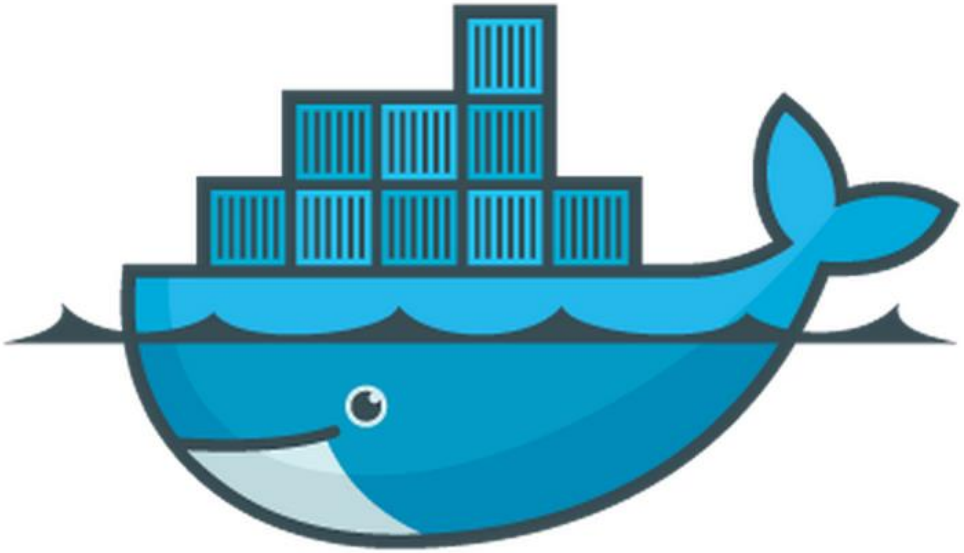


Docker Overview

jnagal@

Containerizing everything @ Google

Containers at scale.
Resource Isolation.
lmctfy
libcontainer

Docker : What & Why

Machine or Application containers

Build Once, Configure Once.
Deploy Everything*
Everywhere*
Reliably & Consistently
Efficiently
Cheaply

