Underpinnings: Intro to Container Hosts

Container Linux and Atomic Host

Mark Lamourine <markllama@everywhere>

usenix LISA17

October 29–November 3, 2017 | San Francisco, CA
www.usenix.org/lisa17 #lisa17

Slides and Code Samples

Slides and Code Samples:
https://github.com/markllama/lisa17

Mandatory Introduction

- My first program was BASIC on paper tape
- I play classical guitar badly
- My daughter and I trade really bad puns
- Emacs AND vi
- I can juggle but not ride a unicycle
- I’ve thrown over Perl 5 for Python and Go
- I still miss Solaris 8
- Still a Sysadmin at heart - Been QA, Lab Mgr, Dev, Ops, Mini-Golf Attendant
- Tinnitus from DC fan noise - wear your hearing protection
- I take naps at lunchtime - the Spanish are onto something

What this presentation is not:

- A Kubernetes Tutorial (That's next door)
- A Docker Tutorial
- An OpenShift or Tectonic Tutorial
- A Comprehensive Course on Anything

What you can expect to leave with:

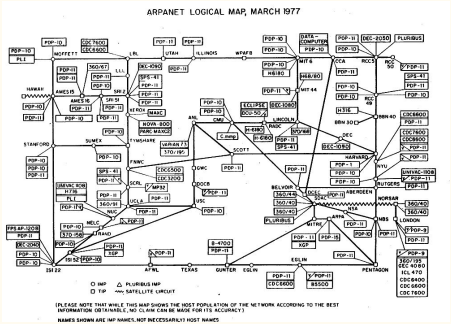
- Understanding why Container Hosts Matter
  - What they're good for
  - What they're not
- How to get started
- Basic Clustering
- Basic Networking
- Basic Admin Tools
- System Containers and Services
- Lots of resources to dig deeper

SECTION 1 - Container & “Conventional” Hosts

- Conventional Hosts - if it ain’t broke.. Is it broke?
- Container Hosts -
  - What is it?
  - What do I get?
  - How does it help?
- Deploying a Container Host
  - CoreOS
  - Atomic host

Conventional Hosts: Prehistory

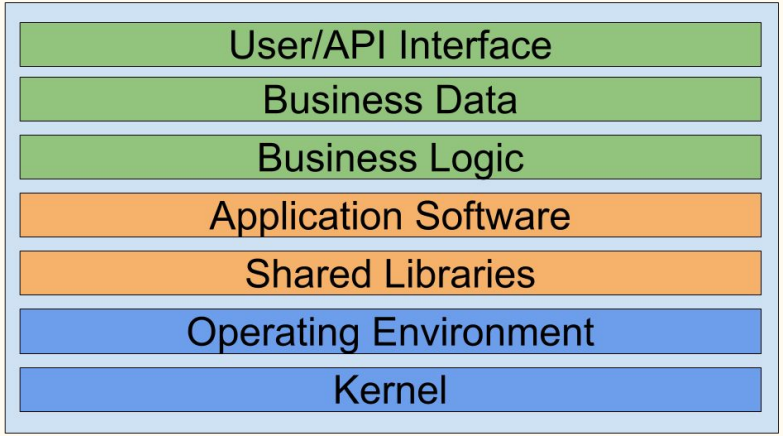
- Up from Monoliths! VMS, OS360...
- gcc-3 on 8mm tape, build 3 times to validate
- `/usr/local`
- `tar -x  ; cd <foo>`
- `configure ; make ; make install`
- Shared libraries are EVIL - DLL Hell



GNU By Aurelio A. Heckert <aurium@gmail.com> - gnu.org, CC BY-SA 2.0, https://en.wikipedia.org/w/index.php?curid=33285325
Map By ARPANET - The Computer History Museum ([1]), en:File:Arpnet-map-march-1977.png, Public Domain, https://commons.wikimedia.org/w/index.php?curid=9990864
Photo By David Monniaux (Own work) CC-BY-SA-3.0

Conventional Hosts: Layers

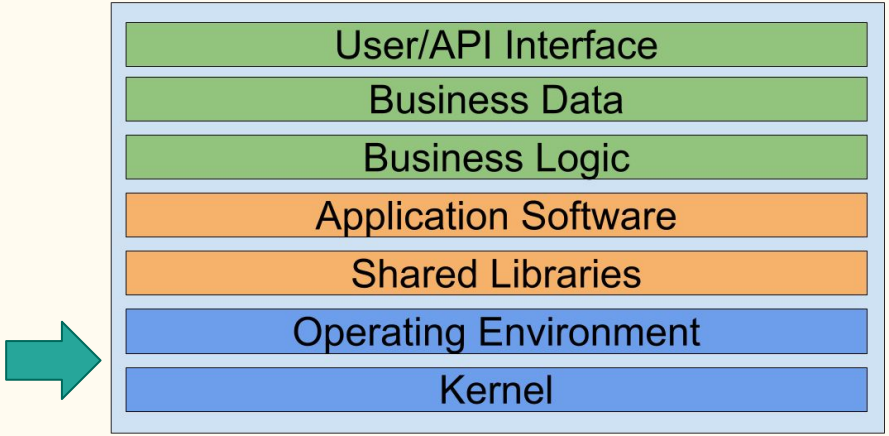
- OS layers map loosely to OSI/ISO layers
- Different organizations are responsible for different layers
- Layers impose dependencies, up and down



