OAK RIDGE National Laboratory

Anchor: Diskless Cluster Provisioning Using Container Tools

Joseph Voss

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

OAK RIDGE National Laboratory

About Me

- Joseph Voss
- Systems Engineer @ Oak Ridge National Laboratory
- Student Cluster Competition Alumnus

Overview

- Background
  - Diskless provisioning, tools
  - Challenges encountered
- Anchor
  - Building images
    - Buildah, base, client
  - Deploying images
    - Kubernetes, Management server
- Benefits/Drawbacks
- How to use it
- Wrap Up

Background

- Diskless provisioning
- How the current tools work
- Challenges encountered
- Goals looking forward

Diskless provisioning

- Consistent configuration across a large machine is hard
  - Rolling out upgrades, configuration drift, disk failures
- Network booting nodes using same OS image
  - Single base image
  - Guaranteed known state
- How are these images created and served?
  - xCAT, HPCM, GeDI
- Full cluster management vs boot process

How do the current tools work? (GeDI)

- Chroot image build
- Run configuration management tools
- Serve image over NFS with SYSLIUX
- Node boots
  - Mount OS image read only
  - Sync writable areas to RAM
  - Post-boot configuration

Challenges encountered

- GeDI
  - Build images in chroot jails, images mounted by NFS
  - Developed in 2003
- Complexity could be handled better by other tools
  - Chroot jails are part of a very *in* technology right now
- Single point of failure at run time
  - NFS server

Goals looking forward

- Simply diskless deployment
- Reduce complexity we maintain
- Leverage existing industry-standard tools
- Remove single point of failure
- Provide deployment flexibility

OAK RIDGE National Laboratory

Anchor

- Building images
  - Base images, client images, examples
- Deploying images

Building images

- Caveat – images builds/configuration changes between centers
- Two image build types; base and client
- Base images built every patching cycle
- Client images are built for system upgrades
- Use container building tool Buildah
  - coreutils-like interface for container builds

How to build a base image?

```
$ buildah from scratch
working-container
$ buildah mount working-container
/var/lib/containers/overlay/…/merged
$ yum -c “${YUM_CONFIG}" install --installroot/var/lib/containers/overlay/…/merged \
        -y @base @core kernel-devel
…
$ buildah commit working-container "${REGISTRY}”/rhel-7-x86_64-base-image:20201119
…
```

How to build a client image?

```
$ buildah from “${REGISTRY}”/rhel-7-x86_64-base-image:20201119
rhel-7-x86_64-base-image-working-container
$ buildah copy rhel-7-x86_64-base-image-working-container build_script.sh /build_script.sh
…
$ buildah run rhel-7-x86_64-base-image-working-container /build_script.sh
…
$ buildah commit …
```

What does this buy us?

- Reducing technical debt, external tools
  - Build process is still custom
- Shippable software stacks
  - Test, stage, recompile
- Centralized image store
  - Free distributed back ups
- Repeatable, immutable images
  - Images referred by hash of contents
  - Git-ops and CI, images tagged with build inputs

Deploying images

- Local squashfs in ram, read-write overlay
  - Squashfs – compressed read-only file containing file system
  - Overlay – union file system to merge mounts
- Needed at early boot – Dracut module
  - Bootstrap authentication
    - ACME, SSH, NFS
  - Copy image to local node, create mounts
    - rsync, buildah
- Different deployment methods vs cluster size

Deploying images: Kubernetes

- Dedicated machines for smaller clusters don't make sense
- Center-wide DHCP/TFTP server, iPXE
- Node classifier: Matchbox
- Authentication: Step-CA ACME provider – Let's Encrypt
- Image serving: Docker registry, download and squash live (!!!)

Example iPXE script

```
$ curl matchbox.example.com/ipxe?hostname=small_cluster_node1

#!ipxe

initrd http://example.com/initrd

initrd http://example.com/ca.pem /ca.pem

kernel http://example.com/vmlinuz ip=dhcp BOOTIF=${netX/mac} root=anchor
init=/sbin/init acme_email=admins@example.com acme_server=step-
ca.example.com/acme/acme/directory buildah_registry=registry.example.com
buildah_image=compute_client_demo:latest overlayfs_size=1024m overlayfs_write=/


boot
```

Deploying images: Management nodes

- Theoretically can scale easily, practically not so much
  - Docker registry
- Local DHCP server on management, TFTP, iPXE
- Node classifier: Matchbox
- Authentication: SSH – push out client images at boot
- Image: None, image in RAM. Just mount

Example configuration

```
$ curl matchbox.management.com/ipxe?hostname=large_cluster_node1

#!ipxe

initrd http://example.com/initrd

initrd http://example.com/root-key.pub /root-key.pub

kernel http://example.com/vmlinuz ip=dhcp BOOTIF=${netX/mac} root=anchor
init=/sbin/init dropbear_auth_key=/root-key.pub squashfs_mount_only=1 overlayfs_write=/
overlayfs_size=1024m overlayfs_write=/


boot
```

Benefits and Drawbacks

- In production >1 year, middle of migration

+	-

• No run-time dependency, independent
  • After nodes boot, services can spin down.
	• Simpler == dumber
	  • More onus on configuration management tooling

• Deployment flexibility
  • Same build process, different deployments
  • Issues with one? Swap to the other
  • OS fully r/w
	• RAM usage
	  • Local compressed image in RAM, loss ~1 GB usable space

Great, how do I use it?

- Github: https://github.com/olcf/anchor
  - Mostly dracut module, but has a few example build scripts/helm charts
- Install dracut module, build initrd, configure through kernel comandline
  - Need PXE environment? Dusty Mabe's blog post here
- Contributing: Future authentication/image copy modules
  - Trust bootstrapping is hard. ACME and SSH used here, TPMs ideal

Wrapping up

- Tl;dr: Cluster management is hard, leverage existing tools that make it easier
- Flexibility is important
- Don’t be afraid to revisit boot processes!
- Available for questions:
  - Links and handles @ josephvoss.com
- Thank you!