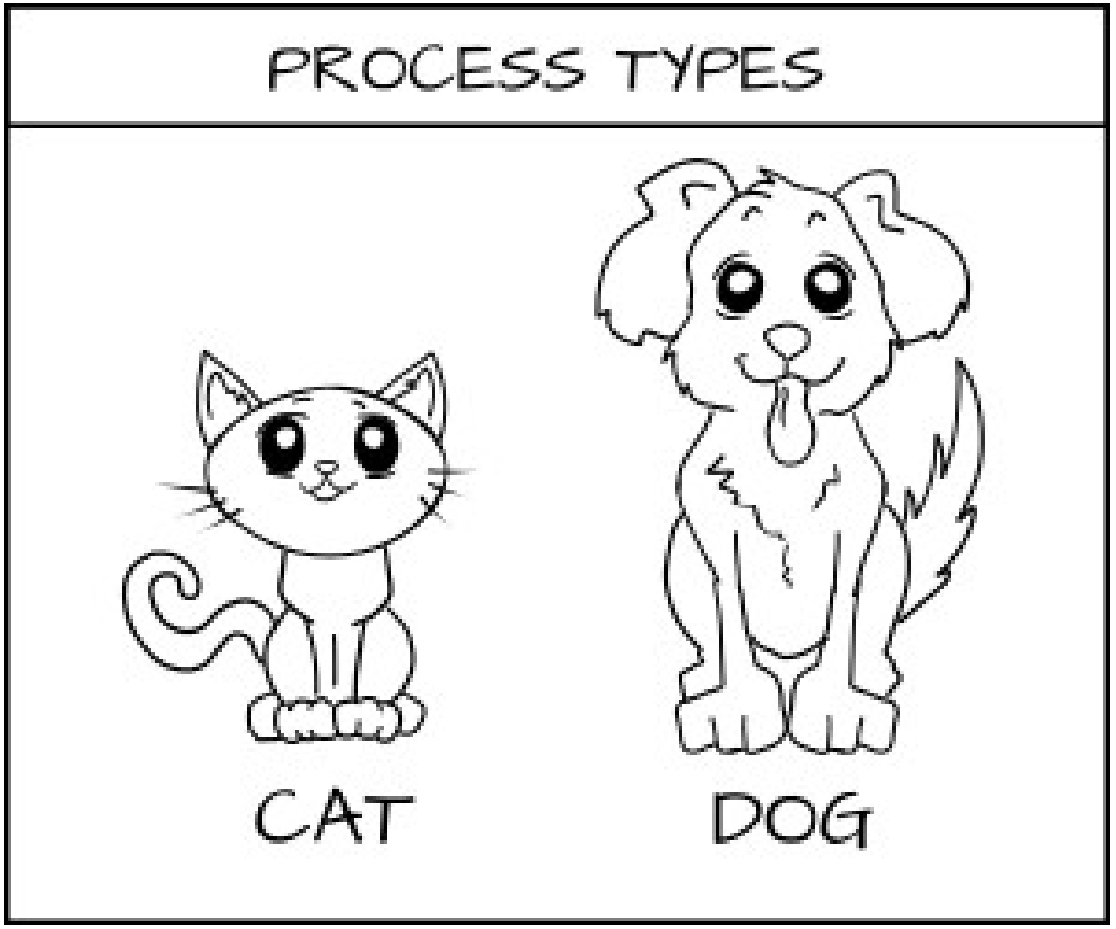
Every File, Directory, System object has a LABEL