Conventional Hosts: Layers

Base: Kernel and OS Environment

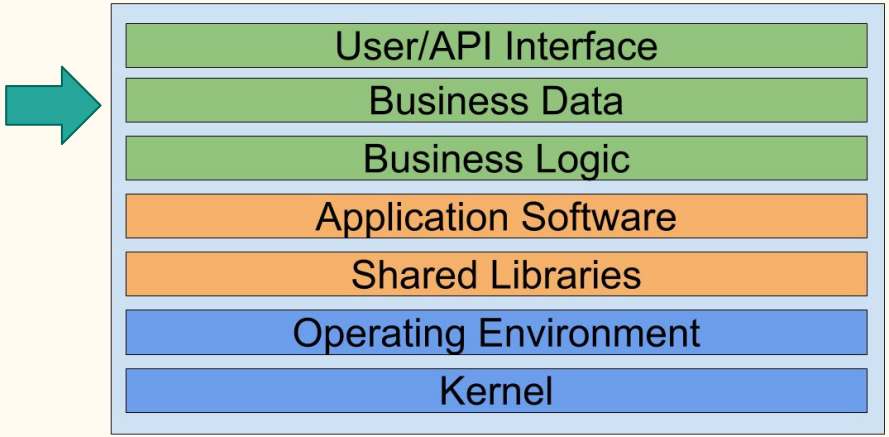
- Managed by Ops/Sysadmin
- Package Management
- Configuration Management
- Stable - update for security



Conventional Hosts: Layers

Top: Applications, Logic and Data

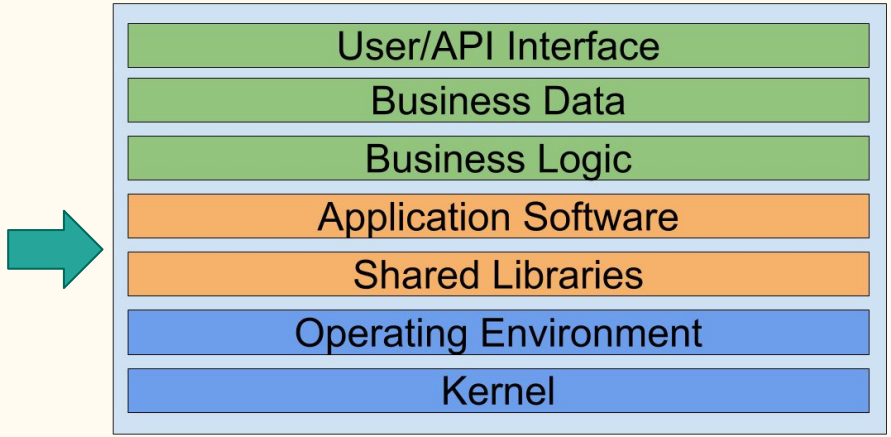
- Managed by App Devs and App Ops
- Ad hoc delivery and management
- Configuration Management?
- Dynamic update schedule



Conventional Hosts: Layers - Ideal

Middle: Languages and Libraries - Ideal

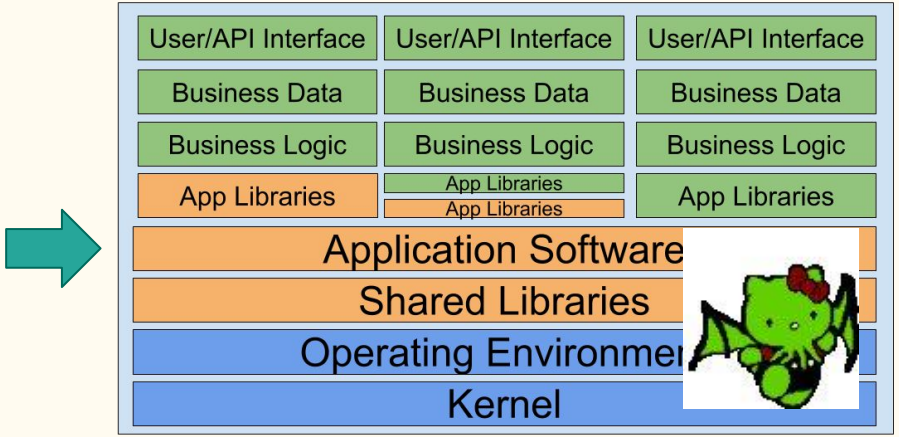
- Controlled by Sysadmin/Ops
- Package Management
- Depends on OS Environment
- Provides App Environment
- Tightly coupled in both directions



Conventional Hosts: Layers - Reality

Middle: Languages and Libraries

- Rigid base
- Apps rebel and include base variants
- Multiple software management systems



Characteristics of Conventional Hosts

- Tight coupling between OS and App
  - OS Ops owns app binaries and libraries - Why does OS Ops care which Apache?
- Host and App Ops vie for control
  - Which version of Apache/PHP/Python/Ruby/Rails/Java?
  - Which version of $LIBRARY?
- Script Libraries managed in user space (gem, pip, npm)
  - Adds another layer of management/coupling
  - Discourages interface stability
- Alternatives in Ops Space: Software Collections (SCL)
- Security - Large Attack Surface
  - App code in root fs space
- Updates are Shoot and Pray - no reliable rollback

