Jetpack
A container runtime for FreeBSD

Maciej Pasternacki <maciej@3ofcoins.net>
BSDCan 2015
Outline

OS-level Virtualization: Not a New Tech

The Container Mindset

Docker & Rocket

App Container Specification

Jetpack
OS-level Virtualization

Single host kernel

⇓

Multiple guest userspaces
## Hypervisor-type Virtualisation

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Host Hypervisor</th>
<th>Guest OS</th>
<th>Userspace</th>
<th>Guest OS</th>
<th>Userspace</th>
<th>Guest OS</th>
<th>Userspace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OS-level Virtualisation

- Hardware
  - Host OS
    - Host Userspace
      - Guest Userspace
      - Guest Userspace
      - Guest Userspace
OS-level Virtualization
versus hypervisor

- Less isolation
- Guest & host OS must be the same\(^1\)
- Lower overhead
- Adjustable isolation level
- Resource sharing is possible

\(^1\)or binary-compatible: Solaris branded zones, FreeBSD, Linuxulator
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

1998  FreeBSD Jail
2001  Linux–VServer, Virtuozzo
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
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
Container Mindset

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

Ubuntu LTS Image (RO)
Layered Storage

Ubuntu LTS Image (RO)

Ruby-2.1.5

Redis server
Layered Storage

- Ubuntu LTS Image (RO)
- Ruby-2.1.5 Redis server
- Rails app
Layered Storage

Ubuntu LTS → Image (RO)

Ruby-2.1.5 → Redis server

Rails app

Bob's App → Container (R/W, volatile)
Layered Storage
Layered Storage

- Ubuntu LTS Image (RO)
- Ruby-2.1.5
- Redis server
- Rails app
- Bob's App Container (R/W, volatile)
- User Uploads
- Redis B Persistence
- Volume (persistent)
Layered Storage

- Ubuntu LTS Image (RO)
- Ruby-2.1.5
- Redis server
- Rails app
- Bob's App (Container (R/W, volatile))
- User Uploads
- Redis B
- Persistence (Volume (persistent))
Layered Storage

- Ubuntu LTS Image (RO)
- Ruby-2.1.5
- Redis server
- Rails app
- Bob's App
  - Container (R/W, volatile)
  - User
  - Uploads
  - Persistence
- Alice's App
  - User
  - Uploads
  - Redis A
  - Persistence
  - Persistence
- Bob's App
  - User
  - Uploads
  - Redis B
  - Persistence
  - Persistence

Redis Persistence Volume (persistent)
Layered Storage

Ubuntu LTS Image (RO)

Ruby-2.1.5
Redis server
Rails app
Sinatra app

Alice's App
Redis A
User Uploads
Persistence

Bob's App
User Uploads
Redis B
Persistence

Claire's App
Redis C
Persistence
Layered Storage

- Ubuntu LTS Image (RO)
- Ruby-2.1.5
- Redis server
- Rails app
- Bob's App
- Container (R/W, volatile)
- User Uploads
- Redis B
- Persistence
- Volume (persistent)
Layered Storage

Ubuntu LTS → Image (RO)

Ruby-2.1.5 → Redis server

Rails app

Bob's App → Bob's App 2

User Uploads

Redis B → Persistence

Container (R/W, volatile)
Layered Storage

Ubuntu LTS

Ruby-2.1.5

Redis server

Rails app

Sinatra app

Alice's App

Bob's App 2

Bob's App

Claire's App

User Uploads

Redis A

User Uploads

Reload A

User Uploads

Redis B

User Uploads

Redis C

Persistence

Volume (persistent)

Container (R/W, volatile)
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

- Well-defined images can be shared & reused across applications
- Containers can be meaningfully managed & monitored by host

Management overhead of a single service
Docker

- First free container runtime
- Defined the container paradigm
- Extremely fast & wide adoption
- Implementation-driven

https://www.docker.com/
First free container runtime
⇒ and the **only** one, for a long time

Defined the container paradigm

Extremely fast & wide adoption

Implementation-driven

https://www.docker.com/
First free container runtime
⇒ and the **only** one, for a long time

Defined the container paradigm
⇒ prototyped it

Extremely fast & wide adoption

Implementation-driven

https://www.docker.com/
Docker

First free container runtime
⇒ and the only one, for a long time
Defined the container paradigm
⇒ prototyped it
Extremely fast & wide adoption
⇒ locked into early design decisions
Implementation-driven

https://www.docker.com/
First free container runtime
⇒ and the only one, for a long time

Defined the container paradigm
⇒ prototyped it

Extremely fast & wide adoption
⇒ locked into early design decisions

Implementation-driven
⇒ Implementation-defined

https://www.docker.com/
The management question, therefore, is not whether to build a pilot system and throw it away. You will do that. [...] Hence plan to throw one away; you will, anyhow.