Policy rules control access between labeled processes and labeled objects

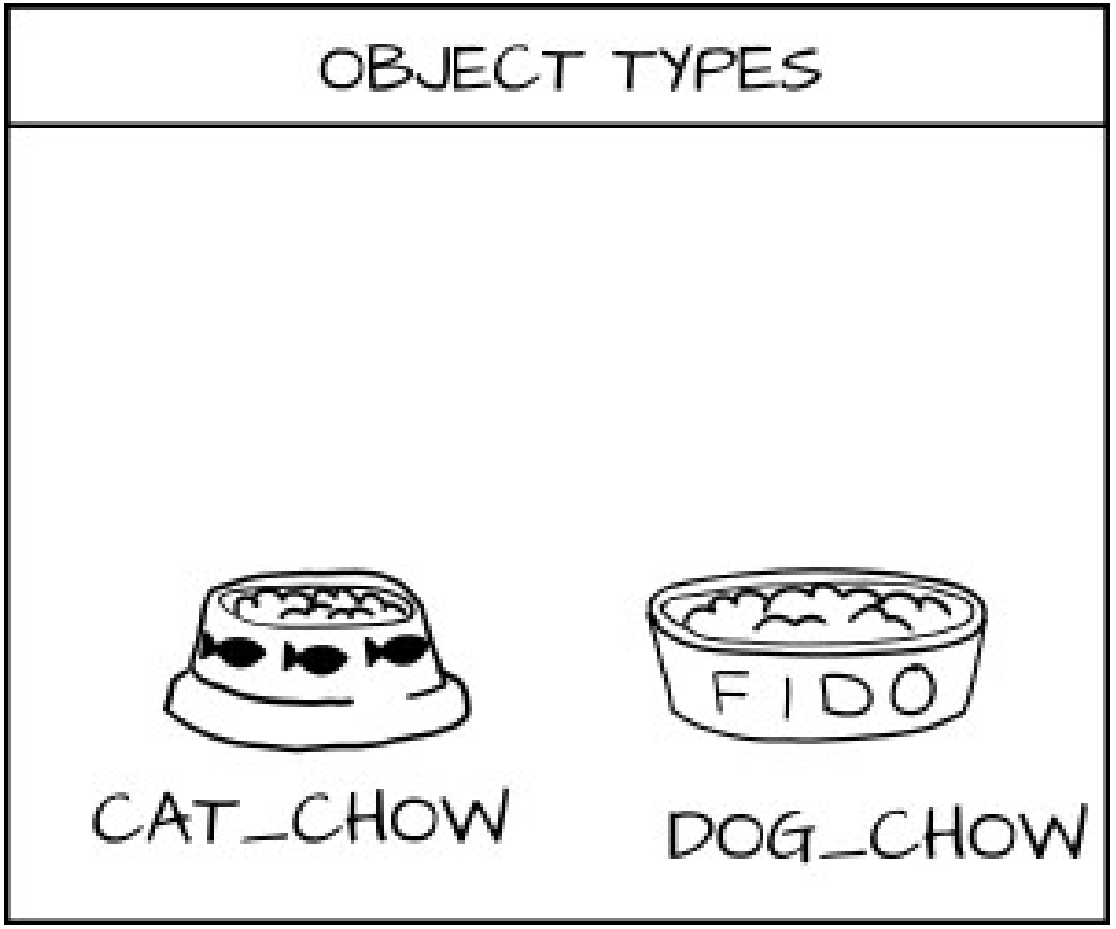
The Kernel enforces the rules

Grab your SELinux Coloring Book

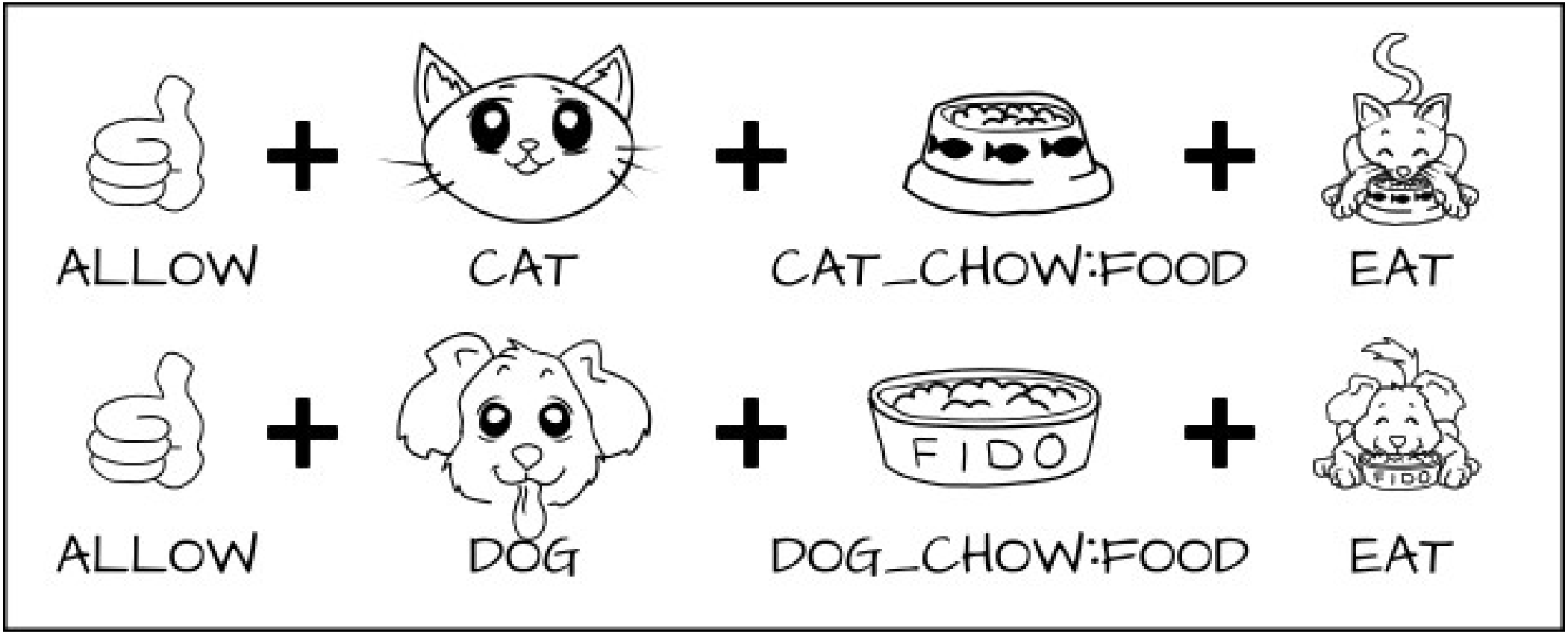
Type Enforcement



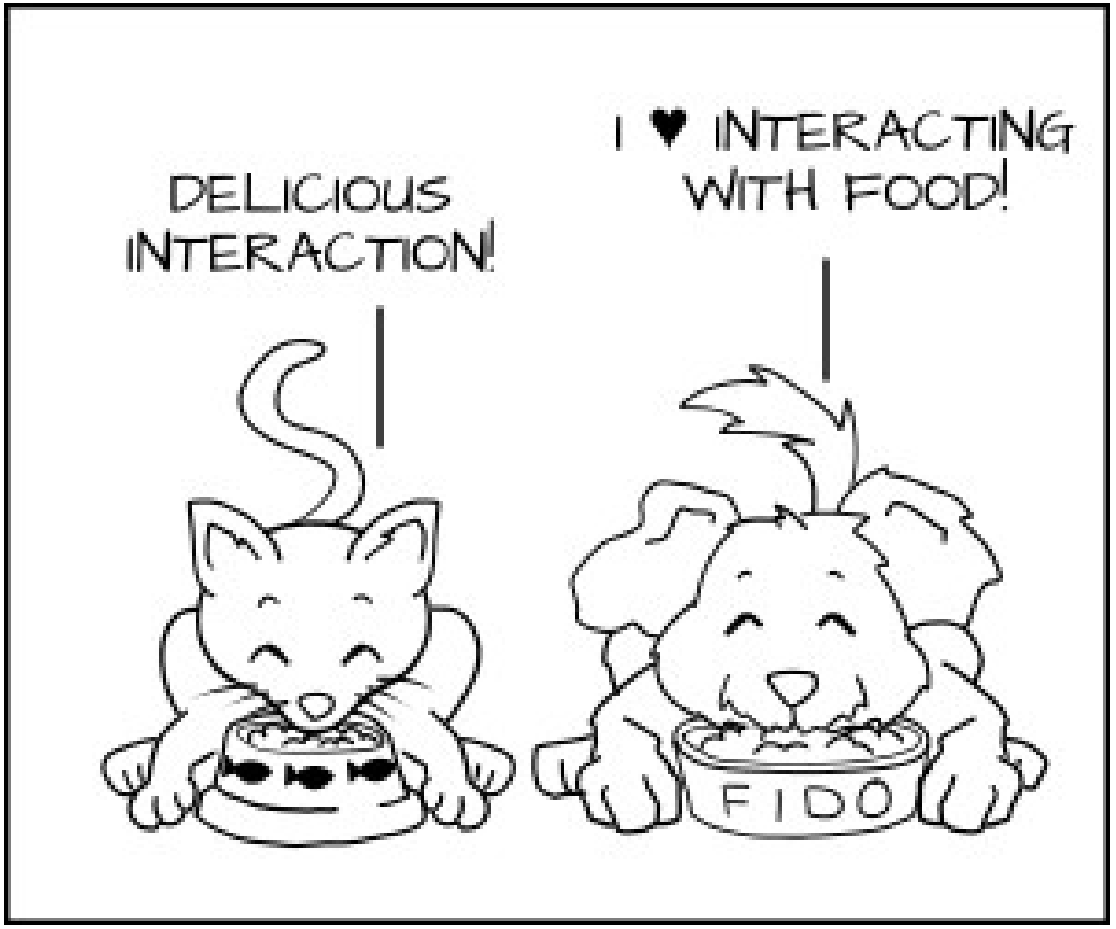
Type Enforcement



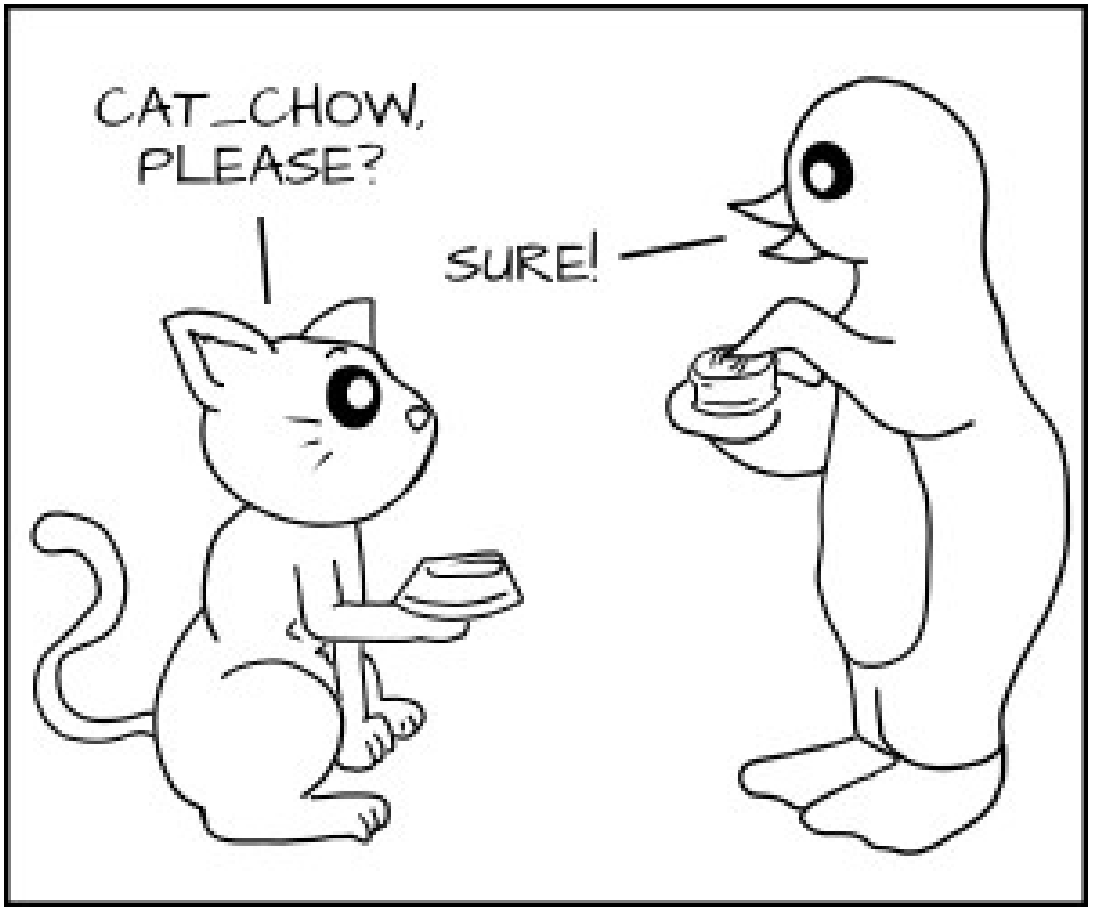
Type Enforcement