A History of Container Hosts

Pre-History

- Embedded Systems
  - pSOS, Windriver, Q?
- Compact Linux
  - BusyBox
  - Android
  - ChromeOS
- Modern Container Hosts
  - Container Linux - Oct 2013 - https://www.wired.com/2013/08/coreos-the-new-linux/
  - Project Atomic - April 2014
  - RancherOS - 2014

Characteristics of Container Hosts

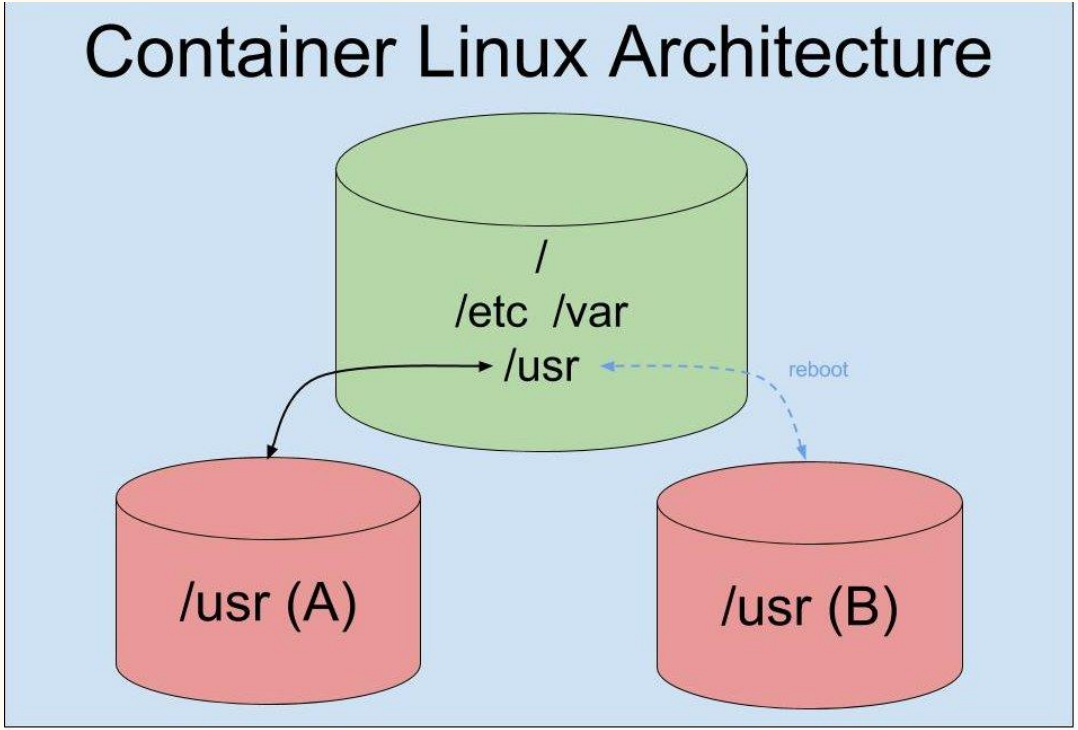
- Minimal Base OS
- Atomic Update
- Reliable Rollback
- Read Only (/usr)
- Minimal Configuration
- No User Access
- Integrated Configuration Cluster
- Integrated SDN
- Integrated Container Runtime

Architecture - Container Linux

- Based on ChromeOS
  - Designed to be embedded or “frozen”
- Gentoo style build process
  - Build from source - clean build every time
    https://coreos.com/os/docs/latest/sdk-modifying-coreos.html
- A/B switching `/usr` partitions
  https://coreos.com/os/docs/latest/sdk-disk-partitions.html
- Image overwrites inactive `/usr` partition
  - Download a single file as a unit

Architecture - Container Linux

- `/, /etc, /var` (rw)
- `/usr` (a and b) (ro)
- `/usr` selected at boot
- Updates detected by `update-engine`
- Cluster lock by `locksmithd`
  Avoids simultaneous updates
- Reboot A -> B after update
- Rollback B -> A if needed