— Fred Brooks, The Mythical Man–Month
CoreOS Rocket

- First implementation of the *appc* specification
- Designed for “composability, security, and speed”
- Breaks Docker monotoculture
- Linux-only

https://github.com/coreos/rkt
App Container Specification

AKA appc

Composable
Secure
Decentralized
Open

appc/spec
App Container Image (ACI)

- A compressed `tar` file containing:
  - manifest JSON file
  - `rootfs/` directory

- Identified by SHA–512 checksum (before compression)

- Addressed by *name* and a set of *labels*

https://github.com/appc/spec/blob/master/spec/aci.md
ACI Manifest

```json
{
    "acKind": "ImageManifest",
    "acVersion": "0.5.2",
    "name": "demo/bsdcan2015/redis",
    "labels": [
        {
            "name": "version",
            "value": "3.0.2"
        },
        {
            "name": "os",
            "value": "freebsd"
        },
        {
            "name": "arch",
            "value": "amd64"
        }
    ],
    "app": {
        "exec": [
            "/usr/local/bin/redis-server",
            "/usr/local/etc/redis.conf"
        ],
        "user": "redis",
        "group": "redis",
        "mountPoints": [
            {
                "name": "redis-datadir",
                "path": "/var/db/redis"
            }
        ],
        "ports": [
            {
                "name": "redis",
                "protocol": "tcp",
                "port": 6379
            }
        ],
        "annotations": [
            {
                "name": "timestamp",
                "value": "2015-06-12T19:41:25-04:00"
            }
        ],
        "dependencies": [
            {
                "app": "3ofcoins.net/freebsd-base",
                "imageID": "sha512-a9c9...91d0",
                "labels": [
                    {
                        "name": "version",
                        "value": "10.1.12"
                    },
                    {
                        "name": "os",
                        "value": "freebsd"
                    },
                    {
                        "name": "arch",
                        "value": "amd64"
                    }
                ]
            }
        ]
    }
}
```
App Container Image Discovery

From ACI name & labels to:

- ACI URL
- ACI Signature URL
- Public Key URL

https://github.com/appc/spec/blob/master/spec/discovery.md
App Container Image Discovery

From ACI name & labels to:

- ACI URL
- ACI Signature URL
- Public Key URL

name 3ofcoins.net/freebsd-base
labels version=10.1.12
        os=freebsd
        arch=amd64

https://github.com/appc/spec/blob/master/spec/discovery.md
First, try to just use name as base URL:

- https://{name}-{version}-{os}-{arch}.aci
- https://{name}-{version}-{os}-{arch}.aci.asc
- No public key discovery

https://github.com/appc/spec/blob/master/spec/discovery.md
App Container Image Discovery

Simple Discovery

First, try to just use name as base URL:

- https://{name}-{version}-{os}-{arch}.aci
- https://{name}-{version}-{os}-{arch}.aci.asc
- No public key discovery

https://3ofcoins.net/freebsd-base-10.1.12-freebsd-amd64.aci

https://github.com/appc/spec/blob/master/spec/discovery.md
Go to https://{name}?ac-discovery=1

https://github.com/appc/spec/blob/master/spec/discovery.md
Go to https://{name}?ac-discovery=1

Look for:

```
<meta name="ac-discovery" content="prefix-match url-tmpl">
<meta name="ac-discovery-pubkeys" content="prefix-match url">
```

https://github.com/appc/spec/blob/master/spec/discovery.md
Go to https://{name}?ac-discovery=1

Look for:

```html
<meta name="ac-discovery" content="prefix-match url-tmpl">
<meta name="ac-discovery-pubkeys" content="prefix-match url">
```

If that fails, strip last component off name and try again.

https://github.com/apmc/spec/blob/master/spec/discovery.md
Go to https://{name}?ac-discovery=1

Look for:

<meta name="ac-discovery" content="prefix-match url-tmpl">
<meta name="ac-discovery-pubkeys" content="prefix-match url">

If that fails, strip last component off name and try again.