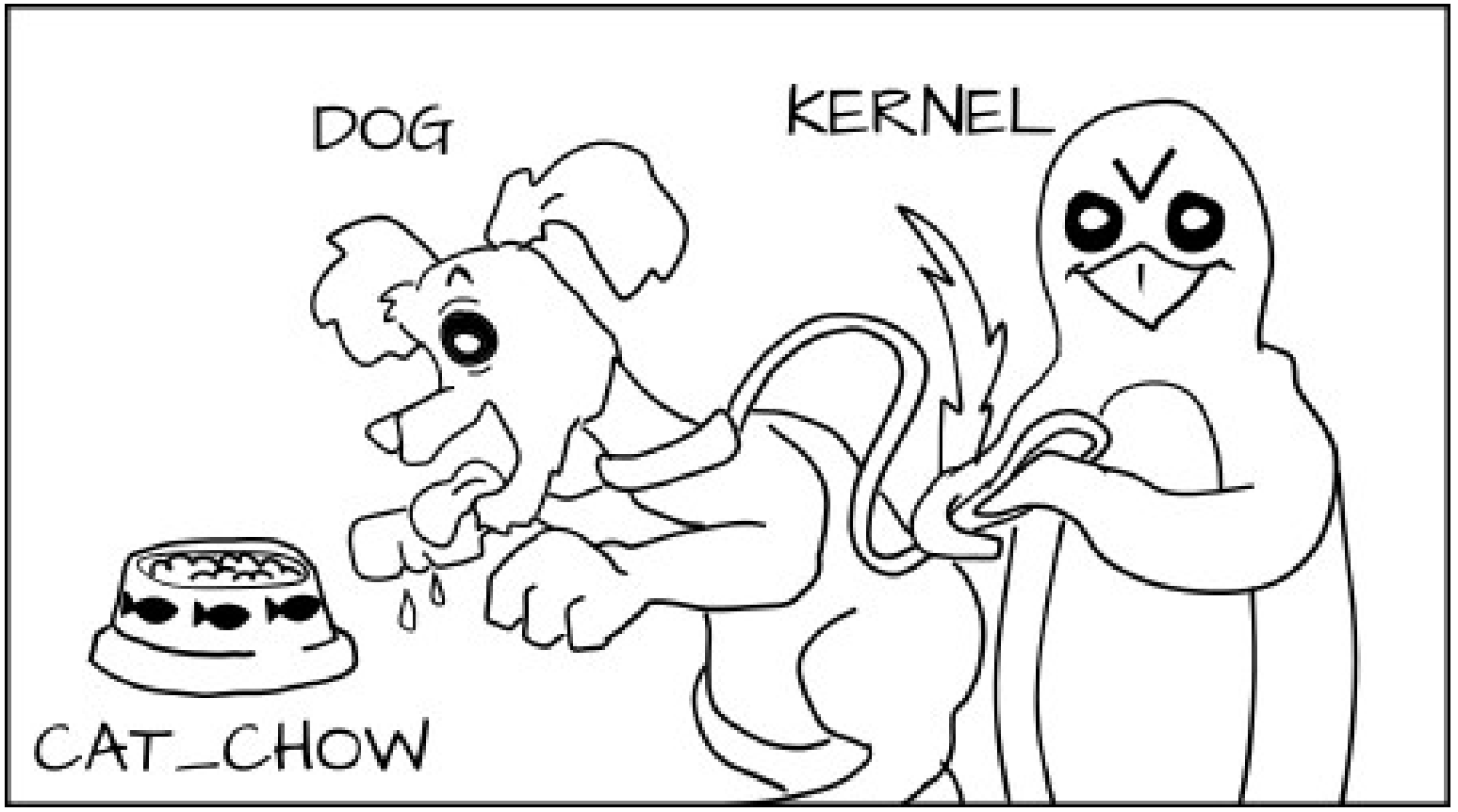
Type Enforcement



Type Enforcement



Type Enforcement



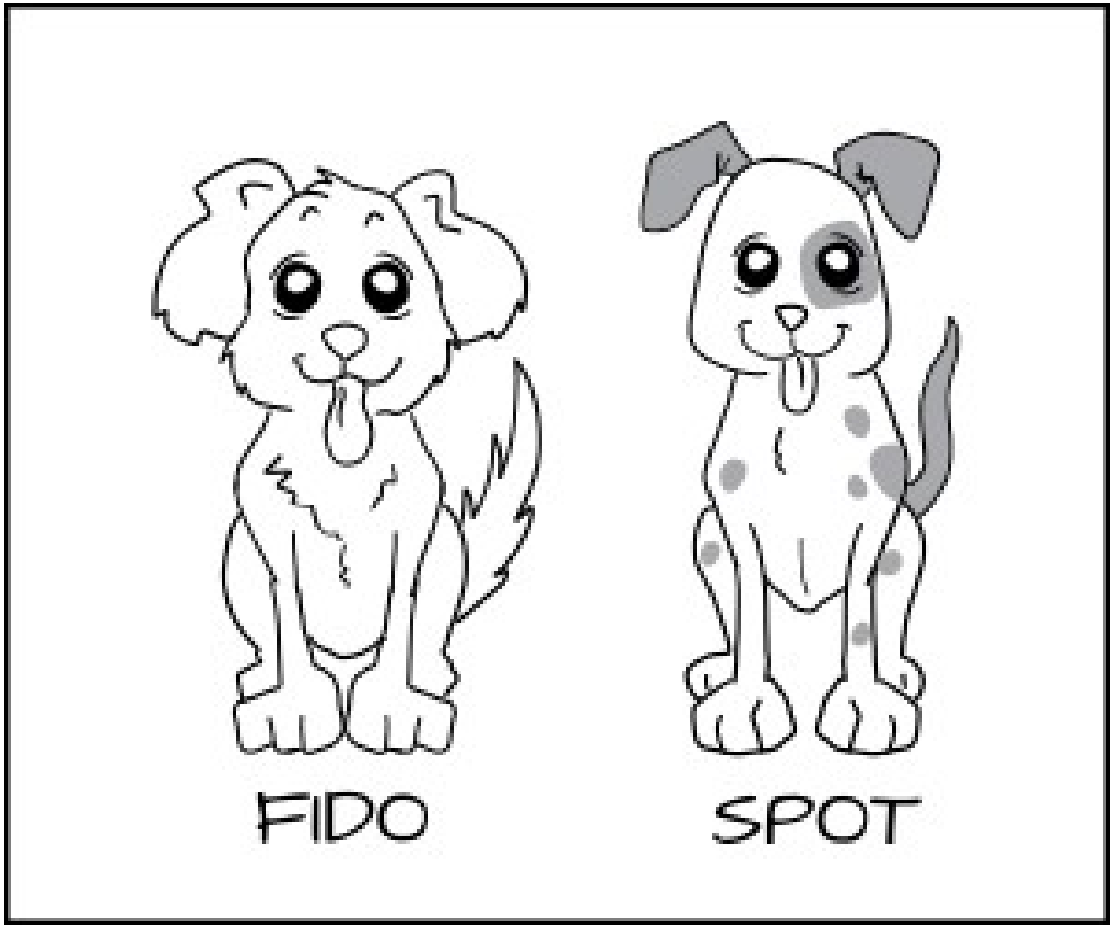
Type Enforcement

- Protects the host system from container processes
- Container processes can only read/execute /usr files
- Container processes only write to container files.
- Process type svirt_lxc_net_t
- file type svirt_sandbox_file_t