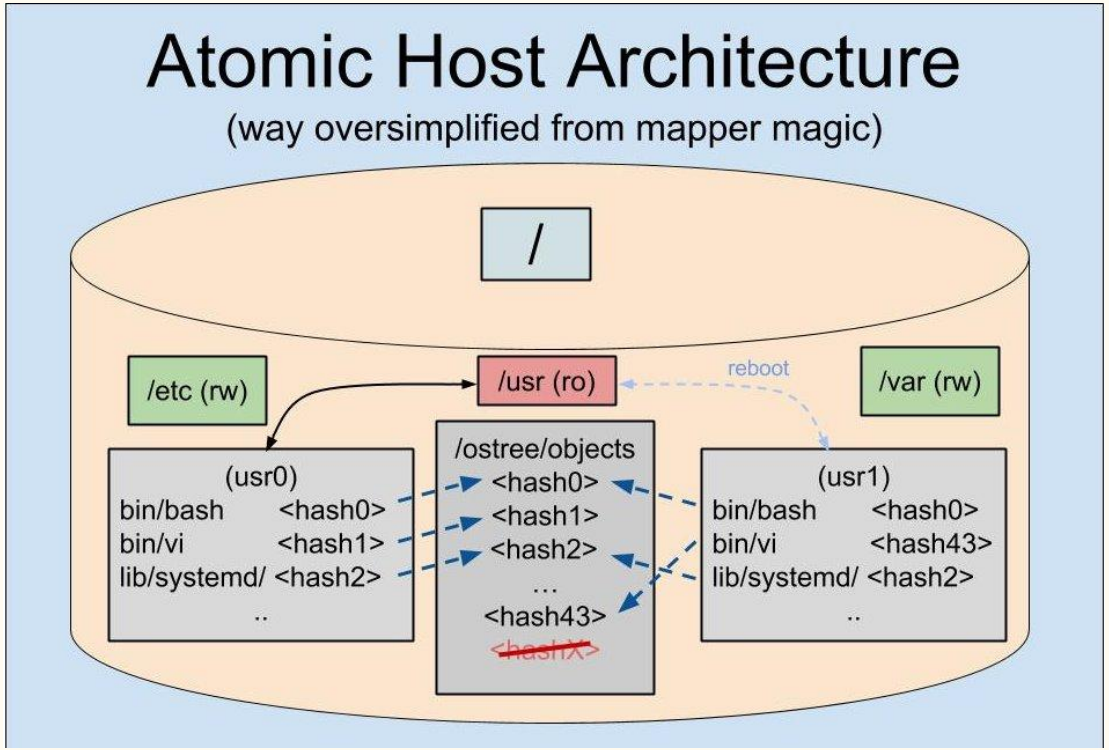


Architecture - Atomic Host

- Based on RPM - RHEL, CentOS or Fedora
- Build using rpm-ostree
  https://rpm-ostree.readthedocs.io/en/latest/
- Updates using rpm-ostree
- Boots using modified grub2 bootloader (upstreamed)
- Uses lvm, udev and mapper to provide and mount all needed partitions
- Downloads only tree “diffs”
- Files removed when refcount == 0

Architecture - Atomic Host

- Boots with grub2
- Uses udev and mapper
- Rpm-ostree maintains hard link trees to hashed files
- Common files shared
- Files removed when refcount = 0
- `atomic host update` compares and merges trees



Running Container Linux (Vagrant)

CoreOS Container Linux: https://coreos.com/os/docs/latest/booting-on-vagrant.html
Vagrant Repository: https://github.com/markllama/coreos-vagrant

```
git clone https://github.com/coreos/coreos-vagrant
cd coreos-vagrant
Git checkout dc8dc0efc3629940b9c66ae2afc96057f3287713
cp config.rb.sample config.rb
export NUM_INSTANCES=3
sed -e "s/<token>/$(curl -s https://discovery.etcd.io/new?size=3 | cut -d/ -f4)/" \
  user-data.sample > user-data
vagrant up
…
vagrant ssh core-01
```

Running Atomic Host (Vagrant)

Project Atomic downloads page: http://www.projectatomic.io/download/

```
git clone  https://github.com/markllama/lisa17
cd lisa17/atomic-vagrant
export VAGRANT_DEFAULT_PROVIDER=virtualbox # unless you prefer libvirt
vagrant up
...
vagrant ssh atomic-01
```

LAB 1 - Vagrant Install

`https://github.com/markllama/lisa17/blob/master/ATOMIC.adoc`

- Install Atomic Host in Vagrant
- Configure etcd
- Configure flannel
- Confirm networking

Alternate - CoreOS:

https://github.com/markllama/lisa17/blob/master/COREOS.adoc

CoreOS has been making changes to their vagrant setup that make a demo obsolete

Section 2 - Clustering and Networking

- Cluster Configuration - etcd
  - Host identity
  - Cluster Membership
  - Encryption and Access
- SDN Configuration - flannel
  - Cluster configuration - in etcd
  - Host configuration
    - Etcd location
    - NIC selection
    - IP masquerade

etcd: a distributed configuration database

- Source: https://github.com/coreos/etcd
- Hierarchical key-value store
  - Keys use file path syntax
- Values stored as strings (commonly JSON)
- Distributed data via raft protocol
- Accessible by http(s) on TCP/2379
- Clustering on TCP/2380
- Accessible by etcdctl client
- “watch” mechanism allows blocking/triggering on value change.

CoreOS etcd intro: https://coreos.com/etcd/
Source Code: https://github.com/coreos/etcd
etcd docs (v2): https://github.com/coreos/etcd/blob/master/Documentation/v2/README.md

etcd configuration: self identification

Report how to reach me.

