What are containers anyway?

Maciej Pasternacki <maciej@3ofcoins.net>

Polish BSD User Group, 2018-10-11


@mpasternacki

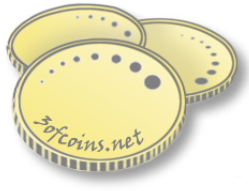

3ofcoins

Outline

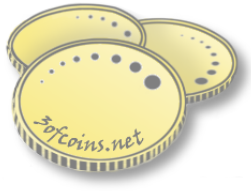
Not a New Tech

The Container Mindset

The Moving Parts

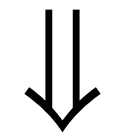


Not a New Tech

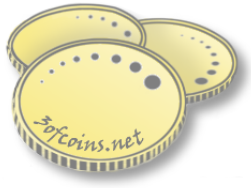


OS–level Virtualization

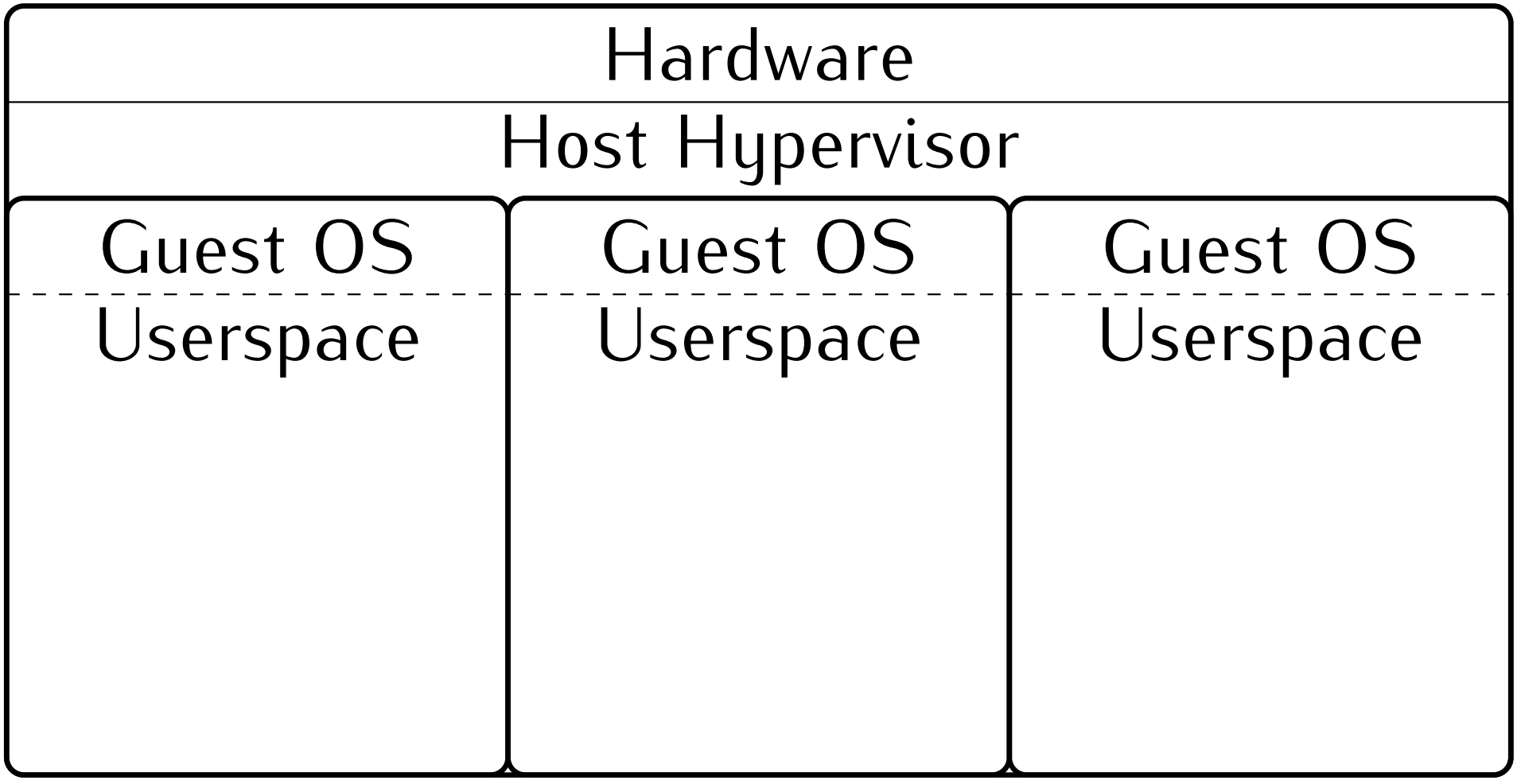
Single host kernel

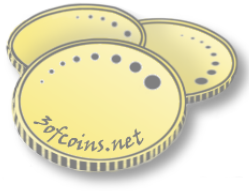


Multiple guest userspaces

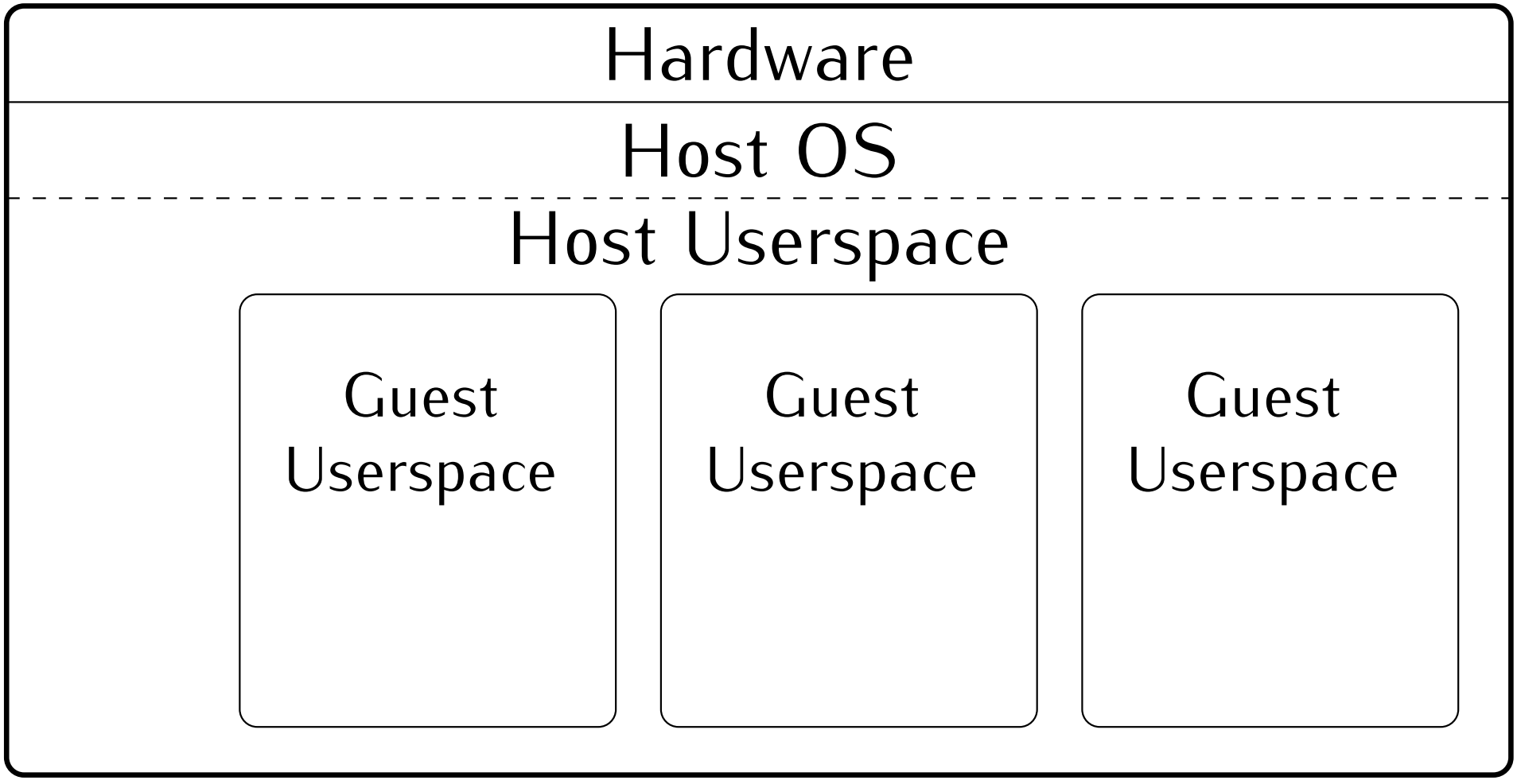


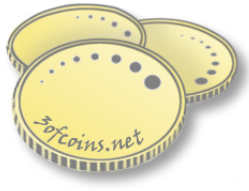
Full Virtualisation





OS–level Virtualisation

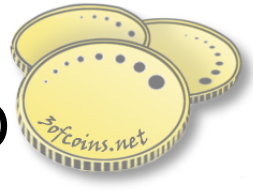




OS–level Virtualization

versus full virtualization

- Less isolation
- Guest must have same OS as host[1]
- Lower overhead
- Adjustable isolation level
- Resource sharing is possible