MCS Enforcement

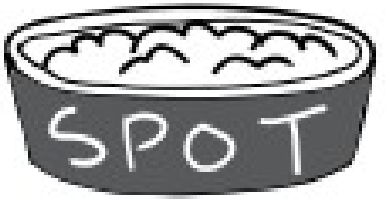
Multi Category Security

MCS Enforcement



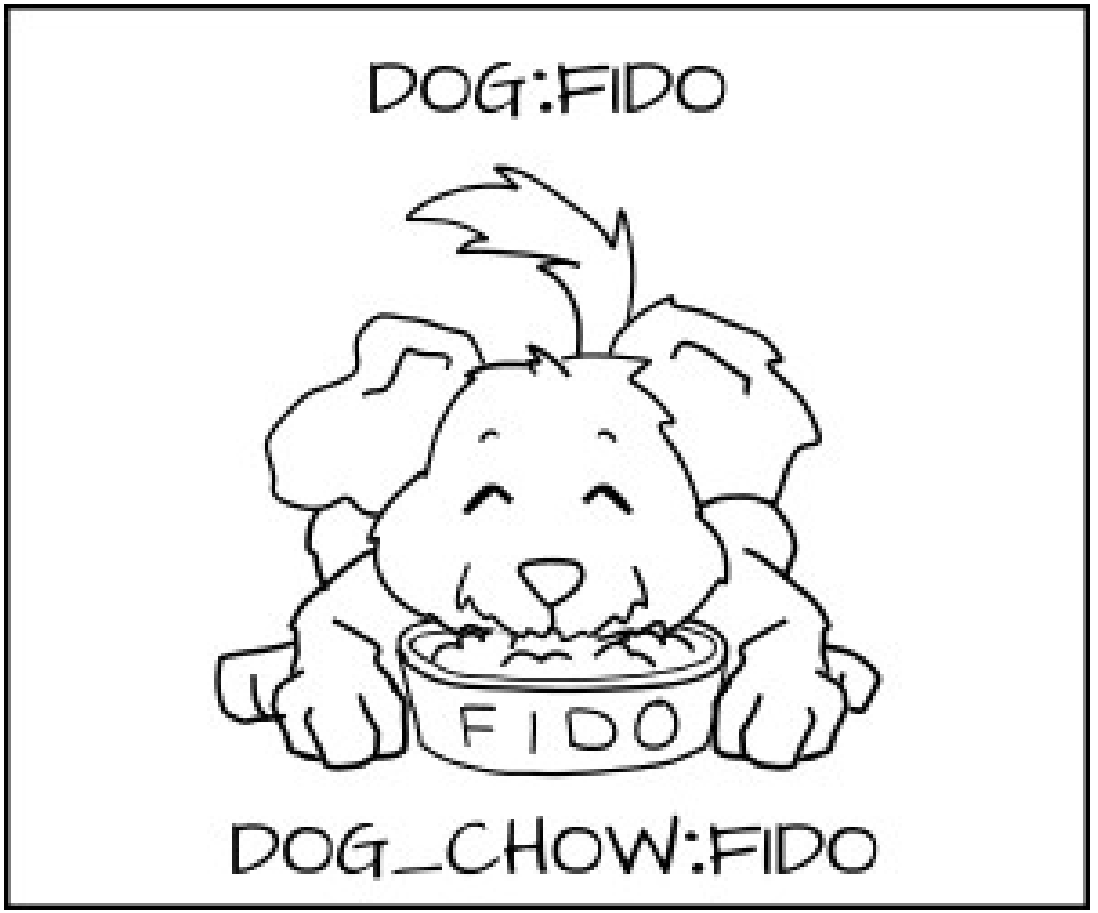
MCS Enforcement

DOG_CHOW:FIDO

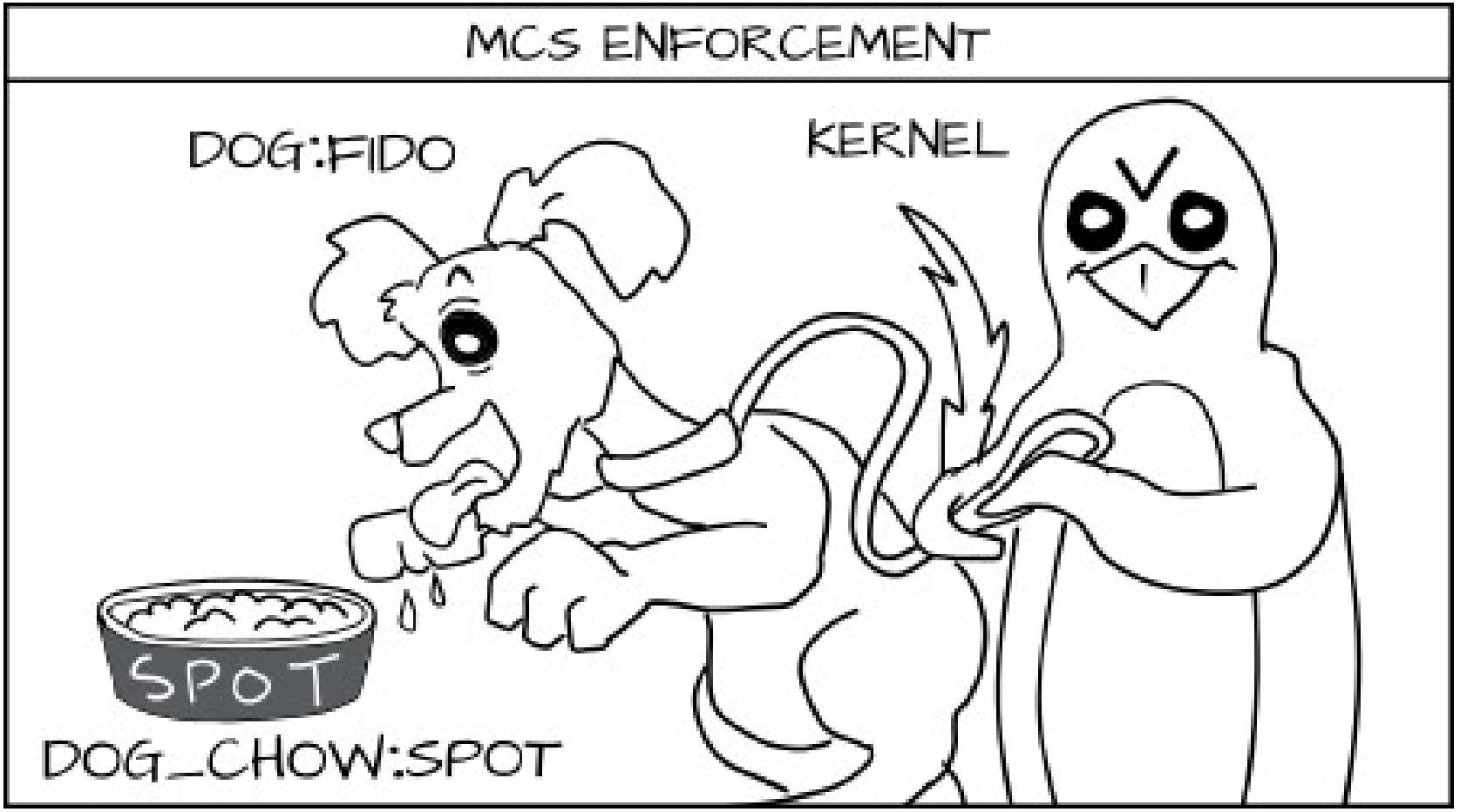


DOG_CHOW:SPOT

MCS Enforcement



MCS Enforcement



MCS Enforcement

- Protects containers from each other.
- Container processes can only read/write their files.
- Docker daemon picks unique random MCS Label.
  - s0:c1,c2
- Assigns MCS Label to all content
- Launches the container processes with same label

Docker Without SELinux

Is like Tupperware without the burp

Future - seccomp

- Shrink the attack surface on the kernel
- Eliminate syscalls
- kexec_load, open_by_handle_at, init_module, finit_module, delete_module, iopl, ioperm, swapon, swapoff, sysfs, sysctl, adjtimex, clock_adjtime, lookup_dcookie, perf_event_open, fanotify_init, kcmp
- block 32 bit syscalls
- block old weird networks

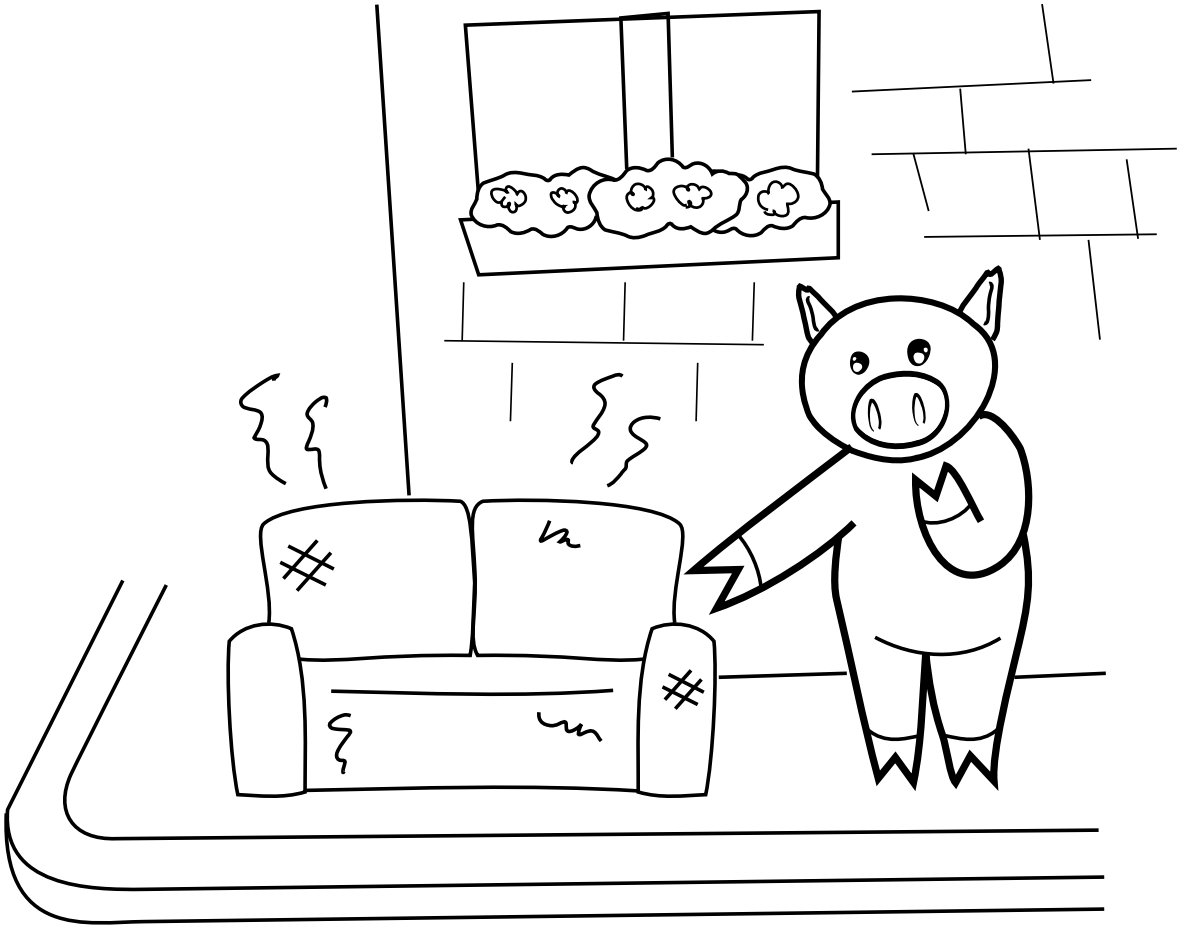
Future – User Name Space

- Map non root user to root within container
- Available in docker-1.9 (Limited)
- Only used to protect the host from containers, not used to protect containers from each other.
- Can we protect one container from another?
- No file system support

Future – Clear Linux Containers