- NAME - unique identifier within the cluster
- PEER_URLS - IP and ports to listen for cluster members
- CLIENT_URL - IP and ports to listen for clients

```
# [member]
ETCD_NAME={{ hostname }}
ETCD_DATA_DIR="/var/lib/etcd/default.etcd"
ETCD_LISTEN_PEER_URLS="http://{{ hosts[hostname].ipaddr }}:2380"
ETCD_LISTEN_CLIENT_URLS="http://127.0.0.1:2379,http://{{ hosts[hostname].ipaddr }}:2379"
```

etcd configuration: cluster membership

- PEER_URL - Name/Address must resolve and route for cluster members
- CLIENT_URL - Name/Address must resolve and route for clients
- INITIAL_CLUSTER - PEER_URLs for all members at start

```
# [cluster]
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://{{ hosts[hostname].ipaddr }}:2380"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_ADVERTISE_CLIENT_URLS="http://{{ hosts[hostname].ipaddr }}:2379"
ETCD_INITIAL_CLUSTER="{{- name }}=http://{{ hosts[name].ipaddr }}:2380"
#ETCD_DISCOVERY=”http://discovery.etcd.io/<token>”
#  new tokens at http://discovery.etcd.io/new?size=3”
```

etcd configuration: security

Cluster Member and Client encryption and access
In production, create and provide SSL certificates and keys for CA, peer and client connections

```
#[security]
#ETCD_CERT_FILE=""
#ETCD_KEY_FILE=""
#ETCD_CLIENT_CERT_AUTH="false"
#ETCD_TRUSTED_CA_FILE=""
#ETCD_AUTO_TLS="false"
#ETCD_PEER_CERT_FILE=""
#ETCD_PEER_KEY_FILE=""
#ETCD_PEER_CLIENT_CERT_AUTH="false"
#ETCD_PEER_TRUSTED_CA_FILE=""
#ETCD_PEER_AUTO_TLS="false"
```

etc cluster configuration: discovery

Clusters can also form by *discovery*. You can use the discovery server provided by CoreOS or run your own.

```
curl https://discovery.etcd.io/new?size=3 ; echo
https://discovery.etcd.io/6d5a3c07e7b19c21d41fcfa649df4fbb
```

Replace the INITIAL_CLUSTER variable with a DISCOVERY URL in `/etc/etcd/etcd.conf`

```
...
ETCD_DISCOVERY="https://discovery.etcd.io/6d5a3c07e7b19c21d41fcfa649df4fbb"
...
```

etcd management: etcdctl

- Etcd uses HTTP, and JSON for communications and control.
- Etcdctl simplifies operations.
- Use `member list` to confirm the cluster

```
etcdctl member list
25d6ce33763c5524: name=atomic-02 peerURLs=http://172.17.8.102:2380 clientURLs=http://172.17.8.102:2379 isLeader=false
6ae27f9fa2984b1d: name=atomic-01 peerURLs=http://172.17.8.101:2380 clientURLs=http://172.17.8.101:2379 isLeader=true
ff32f4b39b9c47bd: name=atomic-03 peerURLs=http://172.17.8.103:2380 clientURLs=http://172.17.8.103:2379 isLeader=false
```

etcd management: curl/wget

Use `--debug` to get `curl` equivalent commands

```
etcdctl --debug member list
start to sync cluster using endpoints(http://127.0.0.1:4001,http://127.0.0.1:2379)
cURL Command: curl -X GET http://127.0.0.1:4001/v2/members
cURL Command: curl -X GET http://127.0.0.1:2379/v2/members
got endpoints(http://172.17.8.102:2379,http://172.17.8.101:2379,http://172.17.8.103:2379) after sync
Cluster-Endpoints: http://172.17.8.102:2379, http://172.17.8.101:2379, http://172.17.8.103:2379
cURL Command: curl -X GET http://172.17.8.102:2379/v2/members
cURL Command: curl -X GET http://172.17.8.102:2379/v2/members/leader
25d6ce33763c5524: name=atomic-02 peerURLs=http://172.17.8.102:2380 clientURLs=http://172.17.8.102:2379
isLeader=false
6ae27f9fa2984b1d: name=atomic-01 peerURLs=http://172.17.8.101:2380 clientURLs=http://172.17.8.101:2379
isLeader=true
ff32f4b39b9c47bd: name=atomic-03 peerURLs=http://172.17.8.103:2380 clientURLs=http://172.17.8.103:2379
isLeader=false
```

etcd data operations - etcdctl set/get/watch

```
[vagrant@atomic-01 ~]$ etcdctl set /test/value '{"test": "value0"}'
{"test": "value0"}
```

```
[vagrant@atomic-03 ~]$ etcdctl get /test/value
{"test": "value0"}
[vagrant@atomic-03 ~]$ etcdctl watch /test/value # blocks
```

```
[vagrant@atomic-01 ~]$ etcdctl set /test/value '{"test": "value1"}'
{"test": "value1"}
```

```
[vagrant@atomic-03 ~]$ etcdctl watch /test/value # unblocks
{"test": "value1"}
```

Use `--debug` to get `curl` equivalent commands

flannel: container networking

- Flannel - a Software Defined Network (SDN) system
- Provides networking between containers
- Multiple undercarriages: - Open vSwitch, “host-gateway mode”...
- Can run on public or private net
- Encapsulates or tunnels traffic

Flannel does not provide dynamic networks for container network isolation
Orchestration systems are moving to Calico and OpenStack Kuryr (or other cloud provider container networking)