[1] or binary–compatible: Solaris branded zones, FreeBSD Linuxulator, CloudABI

1982

The Stone Age

chroot(2)

```
CHROOT(2)                 FreeBSD System Calls Manual                CHROOT(2)

NAME
     chroot — change root directory

LIBRARY
     Standard C Library (libc, -lc)

SYNOPSIS
     #include <unistd.h>

     int
     chroot(const char *dirname);

DESCRIPTION
     The dirname argument is the address of the pathname of a directory,
     terminated by an ASCII NUL.  The chroot() system call causes dirname to
     become the root directory, that is, the starting point for path searches
     of pathnames beginning with ‘/’.
```

1998–2012

The Industrial Age

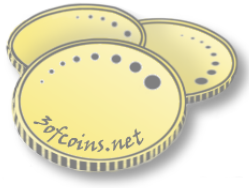
2000 FreeBSD Jail

2001 Linux–VServer, Virtuozzo

2002 Linux namespaces

2005 OpenVZ, Solaris Containers

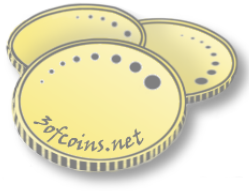
2008 Linux cgroups, LXC



1998–2012

The Industrial Age

- Isolated filesystem, process tree, networking
- Restricted interaction between environments
- Restricted administrative system calls
- Resource usage limits

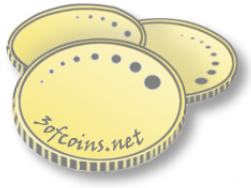


VM Mindset

Guest is a complete system:

- managed from the inside
- runs multiple services
- long-running and mutable
- opaque to host

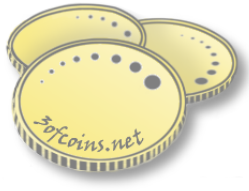
Management overhead of a whole server



2013

Modern Age

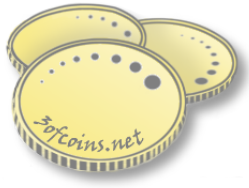
Jan 2013	Docker
Oct 2013	lmctfy†
Dec 2014	App Container Specification, CoreOS Rocket
Jan 2015	Jetpack†



2013

Modern Age

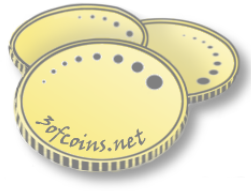
- Inspired by PaaS, service–oriented
- Guest managed from the outside
- Immutable, distributable images
- Fast copy–on–write provisioning



2014–2015

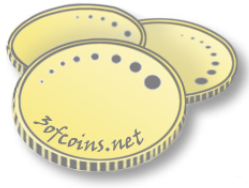
Standardization & Orchestration

Apr '14	Kubernetes
Dec '14	Docker Swarm
Apr '15	Google Borg paper[1]
Jun '15	Open Container Initiative
Jul '15	Cloud Native Computing Foundation
Sep '15	Hashicorp Nomad



[1]Mentions "operating Borg in production for more than a decade."

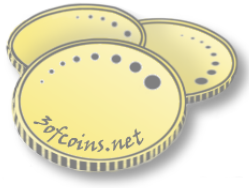
The Container Mindset



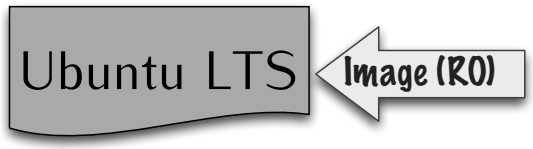
Container Mindset

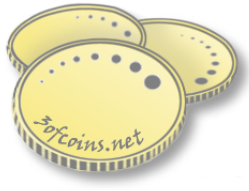
OS–level virtualization plus:

- Layered storage
- Explicit interaction points
- Immutable images, volatile containers
- Service–oriented