- Use KVM with slimmed down kernel
- Intel Introduced
- Better isolation
  - Better SELinux protection
- Breaks certain use cases
- Supports docker containers
- Starts container in .2 seconds

Chapter 4
How do you furnish the pigs apartment?



How do I secure content inside container?

LINUX 1999

LINUX 1999

Where did you get your software?

LINUX 1999

Where did you get your software?

Go to yahoo.com or AltaVista.com and google it?

LINUX 1999

Where did you get your software?
Go to yahoo.com or AltaVista.com
and google it?
Find it on rpmfind.net, download and install.

LINUX 1999

Where did you get your software?
Go to yahoo.com or AltaVista.com
and google it?
Find it on rpmfind.net, download and install.
Hey I hear there is a big Security
vulnerability in Zlib.

LINUX 1999

Where did you get your software?
Go to yahoo.com or AltaVista.com
and google it?
Find it on rpmfind.net, download and install.
Hey I hear there is a big Security
vulnerability in Zlib.
How many copies of the Zlib vulnerability to
you have?

LINUX 1999

Where did you get your software?
Go to yahoo.com or AltaVista.com
and google it?
Find it on rpmfind.net, download and install.
Hey I hear there is a big Security vulnerability in Zlib.
How many copies of the Zlib vulnerability to you have?
I have no clue!!!