Docker Features

Image Management

Resource Isolation

File system Isolation

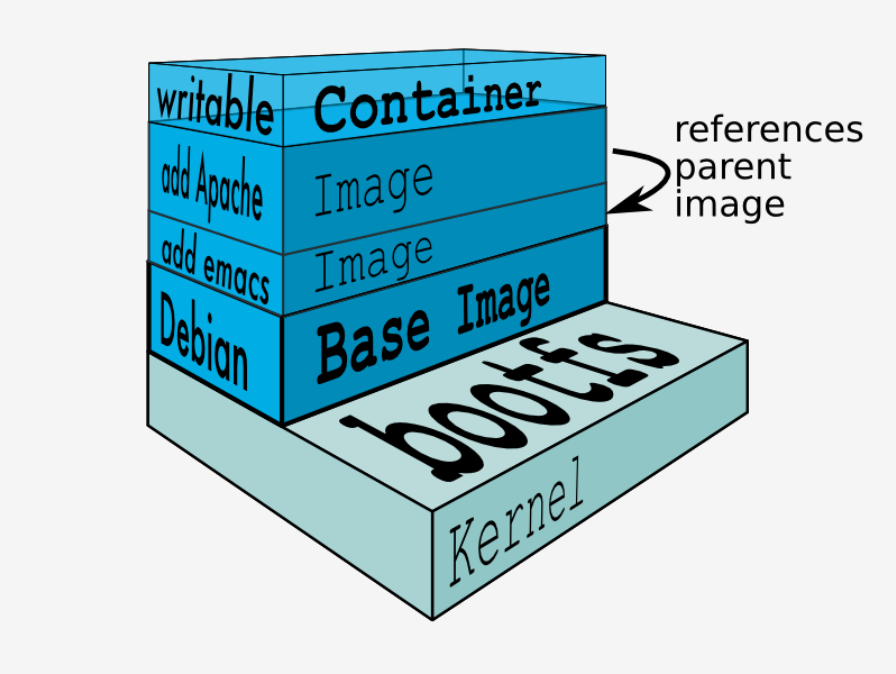
Network Isolation

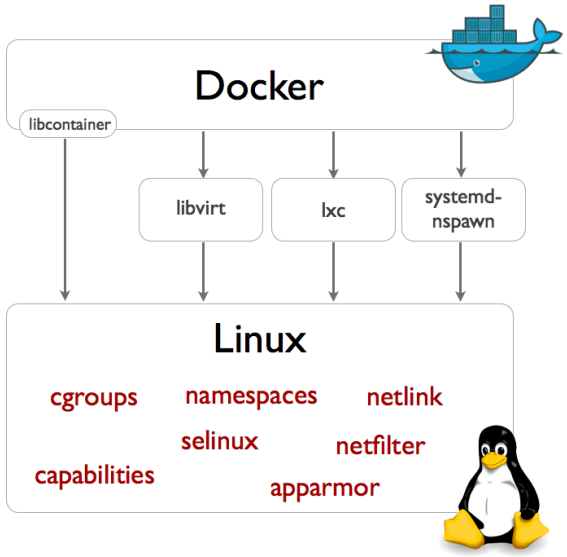
Change Management

Sharing

Process Management

Docker Components





Docker Grounds up: Resource Isolation

Cgroups : Isolation and accounting

- cpu
- memory
- block i/o
- devices
- network
- numa
- freezer

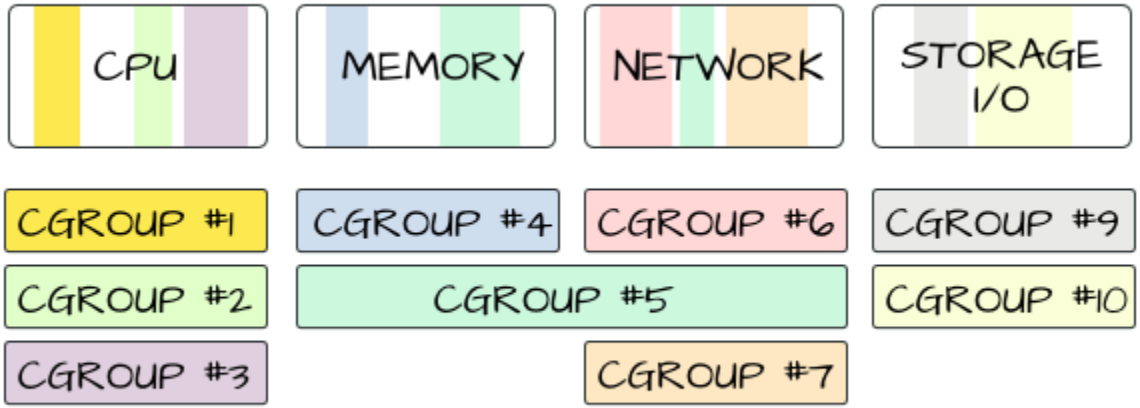


image credit: mairin

Docker Grounds up: Namespaces

- Process trees.
- Mounts.
- Network.
- User accounts.
- Hostnames.
- Inter-process communication.

```
pid_t pid = clone(..., flags, ...)

    CLONE_NEWUTS          hostname,
domainname
    CLONE_NEWIPC          IPC objects
    CLONE_NEWPID          Process IDs
    CLONE_NEWNET          Network
configuration
    CLONE_NEWNS           File system mounts
    CLONE_NEWUSER                 User and
Group IDs

setns(int fd, int nstype)
        CLONE_NEWIPC
        CLONE_NEWNET
        CLONE_NEWUTS

Also: unshare(flags)
```

Docker Grounds up: Add Security

- Linux Capabilities
  - Drops most capabilities.
  - Enable what a task needs.
- GRSEC and PAX
- SELinux
- AppArmor

image credit: Leo Reynolds

Docker Grounds up: Filesystem

File-system Isolation:

Building a rootfs dir and chroot into it.

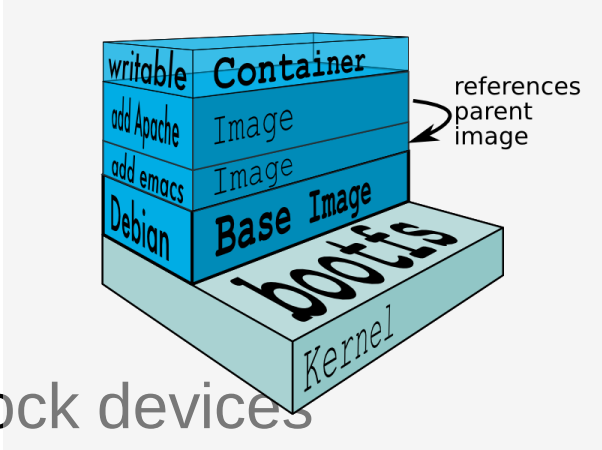
With mount namespace, use pivot-root.