Rinse. Repeat.

https://github.com/appc/spec/blob/master/spec/discovery.md
App Container Image Discovery

Meta Discovery

https://3ofcoins.net/freebsd-base?ac-discovery=1
App Container Image Discovery

Meta Discovery

https://3ofcoins.net/freebsd-base?ac-discovery=1 ⇒ 404
App Container Image Discovery

Meta Discovery

https://3ofcoins.net/freebsd-base?ac-discovery=1 ⇒ 404
https://3ofcoins.net?ac-discovery=1
App Container Image Discovery

Meta Discovery

https://3ofcoins.net/freebsd-base?ac-discovery=1 ⇒404
https://3ofcoins.net?ac-discovery=1

<meta name="ac-discovery" content="3ofcoins.net"
  ➔ https://3ofcoins-aci.s3.eu-central-1.amazonaws.com/{name}-
  ➔ {version}-{os}-{arch}.{ext}>

<meta name="ac-discovery-pubkeys" content="3ofcoins.net"
  ➔ https://3ofcoins-aci.s3.eu-central-1.amazonaws.com/aci-
  ➔ pubkeys.asc">
App Container Image Discovery

Meta Discovery

https://3ofcoins.net/freebsd-base?ac-discovery=1 ⇒ 404
https://3ofcoins.net?ac-discovery=1

<meta name="ac-discovery" content="3ofcoins.net
  ➔ https://3ofcoins-aci.s3.eu-central-1.amazonaws.com/{name}-
  ➔ {version}-{os}-{arch}.{ext}"

<meta name="ac-discovery-pubkeys" content="3ofcoins.net
  ➔ https://3ofcoins-aci.s3.eu-central-1.amazonaws.com/aci-
  ➔ pubkeys.asc"

https://3ofcoins-aci.s3.eu-central-1.amazonaws.com/...
  .../3ofcoins.net/freebsd-base-10.1.12-freebsd-amd64.aci
  .../3ofcoins.net/freebsd-base-10.1.12-freebsd-amd64.aci.asc
  .../aci-pubkeys.asc
appc Pods

A list of apps that will be launched together inside a shared execution context

- Shared PID space, network, IPC, hostname
- Separate filesystem root for each app
- Shared, persistent volumes
- Isolators

https://github.com/appc/spec/blob/master/spec/pods.md
Pod Manifest

template

```json
{
  "acVersion": "0.5.2",
  "acKind": "PodManifest",
  "apps": [
    {
      "name": "redis",
      "image": {
        "name": "demo/bsdcan2015/redis"
      },
      "mounts": [{
        "volume": "redis-datadir",
        "mountPoint": "redis-datadir"
      }]
    },
    {
      "name": "tipboard",
      "image": {
        "name": "demo/bsdcan2015/tipboard"
      },
      "mounts": [{
        "volume": "tipboard",
        "mountPoint": "tipboard"
      }]
    }
  ],
  "volumes": [
    {
      "name": "tipboard",
      "kind": "host",
      "readOnly": true,
      "source": "/home/japhy/Documents/20150607-bsdcan2015-jetpack/demo/data"
    }
  ]
}
```
Pod Manifest

reified

```json
{
    "acVersion": "0.5.2",
    "acKind": "PodManifest",
    "apps": [
        {
            "name": "redis",
            "image": {
                "name": "demo/bsdcan2015/redis",
                "id": "sha512-a9c9...91d0"
            },
            "mounts": [
                {
                    "volume": "redis-datadir",
                    "mountPoint": "redis-datadir"
                }
            ]
        },
        {
            "name": "tipboard",
            "image": {
                "name": "demo/bsdcan2015/tipboard",
                "id": "sha512-8a6d...f0fb"
            },
            "mounts": [
                {
                    "volume": "tipboard",
                    "mountPoint": "tipboard"
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "redis-datadir",
            "kind": "empty"
        },
        {
            "name": "tipboard",
            "kind": "host",
            "readOnly": true,
            "source": "/home/japhy/Documents/20150607-bsdcan2015-jetpack/demo/data"
        }
    ],
    "annotations": [
        {
            "name": "ip-address",
            "value": "172.23.0.2"
        }
    ]
}
```
Appc Executor

Executor Perspective

- Assigns pod UUIDs
- Renders apps' filesystems
- Sets up volumes
- Configures network
- Collects logs from stdout & stderr

https://github.com/appc/spec/blob/master/spec/ace.md
appc Executor
App Perspective

- Environment variables, UID, GID, working directory as per image/pod manifest
- Resource isolation
- Access limits
- Metadata service

https://github.com/appc/spec/blob/master/spec/ace.md
$AC_METADATA_URL/acMetadata/v1/…

- /pod/annotations/NAME
- /pod/manifest (fully reified)
- /pod/UUID
- /apps/$AC_APP_NAME/…
  - /annotations/NAME
  - /image/manifest
  - /image/id

https://github.com/appc/spec/blob/master/spec/ace.md
appc Metadata Service

$AC_METADATA_URL/acMetadata/v1/...