Red Hat to the Rescue

Red Hat to the Rescue

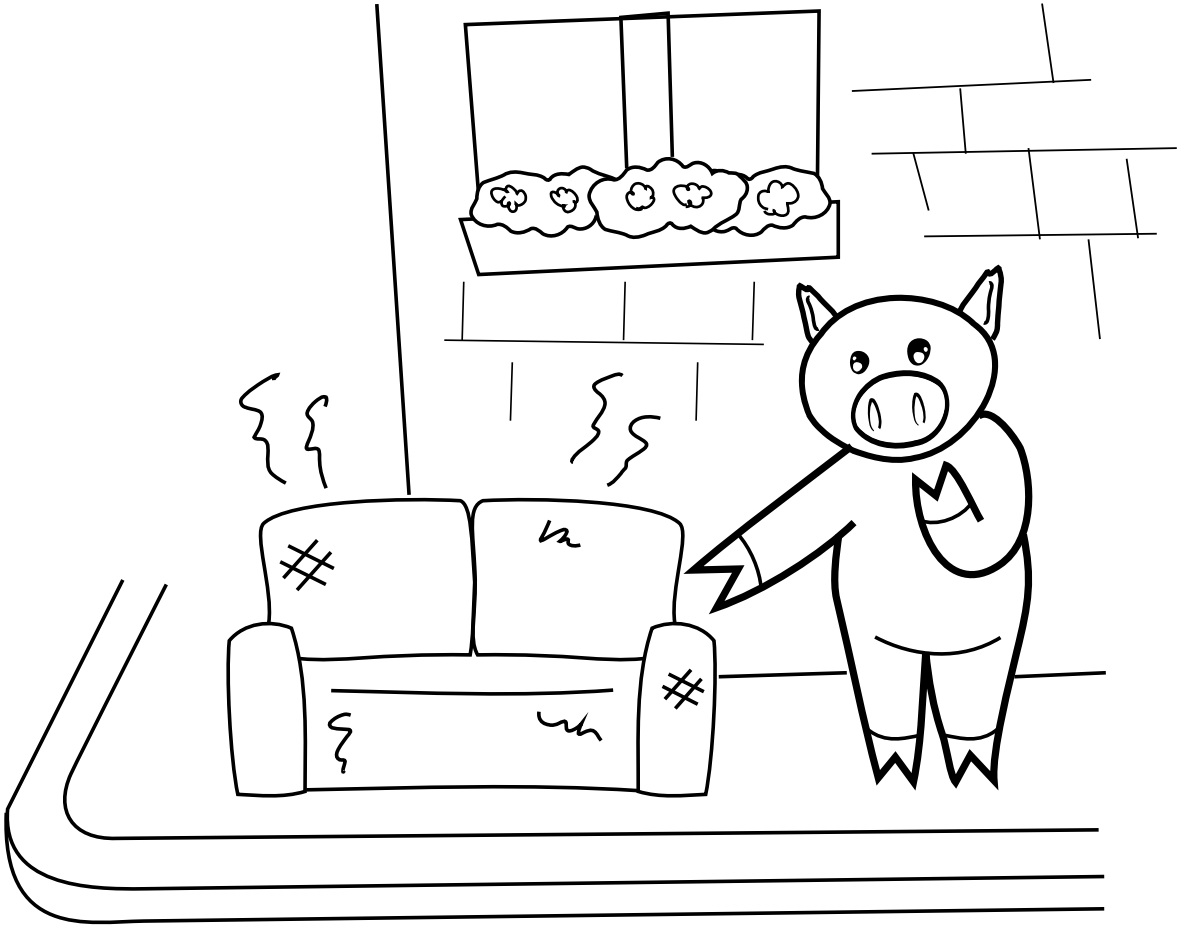
Red Hat Enterprise Linux solved this problem

Red Hat to the Rescue

Red Hat Enterprise Linux solved this problem

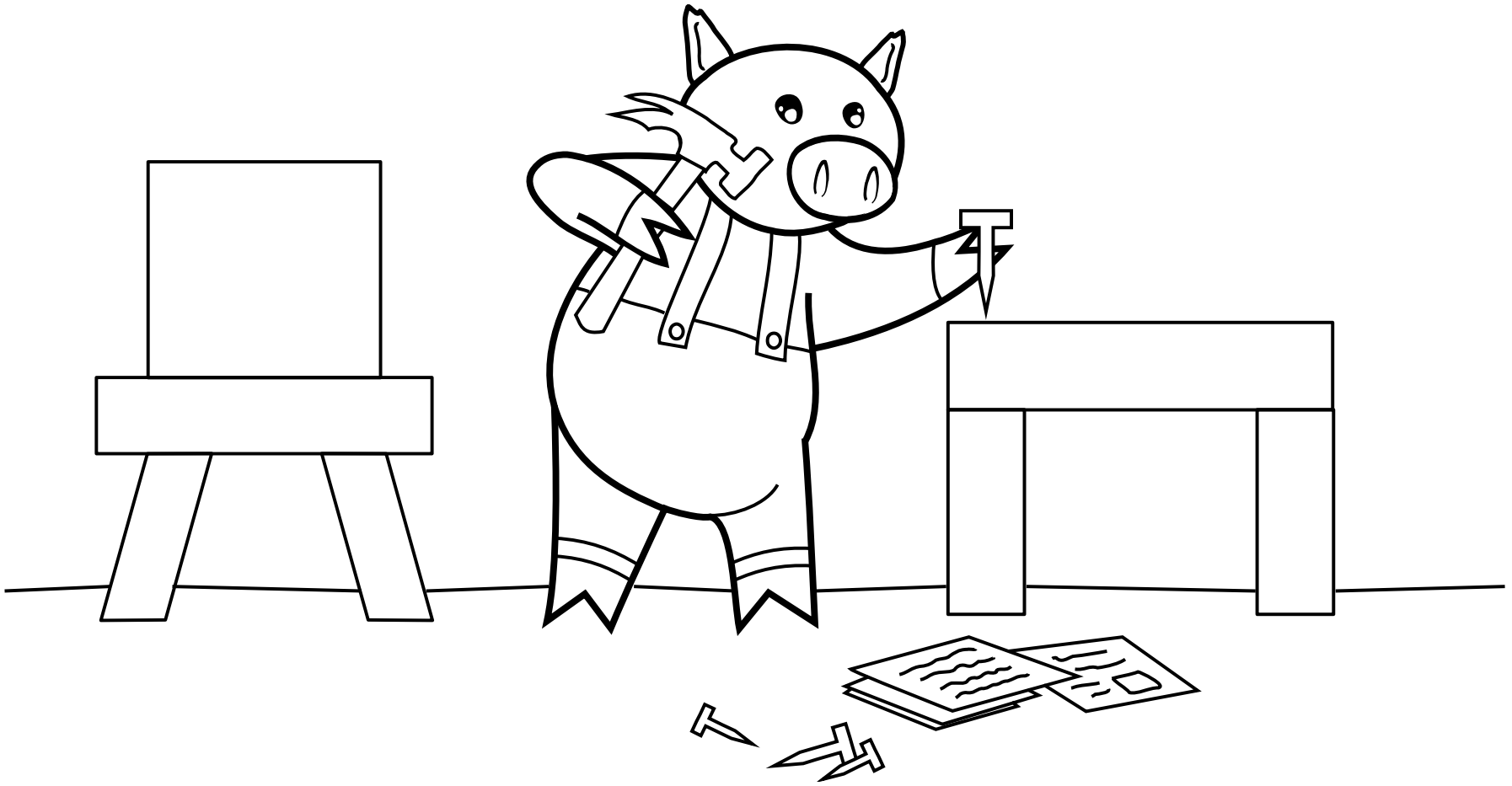
Certified software and hardware platforms

How do you furnish the pigs apartment?