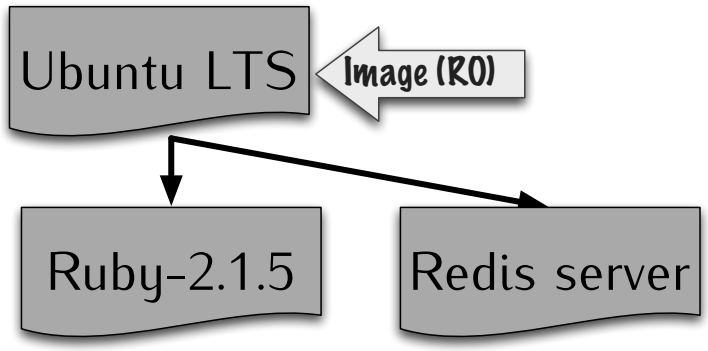


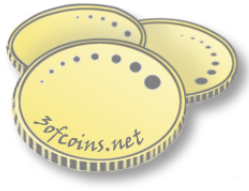
Layered Storage



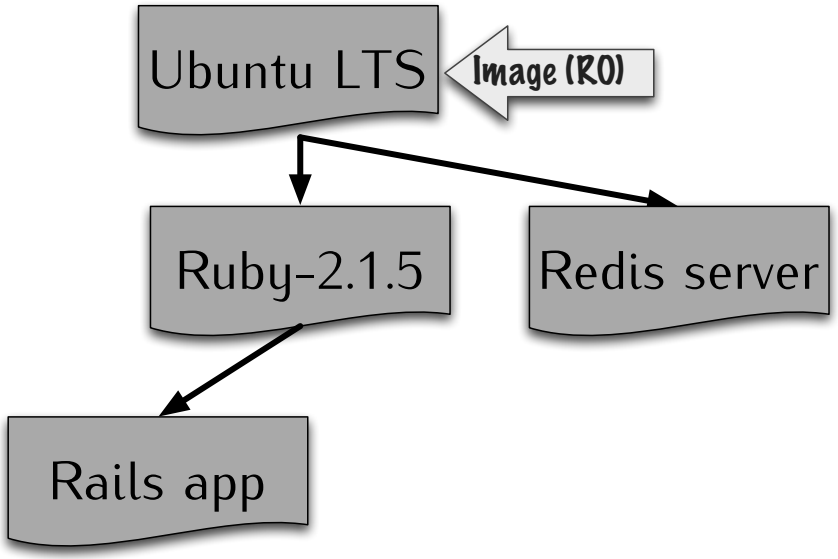


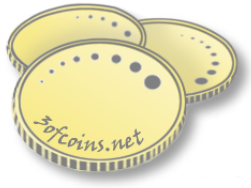
Layered Storage



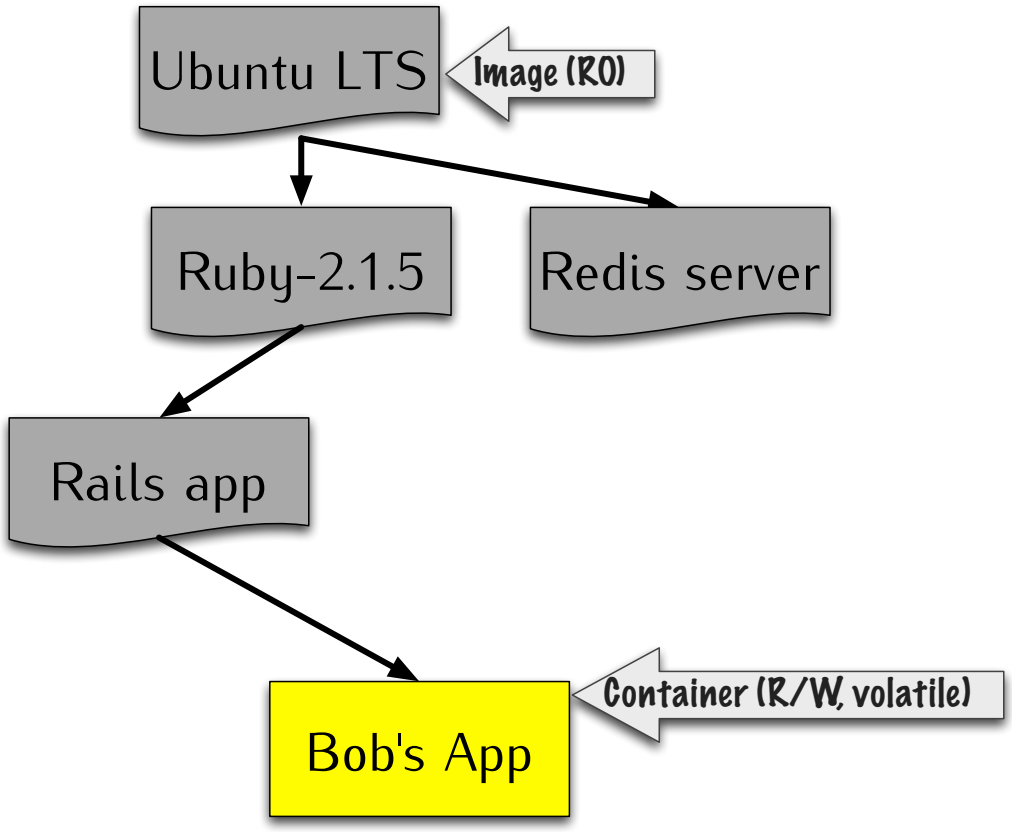


