Agenda

- How it all started
- Why should I care?
- What are containers?
- Gimme more!
- Show me!
How it all started
Bimodal IT – Challenges & Opportunities

Malcom McLean
How does it apply to me?

• Running applications?
• Providing services?
• ...?
Why should I care?
Bimodal IT – Challenges & Opportunities
The two brains of IT

<table>
<thead>
<tr>
<th>Mode 1</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Agility</td>
</tr>
<tr>
<td>Waterfall, ITIL</td>
<td>Agile, DevOps</td>
</tr>
<tr>
<td>Conventional Projects</td>
<td>New &amp; Uncertain Projects</td>
</tr>
<tr>
<td>Long-cycle Times (months)</td>
<td>Short Cycle (days, weeks)</td>
</tr>
</tbody>
</table>
45% of organizations claim to have some form of bimodal capability today.
Challenges in Context of Containers

Developers
Frequent releases vs. staged production schedule.
“It works on my machine.”

Operations
- Manage growing services
- Reliability and uptime of new applications
- Time to market
- Efficiency

New features; Faster please!
What are containers, really?
Containers

OS-level or application virtualization with Linux Containers (LXC) and container engine. Support for Windows Subsystem for Linux (WSL).
What are containers – two views

• **Operations**
  • Components of Linux kernel and OS
  • Image format, specific tools
  • Isolation
  • High density
  • Smaller, lighter, faster
  • Orchestration, management

• **Applications**
  • Packaging
  • Share easily
  • Easily extensible
  • Scale up/down
  • Self-contained
  • Micro-services
Containers are awesome!
Linux Containers

• System containers
  - Full system in the container (no kernel)
  - libvirt-lxc

• Application containers
  - One process per container
  - Docker, podman, ...
  - Rich ecosystem
Linux Containers

Application container

App A
Bins/Libs
Guest OS

App A'
Bins/Libs
Guest OS

System container

App B
Bins/Libs
Guest OS
Kernel

App B'
Bins/Libs
Guest OS
Kernel

Hypervisor (Type 2)

Host OS

Server
Advantages of Linux Containers

Lightweight virtualization solution
- Isolated from the other processes
- 1 kernel to rule them all
- Normal I/O
- Dynamic changes possible without reboot
- Nested virtualization is not a problem
- No boot time or very short one

Isolate services (e.g. web server, ftp, …)
Much more (see further)

...
Linux Containers – Limitations

They cannot run a different OS/architecture
  – Cannot run Windows containers on Linux

Risk of escaping from containers
  – Solution: user namespaces

Shared kernel with the host
  – Syscall exploits can be exploited from within the container
  – Solution: seccomp2

Security measures
  – Patch, don’t use root, kernel capabilities, confinement
  – Use VMs
Containers and orchestration

• **Standalone container host**
  - SLES, container engine, registry (Portus)

• **Orchestrated datacenter**
  - SUSE CaaS Platform (Micro OS, K8s)
  - Containerized applications, micro services

• **Bi-modal datacenter**
  - SUSE CaaS Platform + SUSE OpenStack Cloud
  - Combination of traditional IT + agile (containers)
Too much going on, dive deeper!
(And show me!)
Containers are standardized

• OCI runtime specification:
  • Defines container runtime (API, data structures, …)
  • How to start/stop/... containers
  • OCI provides a reference implementation: runC

• OCI image format specification:
  • Defines how a container image is structured

• Result:
  • Avoid vendor lock-in
  • Avoid fragmentation
  • Containers are truly portable
  • Foster innovation
Running containers

• Stand-alone node:
  • docker
  • podman

• Container orchestration - kubernetes:
  • docker
  • containerd
  • CRI-O
  • ...
Introducing podman

- Drop-in replacement for docker
- Focuses on single node operations, close to docker 1.13
- No daemon
- Relies on runC
- Network implemented using CNI
podman extra features

- Has the concept of "pods":
  - Works like with kubernetes
  - Allows to group several containers together
  - Remove some isolation features on purpose (namespaces, cgroups)
- Can work in rootless mode:
  - Regular unprivileged users can create containers
  - Containers are visible only to the user who created them
  - Makes containers even more secure
Building containers

• Most of you are probably using "docker build" but...
• Other ways to build container images exist
• Images delivered by SUSE are not built using docker:
  • Base container images
  • Derived images, think about all the CaaS Platform ones

How could that work?
Repetition: standards matter!

- Container images follow the OCI image specification
- This is what grants image portability across container engines
- Different ways to build OCI images:
  - docker
  - podman build
  - buildah
  - KIWI
  - ...
Building with docker

- Start from an existing container image (the "base" image)
- Write a Dockerfile
- Use Dockerfile directives to:
  - Execute commands: most used one "RUN" -> install/build software, ...
  - Write image metadata
  - ...

Building with podman

• Start from an existing container image (the "base" image)
• Write a Dockerfile
• Use Dockerfile directives to:
  • Execute commands: most used one "RUN" -> install/build software, ...
  • Write image metadata
  • ...