People have no idea of quality of software in docker images

How do you furnish the pigs apartment?



Or they build it themselves.

Lets Talk about DEV/OPS

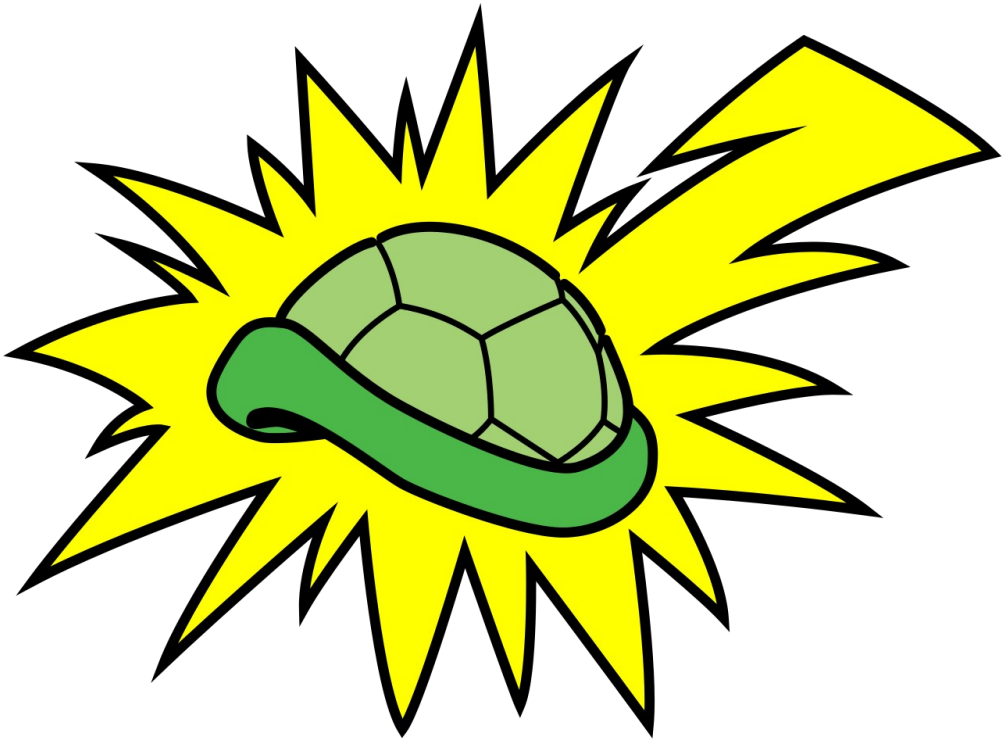
Lets Talk about DEV/OPS

Containers move the responsibility for security updates from the Operator to the Developer.

Lets Talk about DEV/OPS
Containers move the responsibility for security updates from the Operator to the Developer.
Do you trust developers to
fix security issues in their images?


C-SPAN Today
12-08-87
21:09:41

What happens when the next Shell Shock hits



Unpatched Vulnerabilities Common on Docker Hub Images

POSTED BY: PAUL MAY 29, 2015 10:41 2 COMMENTS

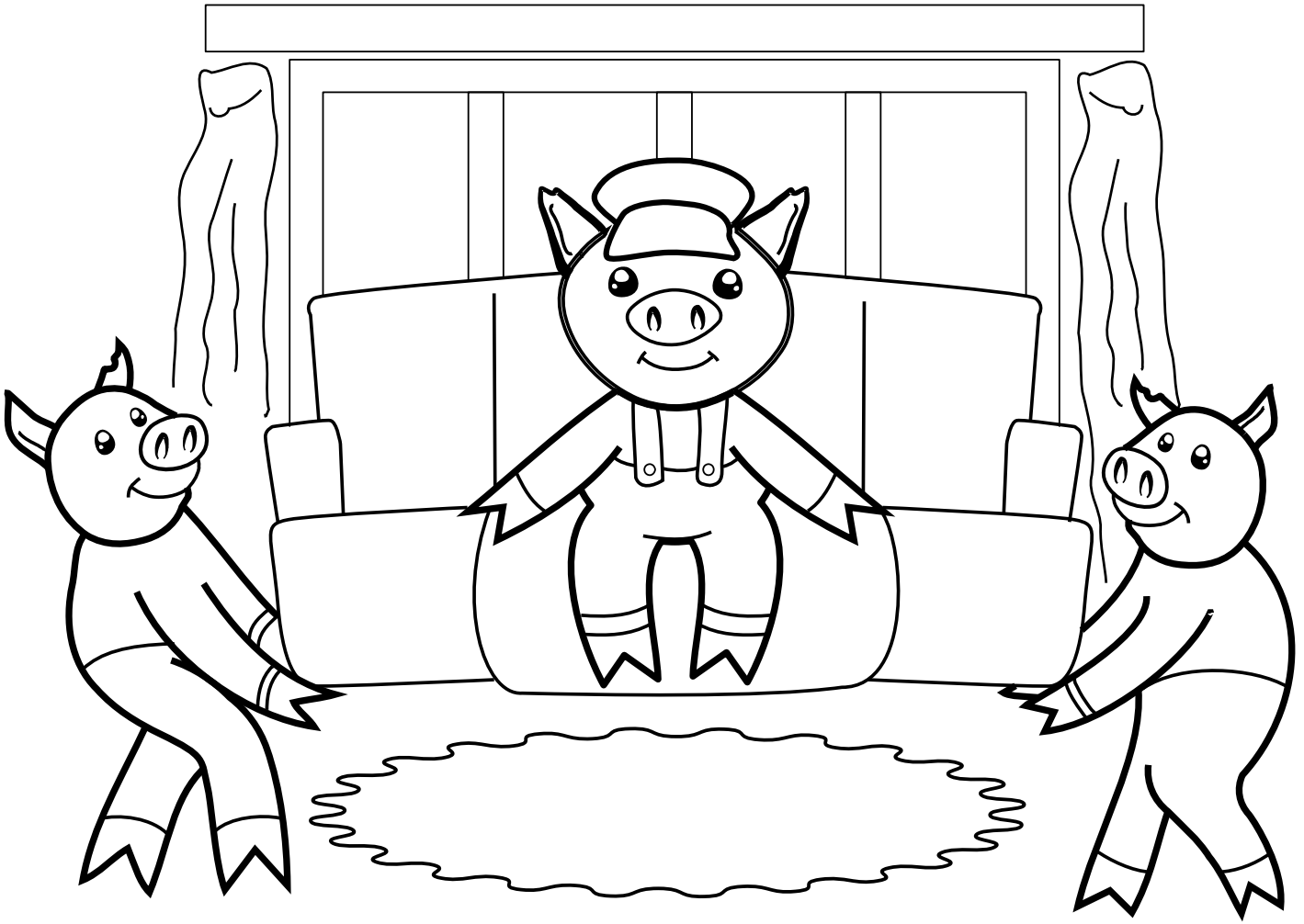
A survey of Docker repositories found that critical vulnerabilities are common in both official and general repositories.

In-brief: A survey out from the firm Banyan finds that official and general repositories on Docker Hub are rife with serious and exploitable software vulnerabilities, including Heartbleed, Shellshock and Poodle.

A survey out from the firm Banyan finds that code repositories on Docker Hub are rife with serious and exploitable software vulnerabilities, including Heartbleed, Shellshock and Poodle.