- /pod/hmac/sign — POST to have ACE sign any data as this pod
- /pod/hmac/verify — verify another pod's (or own) signature on data

https://github.com/appc/spec/blob/master/spec/ace.md
Jetpack

App Container Specification implementation for FreeBSD

3ofcoins/jetpack

(not production ready)
Jetpack

- Written in Go
- Jails for process isolation & lockdown
- ZFS for layered storage
- Runs Linux images (as allowed by FreeBSD's emulation)
- Breaks Linux monoculture (hopefully)
- Half year old this Monday

https://github.com/3ofcoins/jetpack/
Jetpack: ZFS Storage

- Each image's rootfs is a ZFS snapshot
- Dependent image's rootfs is cloned from parent, then updated
- Pod app's rootfs is cloned from image
- Each empty volume is a ZFS dataset

https://github.com/3ofcoins/jetpack/
Jetpack: Runtime

- Jail for pod isolation
- Each app has additional `chroot(2)` inside jail's fs root
- Volumes are `nullfs(5)` mounts

https://github.com/3ofcoins/jetpack/
Jetpack: Image Building

jetpack image IMG build -dir=. CMD ARGS...

- Clone new pod from IMG
- Copy build dir to a new directory
- Run build command CMD... in the build dir
- Copy new manifest from build dir
- Use pod's rootfs (without build dir) as new image's

https://github.com/3ofcoins/jetpack/blob/master/IMAGES.md
Jetpack: Image Building

```
.MAKEFLAGS: -I${HOME}/Src/github.com/3ofcoins/jetpack/share

PARENT_IMAGE = 3ofcoins.net/freebsd-base
PKG_INSTALL = python27 py27-virtualenv libyaml

basedir=/opt/tipboard
projdir=${basedir}/home/.tipboard

build:
    virtualenv ${basedir}
    ${basedir}/bin/pip install tipboard
    install -m 0755 pre-start.sh ${basedir}/bin/pre-start.sh
    install -d ${basedir}/data ${projdir}
    install settings-local.py ${projdir}/settings-local.py
    ln -s /dev/null ${basedir}/home/tipboard.log
    install -m 0755 tipboard.sh /usr/local/bin/tipboard

manifest.json:
    ./manifest.json.sh > @

    .include "jetpack.image.mk"
```

https://github.com/3ofcoins/jetpack/blob/master/IMAGES.md
#!/bin/sh
set -e

version="$(tipboard --version)"
version="${version#Tipboard }"

cat <<EOF
{
 "name": "demo/bsdcan2015/tipboard",
 "labels": [{ "name": "version", "value": "${version}" }],
 "app": {
 "exec": ["/usr/local/bin/tipboard", "runserver", "0.0.0.0", "7272"],
 "eventHandlers": [
  { "name": "pre-start", "exec": [ "/opt/tipboard/bin/pre-start.sh" ],
 "user": "www",
 "group": "www",
 "ports": [{ "name": "http", "protocol": "tcp", "port": 7272 }],
 "mountPoints": [{ "name": "tipboard", "path": "/opt/tipboard/data" }
  
  ]

  }]

EOF

https://github.com/3ofcoins/jetpack/blob/master/IMAGES.md
import os, os.path, urllib

execfile(os.path.expanduser("~/tipboard/settings.py"))

AC_MDS_BASE = os.getenv('AC_METADATA_URL') + '/acMetadata/v1'
REDIS_HOST = urllib.urlopen(MDS_BASE + '/pod/annotations/ip-address').read()
REDIS_PORT = 6379

https://github.com/3ofcoins/jetpack/blob/master/IMAGES.md
Jetpack: TODO

- Isolators
- *pf* anchor management
- Better UI: commands, output
- Boring stuff: docs, acceptance tests
- Native multi-app pod support
- Logging

https://github.com/3ofcoins/jetpack/
Demo time!
Questions?

3ofcoins/jetpack