Layered Storage

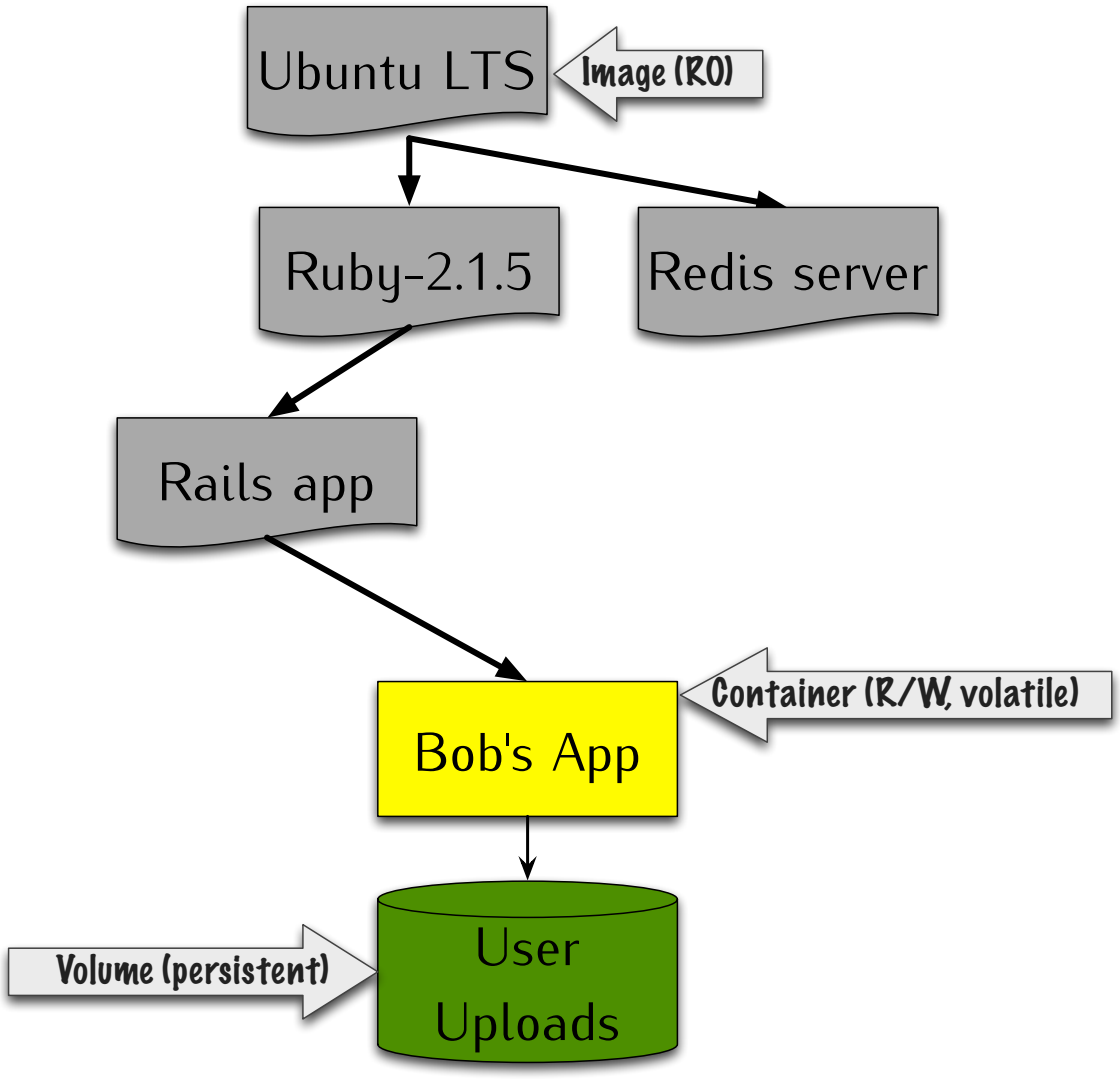




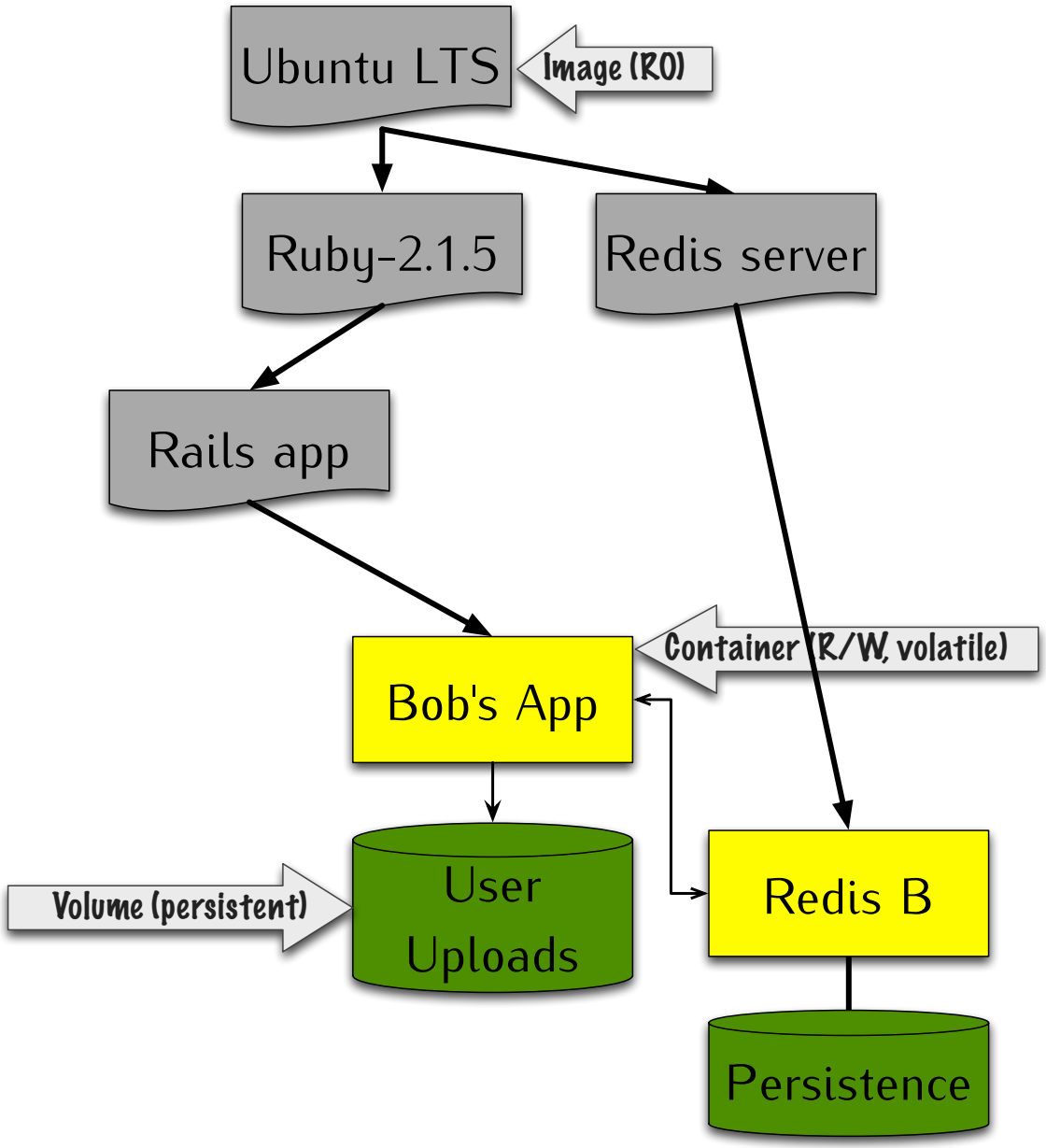
Layered Storage



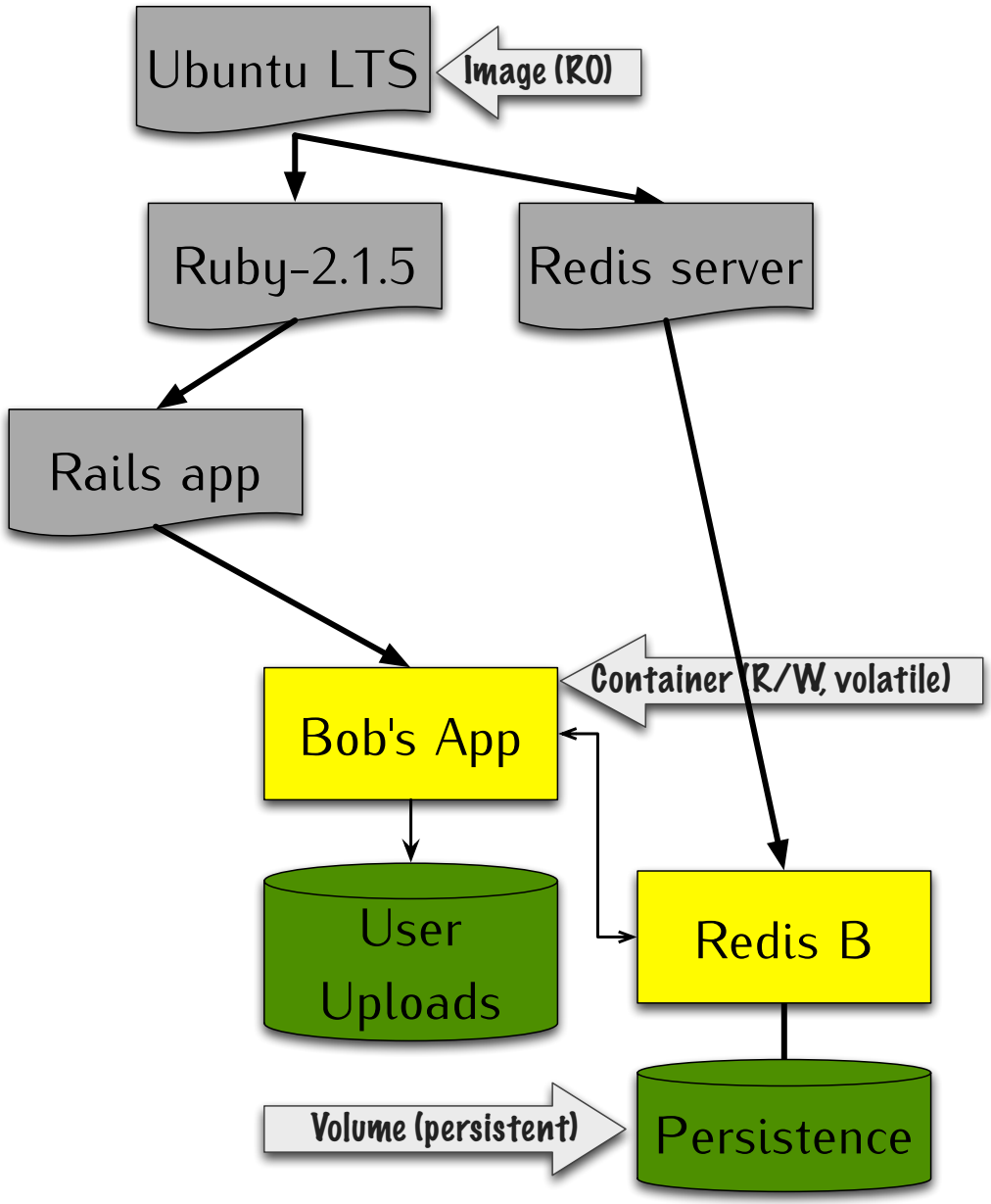
Layered Storage



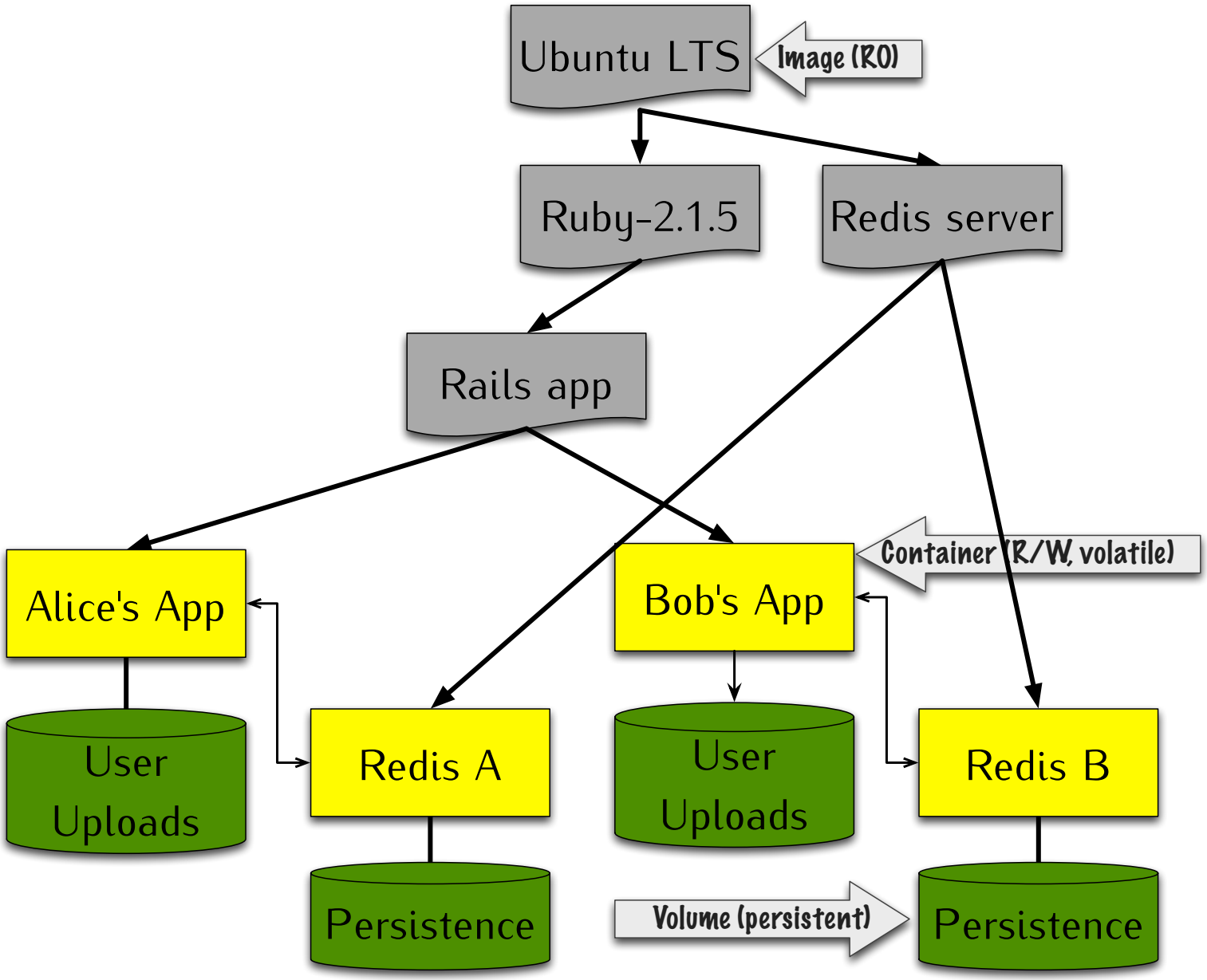
Layered Storage