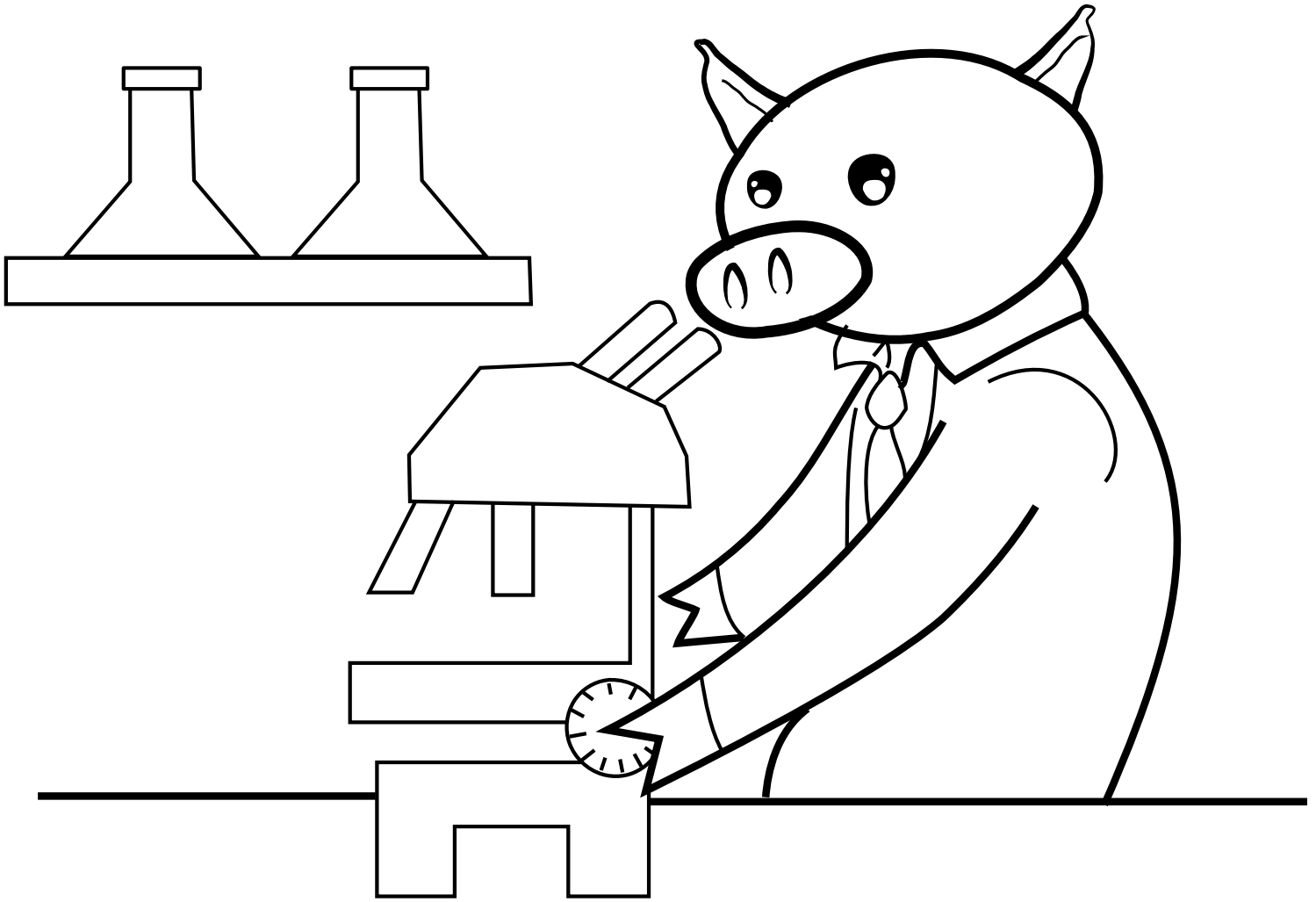
More than 30 percent of official repositories on Docker Hub contain software images that were found to be "highly susceptible to a variety of security attacks," according to a report by Jayanth Gummaraju, Tarun Desikan and Yoshio Turner of the firm Banyan. The report is just the latest to warn of the lingering effects of even high-profile flaws like Heartbleed, which

How do you furnish the pigs apartment?



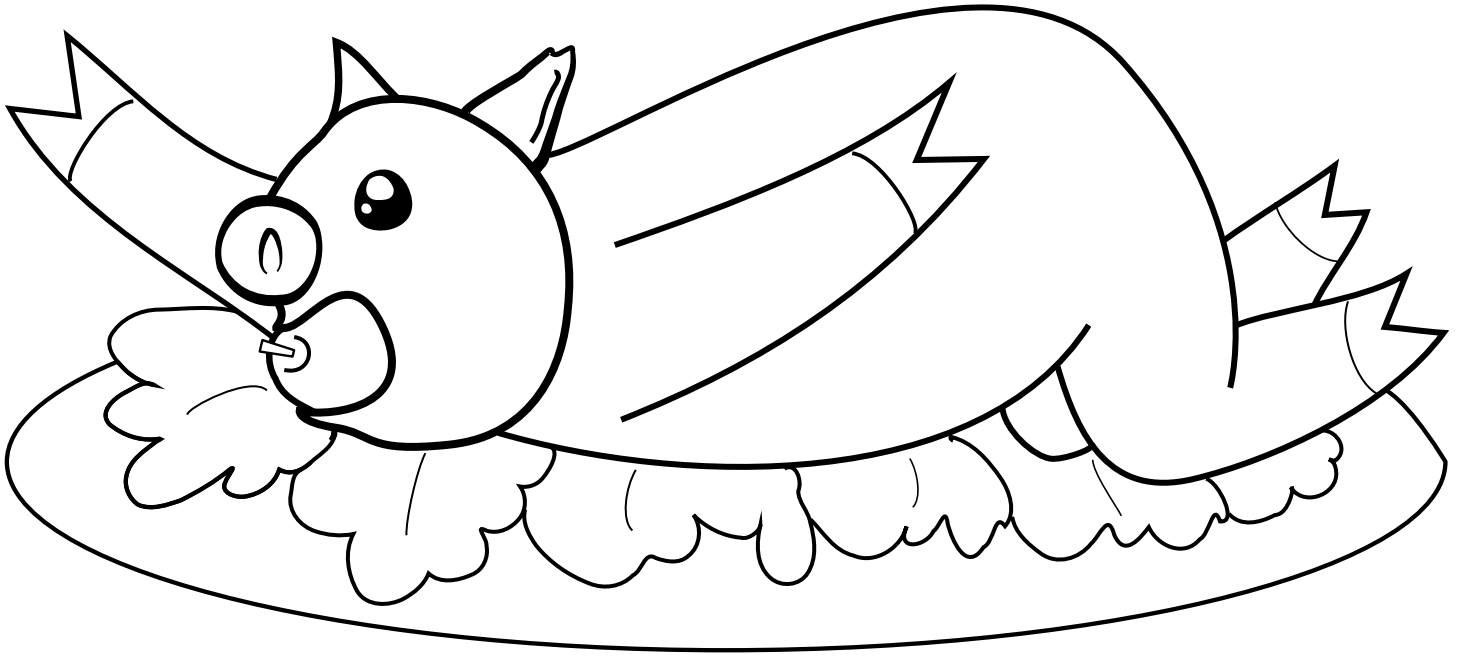
RHEL Certified Images

How do you furnish the pigs apartment?



Red Hat Support and Security teams partnering with you to secure your software

Don't let this be you.



Questions?