Features:

Layering, CoW, Caching, Diffing

Solutions:

UnionFS, Snapshotting FS, CoW block devices



Docker Grounds up: Filesystem

	Union Filesystems	Snapshotting Filesystems	Copy-on-write block devices
Provisioning	Superfast Supercheap	Fast Cheap	Fast Cheap
Changing small files	Superfast Supercheap	Fast Cheap	Fast Costly
Changing large files	Slow (first time) Inefficient (copy-up!)	Fast Cheap	Fast Cheap
Diffing	Superfast	Superfast	Slow
Memory usage	Efficient	Efficient	Inefficient (at high densities)
Drawbacks	Random quirks AUFS not mainline !AUFS more quirks	ZFS not mainline BTRFS not as nice	Higher disk usage Great performance (except diffing)
Bottom line	**Ideal for PAAS and high density things**	**This is the Future (probably)**	**Dodge Ram 3500**

From: Jérôme Petazzoni

Docker Grounds up: Processes & Networking

We have resources, isolation, and file system management.

Docker daemon handles starting/stopping processes with:

- Attach logic
- Logs
- TTY management
- Docker run options
- Events and container state

Network Management

- NAT, Bridge, Veth
- Expose
- Links

Docker Grounds up: Images

Create and share images
Push, pull, commit images.
Registry (public, private) and index.
Dockerfiles