YES – it's like the previous slide, podman is a drop-in replacement for docker open-source engine!
Building with buildah

• Can build using a simple Dockerfile
• Allows more flexible build mode:
  • Start from existing image, create a container
  • Mount the container rootfs on the host
  • Interact with the container rootfs from the host: cp, scripts, zypper,…
  • Can produce small images with zero external dependencies (no need to have zypper around or in the history of the image!)
Building with KIWI

• Appliance builder used at SUSE since a long time
• Steeper learning curve compared to the others
• Integrates nicely with the Open Build Service:
  • Automatic rebuilds of the images on package updates
  • Automatic rebuilds of derived image after base image is updated
• Note well: OBS supports also builds using special Dockerfile
Pre-built images

• Docker HUB
  - Community, handle with care!
• SUSE Registry (registry.suse.com)
  - Enterprise contents, secure, verified, signed
  - SUSE Products (CaaS Platform, Cloud Application Platform, …)
  - What used to be in SLES Containers module (e.g.: Portus)
Interacting with SUSE registry

• SUSE publishes all its product images to registry.suse.com
• SUSE products will automatically download images from there
• This can be done in two ways:
  - Manifest file
  - Helm charts
• SUSE’s helm charts are hosted on a public helm chart repository operated by SUSE
New world, old problems

• Pulling images from an external registry can be expensive (time, bandwidth)
• Pulling isn’t even possible in some scenarios (air-gapped environments)
• The same applies to helm charts
• RPM world had the same problems: solved with tools like SMT (more recently RMT)
Registry mirroring

- Provide our customers a way to mirror the contents of an external registry into an on-premise one
- Solution available since CaaS Platform v3
- More plans to improve it over the time
Air-gapped scenario

- Most complex case
- Container hosts don’t have access to the internet
- Nodes must be able to pull containers from local registry
- We don’t want to change names of the container images

\[ \text{registry.suse.com/caasp:1.0} \]
\[ \text{should NOT change name} \]
\[ \text{(eg: my-registry.acme.lan/caasp:1.0)} \]
Architecture

registry.suse.com

mirror.local.lan

Secured drive with registry contents

mirror.secure.lan

air-gapped network

node1

node2
Helm chart mirroring

• Helm charts can be downloaded using “helm-mirror”
• The charts can be copied to a local HTTP server
• Charts are just static files
Container images mirroring

- Use “helm-mirror” to get a list of all the images referenced by the charts
- Use “skopeo sync” to download all the images:
  - Save the images into a local USB drive
  - Connect the drive to a machine inside of the air-gapped network
  - Use skopeo sync to import all the images into a local registry
- Configure the container engine to use the local registry as a mirror of registry.suse.com → no need to re-write image names
Container engine: mirroring support

- Out of the box docker supports mirroring only for the Docker Hub
- We have a patch extending that, still going through upstream review
- SUSE CaaS Platform v3+ have the patch applied
- CRI-O patch is under review from upstream
Container mirroring: next steps

• Add Portus to the picture: used to secure on-premise container registry in a more granular way
• Allow the same registry to be used to mirror multiple external registries
• Other ideas… please provide feedback
And, finally...
Some Related Content @ SUSECON 19

- Best Practices in Deploying SUSE CaaS Platform [TUT1131]
  - Tuesday @10:15, Wednesday @2:00
- Enabling Business Continuity with SUSE CaaS Platform [BOV1078]
  - Tuesday @2:00
- Getting Started with Kubernetes [HO1113]
  - Tuesday @2:00, Wednesday @2:00
- SUSE CaaS Platform Hands-On [HO1209]
  - Tuesday @4:30, Wednesday @2:00
- OpenStack and Kubernetes Get the Heat Treatment [HO1421]
  - Tuesday @10:15, Wednesday @10:00
- SUSE CaaS Platform vs SUSE Cloud Application Platform - Technical Tutorial [TUT1338]
  - Tuesday @10:15
- Simplifying AI Applications with Containers and K8s [TUT1151]
  - Wednesday @10:00
- Bringing container security to the next level using Kata containers [TUT1201]
  - Tuesday @4:30, Wednesday @3:15

- Visit our Kiosk in the Technology Showcase
- Join the Kubic project in the Developer Lounge
More Related Content @ SUSECON 19

- GitLab on SUSE CaaS Platform [HO1415]
  - Tuesday @10:15, Thursday @2:00
- SUSE CaaS Platform as a Service: Deploying on Demand CaaS Platform Clusters [TUT1308]
  - Tuesday @3:15, Wednesday @4:45
- SUSE Application Delivery Platform Future [FUT1429]
  - Wednesday @4:45, Thursday @3:15
- Tymlez Blockchain on SUSE CaaS Platform [BOV1313]
  - Tuesday @10:15
- SUSE CaaS Platform: Day 2 - Where Do We Go From Here? [TUT1076]
  - Tuesday @4:30
- Secure Modern Application Delivery with SUSE CaaS Platform [TUT1266]
  - Tuesday @4:30, Thursday @3:15
- Integrating Identity with LDAP for SUSE CaaS Platform [TUT1254]
  - Tuesday @10:15, Thursday @3:15
- Deploy and Scale Orchestrated Container Infrastructures on Your Own [TUT1166]
  - Tuesday @4:30

- Visit our Kiosk in the Technology Showcase
- Join the Kubic project in the Developer Lounge
Still More Related Content @ SUSECON 19

- DevOps Pipelines on Jenkins-X with SUSE CaaS Platform [DEV1118]
  - Wednesday @3:15, Friday @10:15
- Deploy a Modern Data Platform with SQL Server 2019 on the SUSE Platform [SPO1456]
  - Thursday @11:15
- openSUSE Kubic - exploring the containerized frontiers [DEV1111]
  - Tuesday @11:30, Thursday @3:15
- Geekos, Containers, and Clouds.. Oh my! [CAS1406]
  - Tuesday @4:30, Thursday @2:00
- Containerized OpenStack: Preview Inside [TUT1273]
  - Thursday @4:30
- Containers Today and Beyond [TUT1198]
  - Tuesday @10:15

Demo?
- Visit our Kiosk in the Technology Showcase
- Join the Kubic project in the Developer Lounge