An Intro to Containers

By Alex Bissessur

History of Containers

- chroot originated with Bill Joy in BSD in 1982
  - lets you create a directory to virtualise the file system
  - merged while drinking, or smoking, or both
- chroot developed into Jails for BSD
  - still in use today
- Jails was targeted at running binaries in an isolated environment
  - no access to main filesystem
  - no ability to make changes to host OS

BSD's Jail

Jails: Confining the omnipotent root.

The FreeBSD "Jail" facility provides the ability to partition the operating system environment, while maintaining the simplicity of the UNIX "root" model. In Jail, users with privilege find that the scope of their requests is limited to the jail, allowing system administrators to delegate management capabilities for each virtual machine environment. Creating virtual machines in this manner has many potential uses; the most popular thus far has been for providing virtual machine services in Internet Service Provider environments.

Sun’s Zones

- Focus shifted to OS virtualisation
- Built to run more complex applications in isolated environments
- Consolidating apps to a single Big Boy (Sun wanted to sell servers)
- Again, protects host OS from the applications, and provides abstraction layers for security

Core OS-based Virtualisation Principles

- Security
- Isolation
- Virtualisation
- Granularity
- Transparency

NOT

- To run foreign binaries
- Emulate other OS
- To consolidate stacks

Virtual Machines

- Before VM OS, there were individual machines for isolation
- Virtual Machines are heavy
- Every VM needs the kernel to be emulated
  10 Vms => 10 kernels
- Operating systems don’t do nothing
- Leads to expensive resource bloat

The Hypervisor

- Is an OS fast?
- Yes.
- Is an OS on an OS fast?
- No.
- Running VMs is expensive.
- VMs today are the embodiment of paranoia.

Containers Today

- Docker is the leading containerisation standard
  RedHat has Podman because RedHat is *different*™
- An encoding of deployment procedures in images
  https://github.com/cloud-native-mauritius/cloudnativemauritius.com/
- Images are reproducible and portable between machines (and OS-es)
- The developer becomes *the operator*.

Containers Today

- Containers sit on top of the kernel
- VMs virtualise hardware. Containers virtualise the OS
  - Containerise applications, not the OS
- Licensed and secure with SUSE/RedHat tooling
- Simplicity is best

Questions or Comments?

The Development Process @ SWAN

- I write a .NET application
- I build it for prod and wait 2 minutes
- Go to the folder, manually edit a config file to make the app run on IIS
- Virtual Desktop to the server
- Copy my files over in the kbps speeds
- (re)Start the service

And it doesn’t work.

The Development Process @ SWAN

- Write application in Python
- Hans clones project & tries to run it. Fails.
- Install/update dependencies
- Conflicting dependencies for different programs
- Inconsistencies between my laptop & Linux server

And the deadline’s approaching.

How Containers Help

- Uniform development environment
- Dependency and package conflicts avoided
- Security risks do not spread to other applications
- Easier control over environment your apps run in

How Containers Help

- Containers use the host kernel
  Less wasted resources
- No 2nd OS (hypervisor) to run
- Integration with CI/CD pipelines
- Prevent over-provisioning of resources
- Facilitates microservices architecture
- Opens the door to Kubernetes & Cloud

Downside of Containers

- Containers are Linux-centric (sorry Windows)
- Containers are not forcibly secure
  - Your applications and packages must be kept up-to-date
  - Containers are not *entirely* isolated – on Linux they are just another process

Questions and Demo