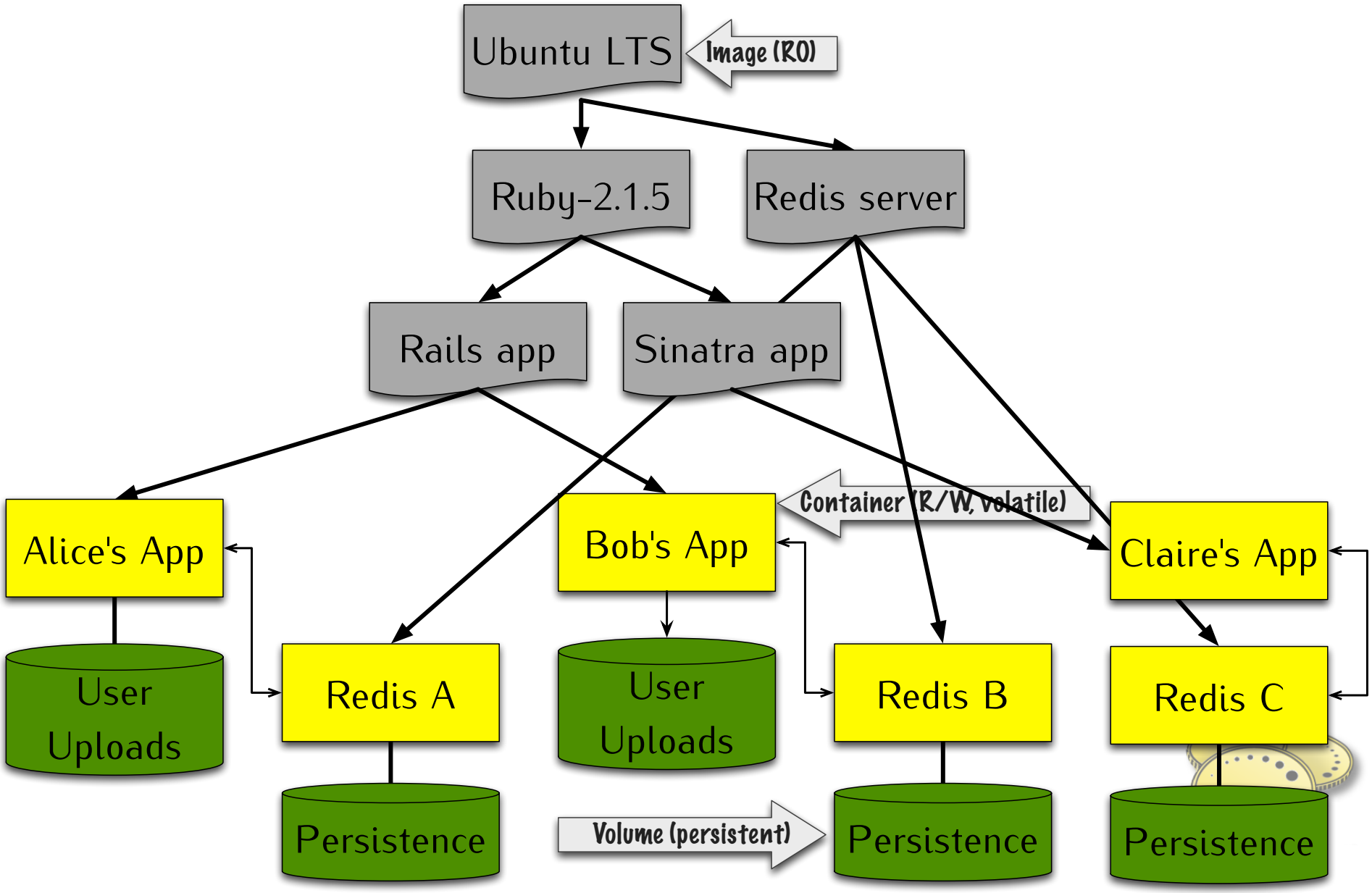
Layered Storage



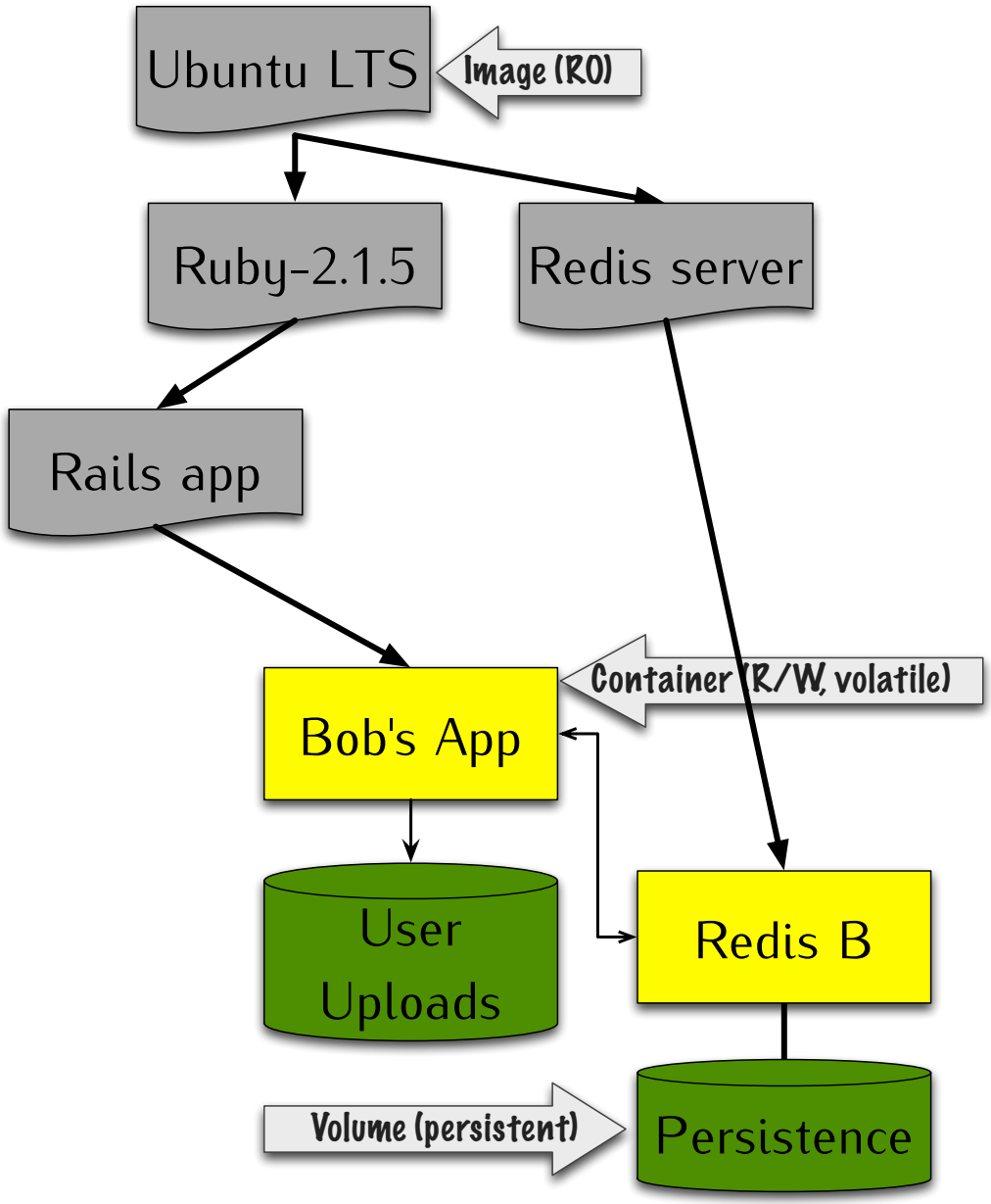
Layered Storage



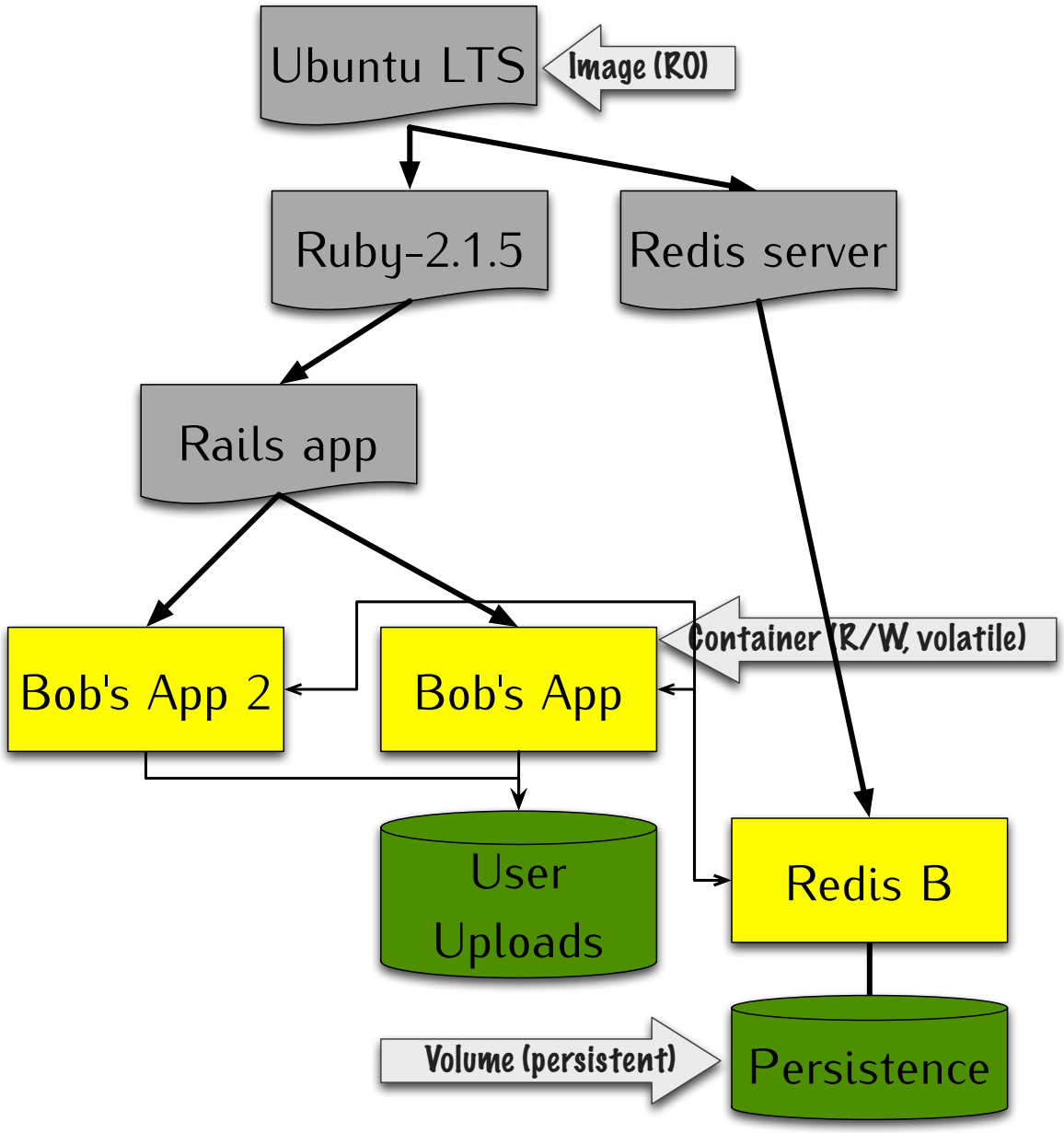
Layered Storage



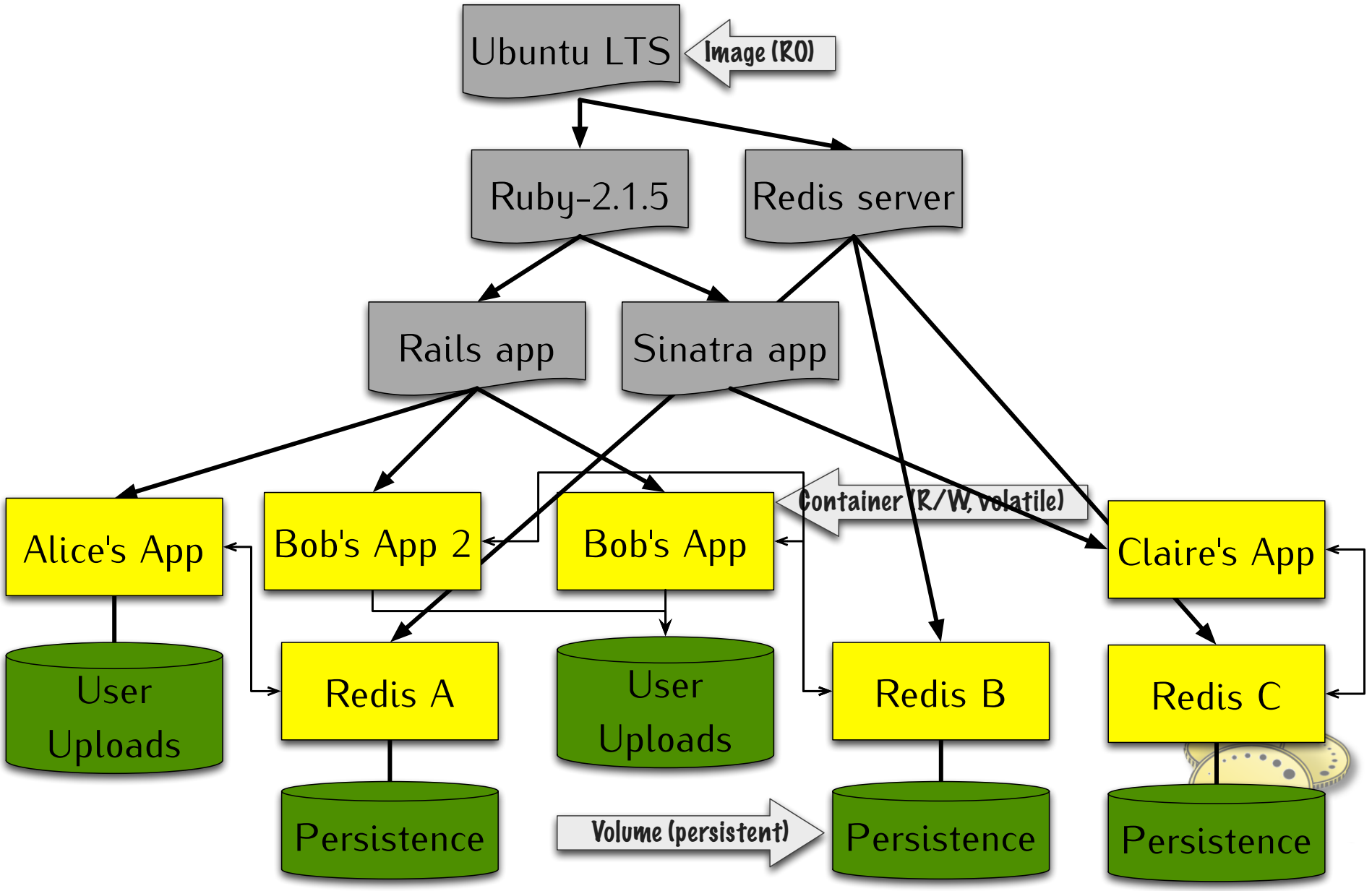
Layered Storage



Layered Storage

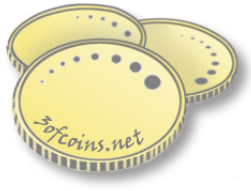


Layered Storage



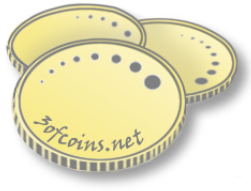
Explicit Interaction Points

- Command line arguments
- Environment variables
- Network ports
- Persistent/shared volumes
- Stdin, stdout, stderr
- Exit status