Orchestration:
Linking Containers
Multi-host linking
Dynamic discovery

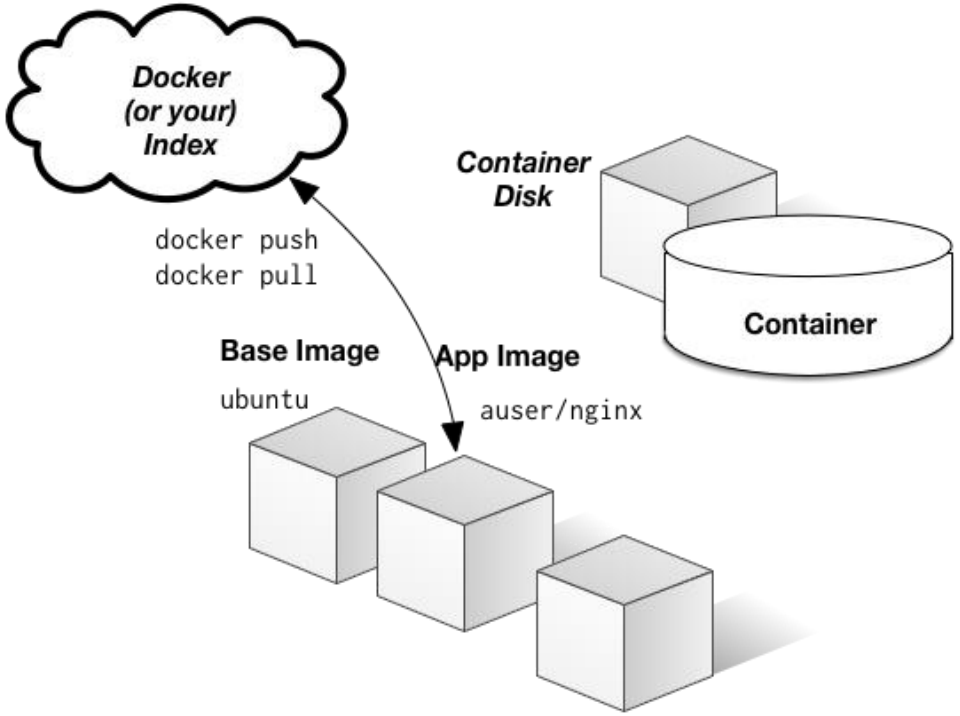


image: jbarratt

Docker Codewalk

github.com/dotcloud/docker/

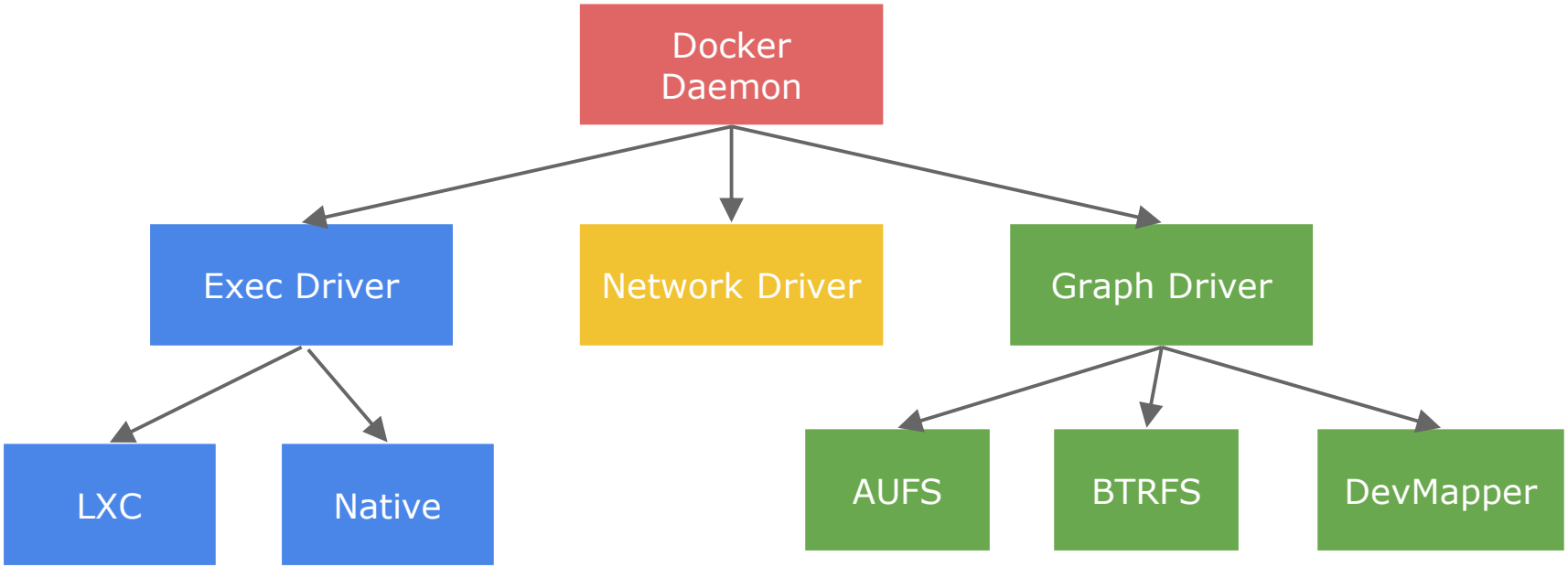
api : docker client and server api
daemon : Managing containers and images
engine: commands/jobs processing
graph: store for versioned filesystem images and their relationship.
registry: handling registry and repository.
links: Linking containers.

integration-cli: Integration tests.
docs: documentation.
pkg: collection of standalone utility packages that are not docker specific.

-> Great place to start contributing.

Time for actual walkthrough...

Docker Codewalk : docker/daemon



Docker Codewalk : pkg

github.com/dotcloud/docker/pkg

libcontainer: cgroup and namespaces. Uses lot of other utility packages.
nsinit binary.

apparmor, selinux, label : applying security profiles.

mount, signals : system utilities.

iptables, networkfs, netlink : network utilities.

term: terminal handling

systemd

Let’s look through some of these.

Thanks!

Rohit Jnagal

jnagal@google
@jnagal

Kesden Additional Slide:

- VMs vs Containers
  - VMs virtualize Hardware, OS, etc
  - Containers virtualize application environment

- Containers may not provide as strong a security model
  - What is virtualized? What is real?
  - What about the super users?
  - Things that are sideways, e.g. virtual file system, devices, etc

- Generally use containers to virtualize for one application in shared host or VM
- Use VMs to virtualize for many applications
- VMs probably 2-3x as resource intensive as containers
  - Corollary: Can get 2-3x as much from containerized solution vs VMs