CoreOS Flannel Docs: https://coreos.com/flannel/docs/latest/
Source Code: https://github.com/coreos/flannel

flannel: etcd service configuration

- Define docker container network spaces
- Configuration stored in etcd
  - Contact info in `/etc/sysconfig/flanneld`
  - Default etcd endpoint: localhost
  - Default etcd prefix
    - Atomic: `/atomic.io/network`
    - CoreOS: `/flannel/network` -> `/coreos.com/network`

```
cat /etc/sysconfig/flanneld
# Flanneld configuration options
# etcd url location.  Point this to the server where etcd runs
FLANNEL_ETCD_ENDPOINTS="http://127.0.0.1:2379"
# etcd config key.  This is the configuration key that flannel queries
# For address range assignment
FLANNEL_ETCD_PREFIX="/atomic.io/network"
# Any additional options that you want to pass
#FLANNEL_OPTIONS=""
```

flannel: service management

The flannel shared configuration is stored in etcd
Path is arbitrary, depends on deployment

- Atomic: `/atomic.io/network/config`
- CoreOS: `/flannel/network/config` or `/coreos.io/network/config`

```
etcdctl set /atomic.io/network/config {'
      "Network": "172.24.0.0/16",
      "SubnetLen": 24,
      "Backend": {
            "Type": "vxlan",
      }
}'
sudo systemctl start flanneld
sudo systemctl enable flanneld
```

flannel: docker configuration

- Integration is automatic:
  - Atomic: `/run/flannel/docker`
  - CoreOS: `/run/flannel/flannel_docker_opts.env`
- Docker reads on startup
  - `systemctl restart docker`

```
[atomic-01]# cat /run/flannel/docker
DOCKER_OPT_BIP="--bip=172.24.32.1/24"
DOCKER_OPT_IPMASQ="--ip-masq=false"
DOCKER_OPT_MTU="--mtu=1450"
DOCKER_NETWORK_OPTIONS=" --bip=172.24.32.1/24 --ip-masq=false --mtu=1450"
```

```
core-01 ~ $ cat /run/flannel/flannel_docker_opts.env
DOCKER_OPT_BIP="--bip=10.1.60.1/24"
DOCKER_OPT_IPMASQ="--ip-masq=false"
DOCKER_OPT_MTU="--mtu=1472"
```

BREAK

SESSION 3

- Review - what does “container” mean?
- Container Host Management
- Deploying
- Update and Rollback
- Toolbox Containers
- System Containers

Diversion: What is a Container?

- Container image
  - “Software package”
  - Payload file tree - code and data
  - Metadata - runtime information
- Container Instance
  - A process (with blinders on) - usually restricted view
  - Cgroups and namespaces define the process view
- Container Runtime
  - Can manage retrieval
  - Manages local image cache
  - Initializes instances from images
  - Manages instances

Container Review or “Get back in the box!”

What is a container?

First, what is it **not**?

What Containers Are Not

A Linux container is not:

- Just chroot/jails

What Containers Are Not

A Linux container is not:

- Just chroot/jails
- Just Solaris Containers

What Containers Are Not

A Linux container is not:

- Just chroot/jails
- Just Solaris Containers
- Just LXC

What Containers Are Not

A Linux container is not:

- Just chroot/jails
- Just Solaris Containers
- Just LXC
- Just another packaging format

What Containers Are Not

A Linux container is not:

- Just chroot/jails
- Just Solaris Containers
- Just LXC
- Just another packaging format
- VM Lite™

What Containers Are Not

A Linux container is not:

- Just chroot/jails
- Just Solaris Containers
- Just LXC
- Just another packaging format
- VM Lite™
- Docker™

OK, so what **is** it?

When people say “container” they often confuse at least three things:

- Software execution - Container Instance
- Software package - Container Image
- Software management - Container Runtime

Also -

- Software distribution - Container Image Repository

OK, so what **is** it?

A *container instance* is:

- A process[1]

1: and it's child processes

OK, so what **is** it?

A *container instance* is:

- A process[1] *with blinders* (or VR glasses)
  - Kernel Namespaces

1: and it's child processes

OK, so what **is** it?

A *container instance* is:

- A process[1] *with blinders* (or VR glasses)
  - Kernel Namespaces
    - Process

1: and it's child processes

OK, so what **is** it?

A *container instance* is:

- A process[1] *with blinders* (or VR glasses)
  - Kernel Namespaces
    - Process
    - Filesystem

1: and it's child processes

OK, so what **is** it?

A *container instance* is:

- A process[1] *with blinders* (or VR glasses)
  - Kernel Namespaces
    - Process
    - Filesystem
    - Network

1: and it's child processes

OK, so what **is** it?

A *container instance* is:

- A process[1] *with blinders* (or VR glasses)
  - Kernel Namespaces
    - Process
    - Filesystem
    - Network
    - User

1: and it's child processes

OK, so what **is** it?

A *container instance* is:

- A process[1] *with blinders* (or VR glasses)
  - Kernel Namespaces
    - Process
    - Filesystem
    - Network
    - User
  - Capabilities - per-thread operation permissions - (From VMS and Windows)

1: and it's child processes

OK, so what **is** it?

A *container instance* is:

- A process[1] *with blinders* (or VR glasses)
  - Kernel Namespaces
    - Process
    - Filesystem
    - Network
    - User
  - Capabilities - per-thread operation permissions - (From VMS and Windows)
  - Selinux

1: and it's child processes

OK, so what **is** it?

A *container image* is:
One (or more) archive files containing:

OCI Image Spec v 1.0 - https://github.com/opencontainers/image-spec/releases/tag/v1.0.0

OK, so what **is** it?