Immutability

- Images, once built, are read-only
- Containers' write layer is throwaway
- Volumes are persistent and shareable



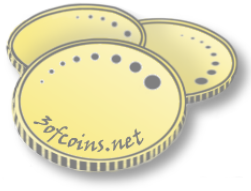
Immutability

- Images, once built, are read-only

  ⇒ reusable; uniquely identified; verifiable

- Containers' write layer is throwaway

- Volumes are persistent and shareable



Immutability

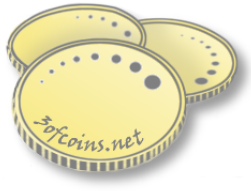
- Images, once built, are read-only

  ⇒ reusable; uniquely identified; verifiable

- Containers' write layer is throwaway

  ⇒ exchangeable; upgradeable

- Volumes are persistent and shareable



Immutability

- Images, once built, are read-only

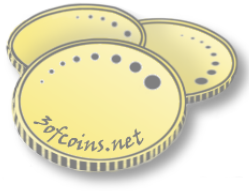
  ⇒ reusable; uniquely identified; verifiable

- Containers' write layer is throwaway

  ⇒ exchangeable; upgradeable

- Volumes are persistent and shareable

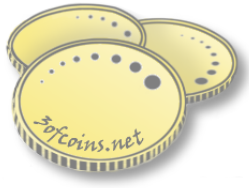
  ⇒ precious user data is clearly declared



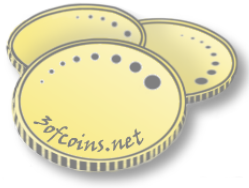
Service–oriented

- One container—one service
- Well–defined images can be shared & reused across applications
- Containers can be meaningfully managed & monitored by host

Management overhead of a single service



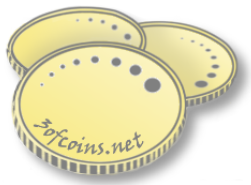
The Moving Parts



Open Container Initiative

Specify Critical Interfaces

- Runtime Specification
  + *runC*, a reference implementation
- Image Format Specification
- Distribution Specification

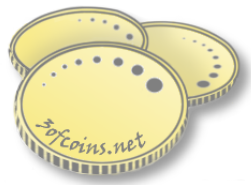


https://github.com/opencontainers/

OCI Runtime Spec

Configuration

- filesystem: root fs path, mounts, devices
- process: command+args, cwd, environment, uid+gids
- isolation: namespaces, rlimits, apparmor, capabilities, cgroups
- hostname
- hooks

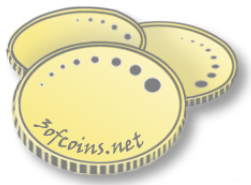


https://github.com/opencontainers/runtime-spec

OCI Runtime Spec

Runtime Operations

- `state ID` — query state
- `create ID PATH_TO_BUNDLE`
- `start ID`
- `kill ID SIGNAL`
- `delete ID`



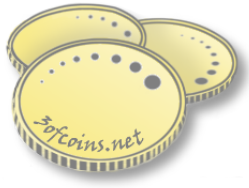
https://github.com/opencontainers/runtime-spec

OCI Runtime Spec

Implementations

General:

- opencontainers/runc (Go)
  - ⇒ *The* reference implementation
- oracle/railcar (Rust)
- giuseppe/crun (C)



OCI Runtime Spec

Implementations

Hypervisor–based:

- hyperhq/runv

  ⇒ KVM, Xen

- clearcontainers/runtime

  ⇒ Intel VT–x

- kata-containers/runtime

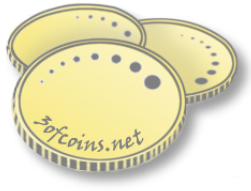
  ⇒ Intel VT-x, ARM Hyp, IBM Power Systems;
  combines ideas from runV & clearcontainers

OCI Runtime Spec

Implementations

Security–focused:

- google/gvisor (Go)
  ⇒ Intercepts system calls and acts as guest's kernel in the userspace.



OCI Runtime Spec

Implementations

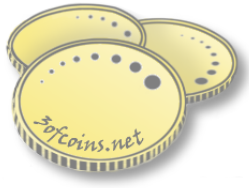
Wrappers:

- NVIDIA/nvidia-container-runtime

  ⇒ runC, patched to allow GPU access for guest

- projectatomic/bwrap-oci

  ⇒ C, uses Bubblewrap as sandbox



OCI Image Spec

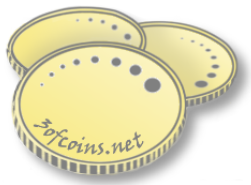
Image Layout

Content–addressable blobs:

`blobs/sha256/afff…2d51`

Referenced by content descriptors:

- Media type
- Digest (e.g. `sha256:afff…2d51`)
- Size
- Optional URLs

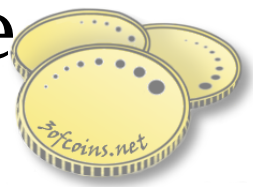


https://github.com/opencontainers/image-spec/

OCI Image Spec

Image Layout

- Index (JSON) points to manifest blobs
- Manifest (JSON) points to layer blobs and a configuration blob
- Layers are *tar* files with filesystem contents
- Configuration (JSON) is metadata and base configuration for runtime bundle

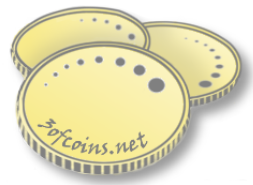


https://github.com/opencontainers/image-spec/

OCI Image Spec

Implementations

- projectatomic/skopeo
- openSUSE/umoci
- cloudfoundry/grootfs
- containerd/containerd
- containers/image, containers/build
- coreos/rkt
- Amazon Elastic Container Registry



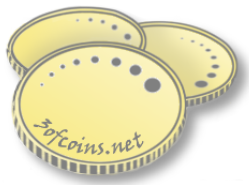
https://github.com/opencontainers/image-spec/

OCI Distribution Spec

API protocol for distribution of images

- Based on Docker Registry HTTP API V2
- Namespace–oriented URI layout
- Image verification
- Resumable push & pull
- Layer deduplication

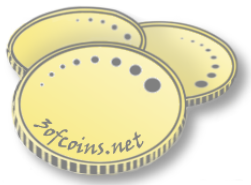
Seems to be WIP.