A *container image* is:
One (or more) archive files containing:

- Runtime payload
  - Base Binaries
  - Scripting language/libraries
  - Code
  - Startup (systemd?)

OCI Image Spec v 1.0 - https://github.com/opencontainers/image-spec/releases/tag/v1.0.0

OK, so what **is** it?

A *container image* is:
One (or more) archive files containing:

- Runtime payload
  - Base Binaries
  - Scripting language/libraries
  - Code
  - Startup (systemd?)
- Metadata
  - Identifiers
  - Runtime
    - Environment
    - Execution
  - Security (hash - content integrity)
  - Security (signature - source verification)

OCI Image Spec v 1.0 - https://github.com/opencontainers/image-spec/releases/tag/v1.0.0

OK, so what **is** it?

A *container runtime* is:

Well, it's more what it does.

- Initializes namespaces
- Unpacks image layers
  - overlayfs
- Mounts remote file services (ceph, gluster, cloud storage)
- Sets or removes Kernel Capabilities
- Initializes container root/init process
- Exec's container command

OCI Runtime Spec v 1.0 - https://github.com/opencontainers/runtime-spec/releases/tag/v1.0.0

OK, so what **is** it? (2)

A *container runtime* is:

Well, it's more what it does.

- Provides admin access
  eg. `docker exec -it <container> /bin/bash`
- Provides status
  eg. `docker inspect <container>`
- Can provide boot-time control
  `dockerd`, (though systemd removes dependency loops)

OCI Runtime Spec v 1.0 - https://github.com/opencontainers/runtime-spec/releases/tag/v1.0.0

Container Host Management - limited tools

Container hosts come with minimal tools installed

- Boot management
- Network Management (minimal)
- Process Management (start system services and containers)

Minimum requirements:

- Identity (hostname)
- Network (Primary Interface, IP Address/CIDR, gateway route)
- IP nameservice (`/etc/resolv.conf`)
- Time sync (`chrony` or `ntpd`) extra?
- Access/Authentication (root or admin SSH key, IdM) extra?

So how do I DO stuff?

Any tools or services that are not embedded must be installed as containers[1]

But containers can't *see* the host, they have *blinders on*!?

- containers are just processes in namespaces (mostly)
- But the system processes are also in a namespace. The system namespaces.
- You can run containers *in the system namespaces*.

You're not letting the container out, you're letting the system in.

1) or rpms in ostree space. See `atomic install --storage=ostree`

Special Containers: “super-privileged” containers

- Run containers in system namespaces
- Doesn’t let the container out, let’s the system “leak” in.
- With atomic cli, just `atomic run --spc` …
- Comparable to CoreOS toolbox
- Generally single run or shell interaction

```
docker run -it --name fedora-tools --privileged                \
  --ipc=host --net=host --pid=host                             \
  -e HOST=/host -e NAME=fedora-tools -e IMAGE=fedora/tools     \
  -v /run:/run -v /var/log:/var/log                            \
  -v /etc/localtime:/etc/localtime -v /:/host fedora/tools [tool name]

atomic run --spc fedora/tools [tool name]
```

- https://developers.redhat.com/blog/2014/11/06/introducing-a-super-privileged-container-concept/
- https://www.projectatomic.io/blog/2015/09/using-a-spc-to-troubleshoot-containers/
- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html/managing_containers/running_super_privileged_containers

Special Containers: System Containers

Run system services as containers:

- Startup with systemd - No dockerd dependency
- Persistent across reboots
- `atomic install --system [image]`
- Uses container metadata to define startup controls
- Basis of OCF and the OCI spec for runtime environment

http://www.projectatomic.io/blog/2016/09/intro-to-system-containers/
https://github.com/projectatomic/atomic-system-containers
https://www.slideshare.net/GiuseppeScrivano/atomic-system-containers
https://hub.docker.com/u/modularitycontainers/
https://github.com/container-images

Cockpit - System Container Example

A web UI system management tool.
Listens on port 9090 by default (Atomic vagrant forwards to 9091-3)
Can offer remote control and clustered monitoring (some assembly required)

```
[vagrant@atomic-01 ~]$ sudo atomic install cockpit/ws
[vagrant@atomic-01 ~]$ sudo atomic run cockpit/ws
```

http://cockpit-project.org/guide/latest/cockpit-ws.8.html

Fedora Modularity/Boltron “True” Container Host

- Modularity and Boltron are fairly new
- Focused from the start on containers for every day use (!?)
- All non boot-essential services in “modules”
  Tuned containers for *local* use
- Containerized desktops?
- Existing modules in Dockerhub to try
- Continuous OS update
- Decouple OS and app update schedules

https://docs.pagure.org/modularity/
https://docs.pagure.org/modularity/boltron/
https://hub.docker.com/u/modularitycontainers/

CLI Tools

- Limited software installed
- No RPM or .deb available
- Tools in Containers - “Super-Privileged” containers and System Containers
- CLI tools to manage tools as containers
  - CoreOS: toolbox
  - Atomic: atomic [install|run]

CLI Tools - Container Linux: toolbox

- https://github.com/coreos/toolbox
- Fedora base container
- Adjustable container path in config

```
core@core-01 ~ $ toolbox
Downloading sha256:00ddb097f3f [==============================] 79.3 MB / 79.3 MB
successfully removed aci for image:
"sha512-435b5cd978b49ab9b4d40187127fc9b24fadad1afd283aeaa7f07486da54878e"
rm: 1 image(s) successfully removed
Spawning container core-fedora-latest on /var/lib/toolbox/core-fedora-latest.
Press ^] three times within 1s to kill container.
[root@core-01 ~]# ...
```

CLI Tools - Container Linux: toolbox

- Install RPMs “in the container” and apply them to the system

```
...
Press ^] three times within 1s to kill container.
[root@core-01 ~]# dnf -y install traceroute
...
```

Atomic CLI

Used to manage host updates and system containers

```
atomic install [--system] [--storage=ostree] <image>
atomic run <image>

atomic host upgrade
...
Reboot
...
atomic host rollback
reboot
```

http://www.projectatomic.io/docs/usr-bin-atomic/
https://github.com/projectatomic/atomic

Deployment Targets and Methods

- Bare Metal
  - PXE boot - run in memory or install to disk
  - cloud init - set identity and SSH keys
- Cloud Providers
  - AWS - official images for CoreOS, load your own image for Atomic
  - GCE
  - OpenStack - load boot/install images
  - ...
- Personal VM (vagrant)
  - Virtualbox - Atomic and CoreOS
  - Libvirt - Atomic only - much faster

LAB 2 - CLI Tools

Session 4 - Customizing Container Hosts

- Build Process
- Update Repositories
- Generating Images

Customization - Container Linux

- Build process based on Chromium (upstream ChromeOS)
- Uses ‘repo’ tool, packaged for .deb, binary download for Fedora
- Builds in chrootish environment
- Leaves behind bind mounts - clean up carefully
- Must always build “image” - image == /usr+

SOURCE - https://coreos.com/os/docs/latest/sdk-modifying-coreos.html

Requires manual patch: https://github.com/coreos/chromite/pull/23 - 20171008

Customization - Container Linux

The install process pulls from a remote web server and checks the headers.
Fix cros_sdk.py

```
mkdir -p ~/bin
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod 755 ~/bin/repo
mkdir coreos
cd coreos
~/bin/repo init -u https://github.com/coreos/manifest.git
~/bin/repo sync
sed -e -i '/HTTP\/1.1/a\          header.startswith("HTTP/2 200") or' \
 chromite/scripts/cros_sdk.py
sudo ./chromite/bin/cros_sdk
... build commands
```

SOURCE - https://coreos.com/os/docs/latest/sdk-modifying-coreos.html

Requires manual patch: https://github.com/coreos/chromite/pull/23 - 20171008

Customizing Atomic Host - rpm-ostree repo

- ostree built from RPM sources
- ostree is both a host management system and a code repository
  yum/apt-get for binary installation
- No requirement to build bootable images (though it is possible)
  Install from stock, repoint, rebase
- Fedora, CentOS and RHEL are maintained
- Goal is continuous automatic update of the OS

rpm-ostree-toolbox:

- https://github.com/projectatomic/rpm-ostree-toolbox
- https://developers.redhat.com/blog/2015/01/08/creating-custom-atomic-trees-images-and-installers-part-1/
- https://developers.redhat.com/blog/2015/01/15/creating-custom-atomic-trees-images-and-installers-part-2/

Repo container:

- http://www.projectatomic.io/docs/compose-your-own-tree/ < +++!
- http://www.projectatomic.io/blog/2014/08/build-your-own-atomic-centos-or-fedora/
- https://dustymabe.com/2017/08/08/how-do-we-create-ostree-repos-and-artifacts-in-fedora/

Building an rpm-ostree

The build process for ostree images has changed somewhat. It has been automated to the point that doing it manually is not well documented.

The process uses the `rpm-ostree` tool and the `rpm-ostree-toolbox` but it is unclear which is supported

```
Sudo dnf install rpm-ostree-toolbox
git clone https://pagure.io/fedora-atomic.git
sudo rpm-ostree-toolbox treecompose -c fedora-atomic/config.ini
```
…. Wait a long time

SOURCE: https://dustymabe.com/2017/08/08/how-do-we-create-ostree-repos-and-artifacts-in-fedora/

Using your ostree

Updates - CoreOS

Update-engine and Locksmithd
Controlled by:

- `/usr/share/coreos/update.conf`
- `/etc/coreos/update.conf`

Sets path to repo and stream (alpha, beta, stable)

Disable auto update by ~~`/usr/.noupdate`~~ stop `update-engine` and `locksmithd`

https://coreos.com/os/docs/latest/update-strategies.html

Updates - Atomic

Define repo

```
/etc/ostree/remotes.d/centos-atomic-host.conf
```

```
atomic host upgrade
reboot
atomic host rollback
```

http://www.projectatomic.io/docs/os-updates/

Wrap Up

Container Hosts:

- Good for container orchestration, small single-service hosts
  - Light weight
  - Cattle
- Not so good for custom or complex monolithic apps
  - Decompose to containers (microservices - blegh, but yeah)
- Need to adapt sysadmin skills
- Need to adapt dev practices
- Use System Containers
- Plan updates then automate and let them go

Thanks!

Leave me your email address and I'll compile a set of links and references and send them out to you.

- Email: markllama@gmail.com
- IRC markllama on freenode