https://github.com/opencontainers/distribution-spec

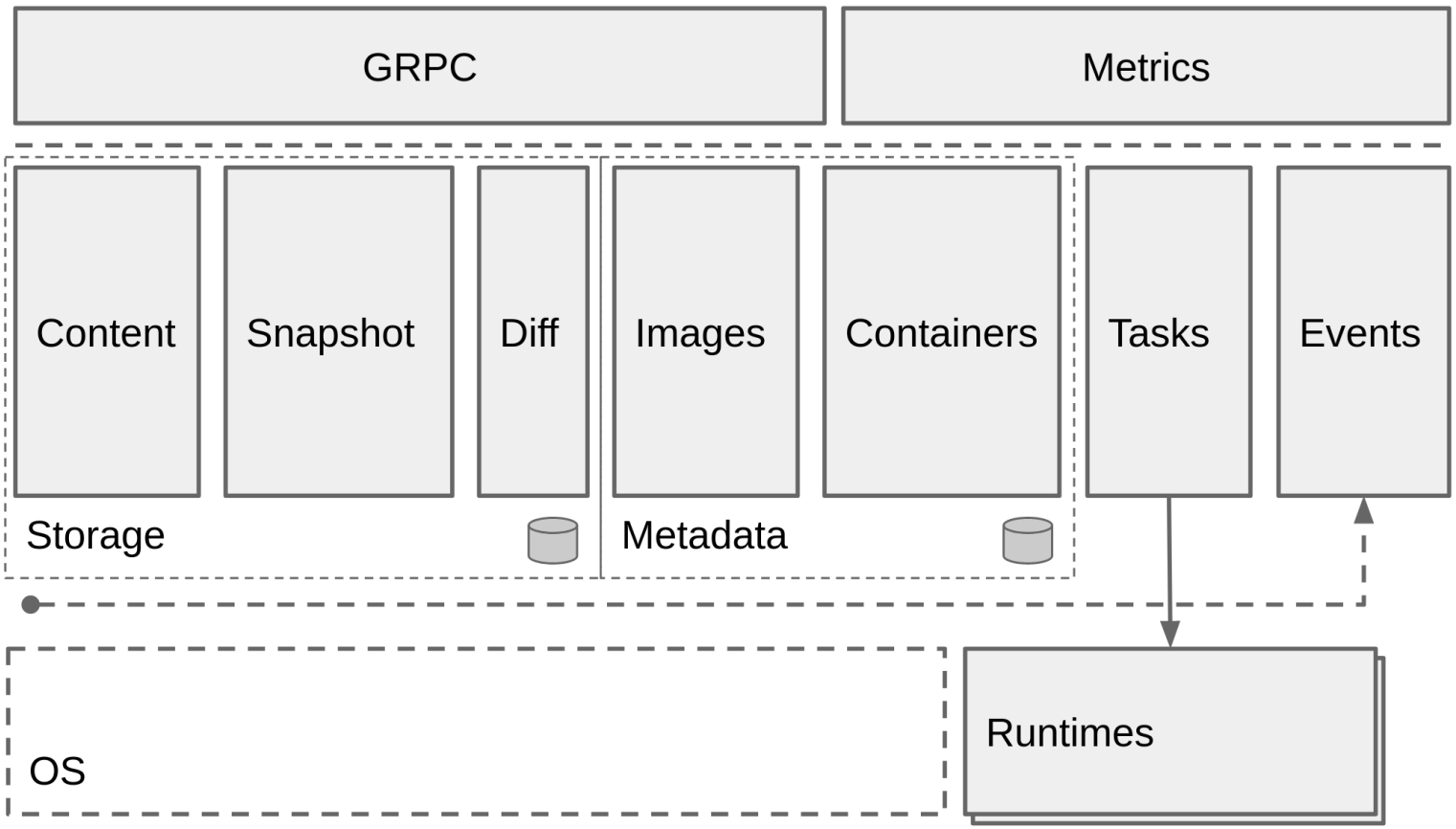
Containerd

- High–level runtime (image push/pull, storage, etc) on top of OCI runtime
- Exposes gRPC API
- CNCF member project



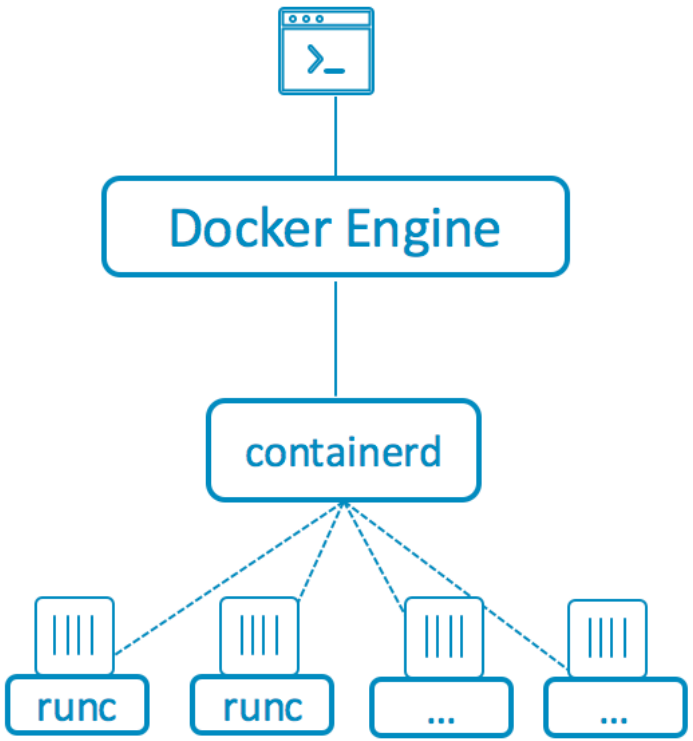
https://github.com/containerd/containerd

Containerd



https://github.com/containerd/containerd

Docker Architecture

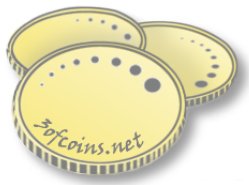


Same Docker UI and commands

User interacts with the Docker Engine

Engine communicates with containerd

containerd spins up runc or other OCI compliant runtime to run containers



https://blog.docker.com/2016/04/docker-engine-1-11-runc/

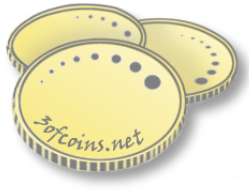
Kubernetes & CRI

Container Runtime Interface

Kubernetes' gRPC API for runtime plugins:

- dockershim

- containerd/cri
- kubernetes-incubator/rktlet
- kubernetes-sigs/cri-o

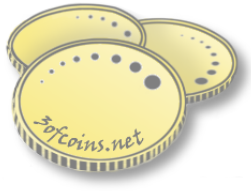


Kubernetes & CRI

Container Runtime Interface

Kubernetes' gRPC API for runtime plugins:

- dockershim (kubelet → dockershim → docker → containerd → OCI runtime)
- containerd/cri
- kubernetes-incubator/rktlet
- kubernetes-sigs/cri-o

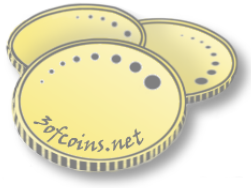


Kubernetes & CRI

Container Runtime Interface

Kubernetes' gRPC API for runtime plugins:

- dockershim (kubelet → dockershim → docker → containerd → OCI runtime)
- containerd/cri (calls containerd directly)
- kubernetes-incubator/rktlet
- kubernetes-sigs/cri-o

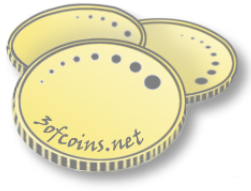


Kubernetes & CRI

Container Runtime Interface

Kubernetes' gRPC API for runtime plugins:

- dockershim (kubelet → dockershim → docker → containerd → OCI runtime)
- containerd/cri (calls containerd directly)
- kubernetes-incubator/rktlet (calls rocket)
- kubernetes-sigs/cri-o

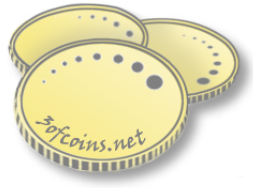


Kubernetes & CRI

Container Runtime Interface

Kubernetes' gRPC API for runtime plugins:

- dockershim (kubelet → dockershim → docker → containerd → OCI runtime)
- containerd/cri (calls containerd directly)
- kubernetes-incubator/rktlet (calls rocket)
- kubernetes-sigs/cri-o (Made for Kubernetes, manages everything and directly calls OCI runtime)



DIY Container Environment

- OCI Runtime (any)
- containers/storage
- projectatomic/skopeo / containers/image
- openSUSE/umoci
- containernetworking/cni + containernetworking/plugins
- projectatomic/buildah
- A bit of duct tape