NFV and Containers
Evolution or Revolution ?

Huawei Nov 2015
Daniel Veillard <veillard@redhat.com>
a.k.a. 李达尼

Presentation

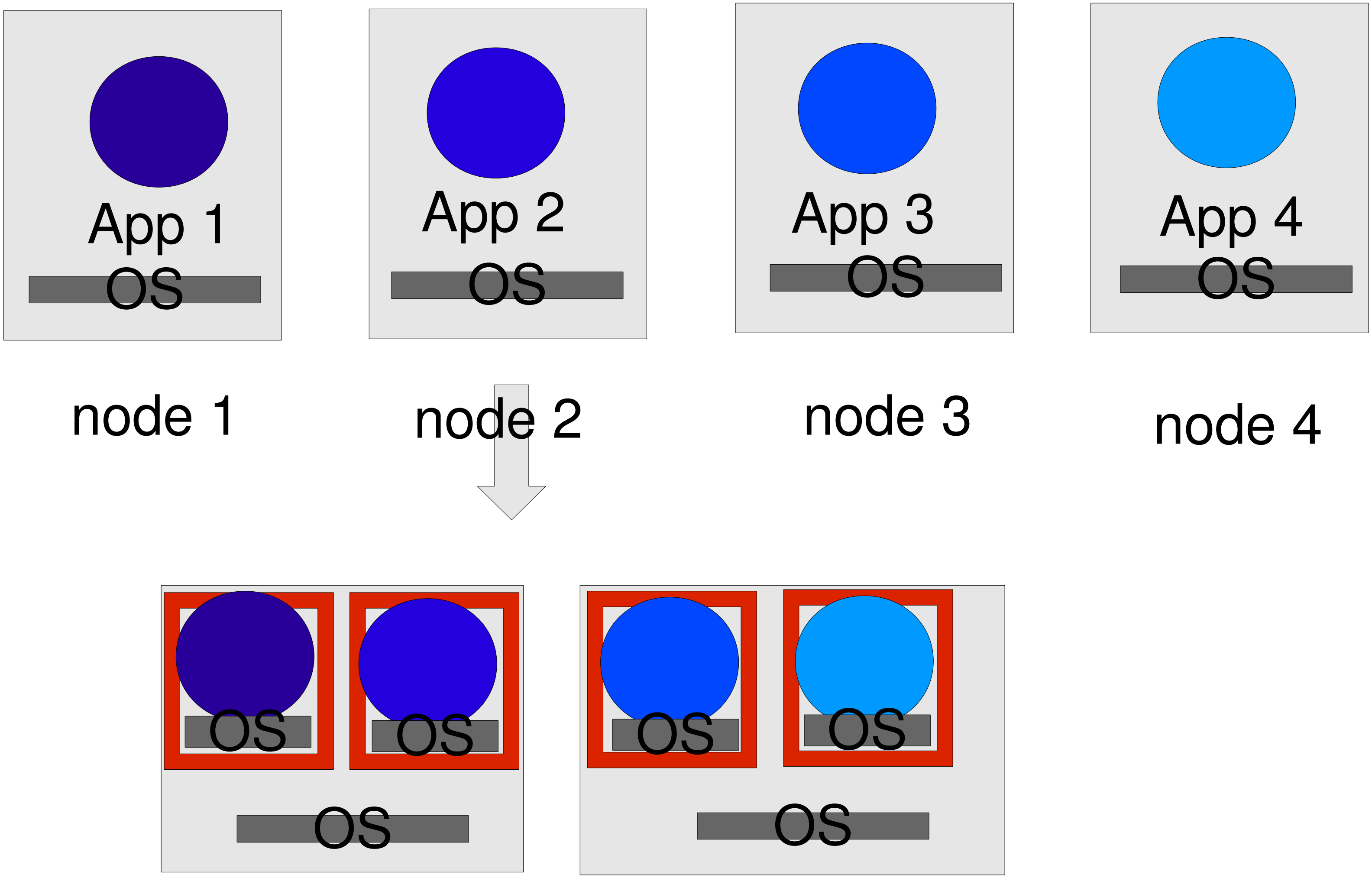
- Working at Red Hat
  - Since 2001, previously at W3C
  - RHN, Desktop, Virtualization as developer
  - Manager for Standards and NFV in OSAS
  - Manager of a tools team on Containers
- Libxml2 and libxslt
  - Created in 98
  - Main author, maintainer of the libraries
- Libvirt
  - Created in 2005, 10 year anniversary on Monday!
  - Main initial author
  - Releases maintainer

NFV revolution

- Virtualize the compute nodes
- Possible due to technology improvements in virt
- Cheaper
- Cost effectiveness of dynamic placement and scaling
- More control over execution
    - Migration
    - Resource control
- Most workload can be kept mostly unchanged

Virtualizing the workloads

- This mostly leaves the application and their OS untouched

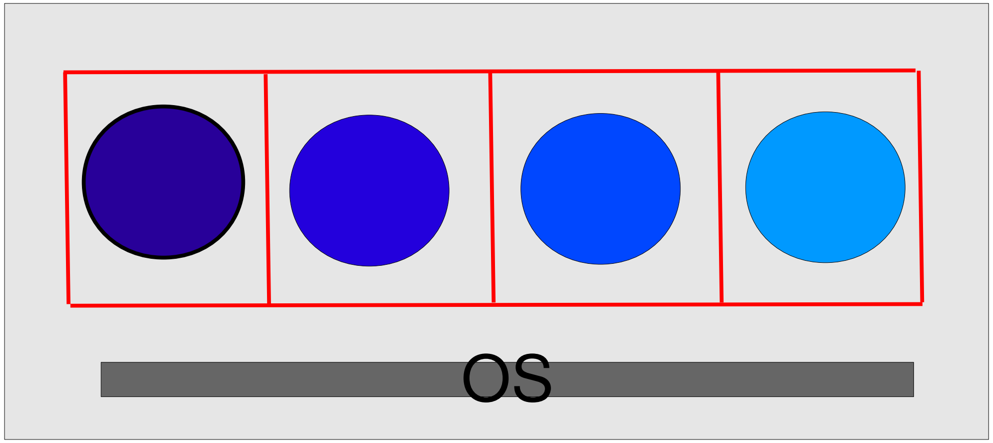


Containers

- First seen as a lighter way way to virtualize
- Based on partitioning the resources between applications
- Based on kernel support like cgroups and namespaces
- Single kernel on the node
- The applications run on top of the base OS but with a limited view of the resources
- Weaker inter-application protection
- No support for migration in general
- But very efficient:
  - Very lightweight
  - One kernel to rule them all
  - Achieves very high density levels

Containerizing the workload

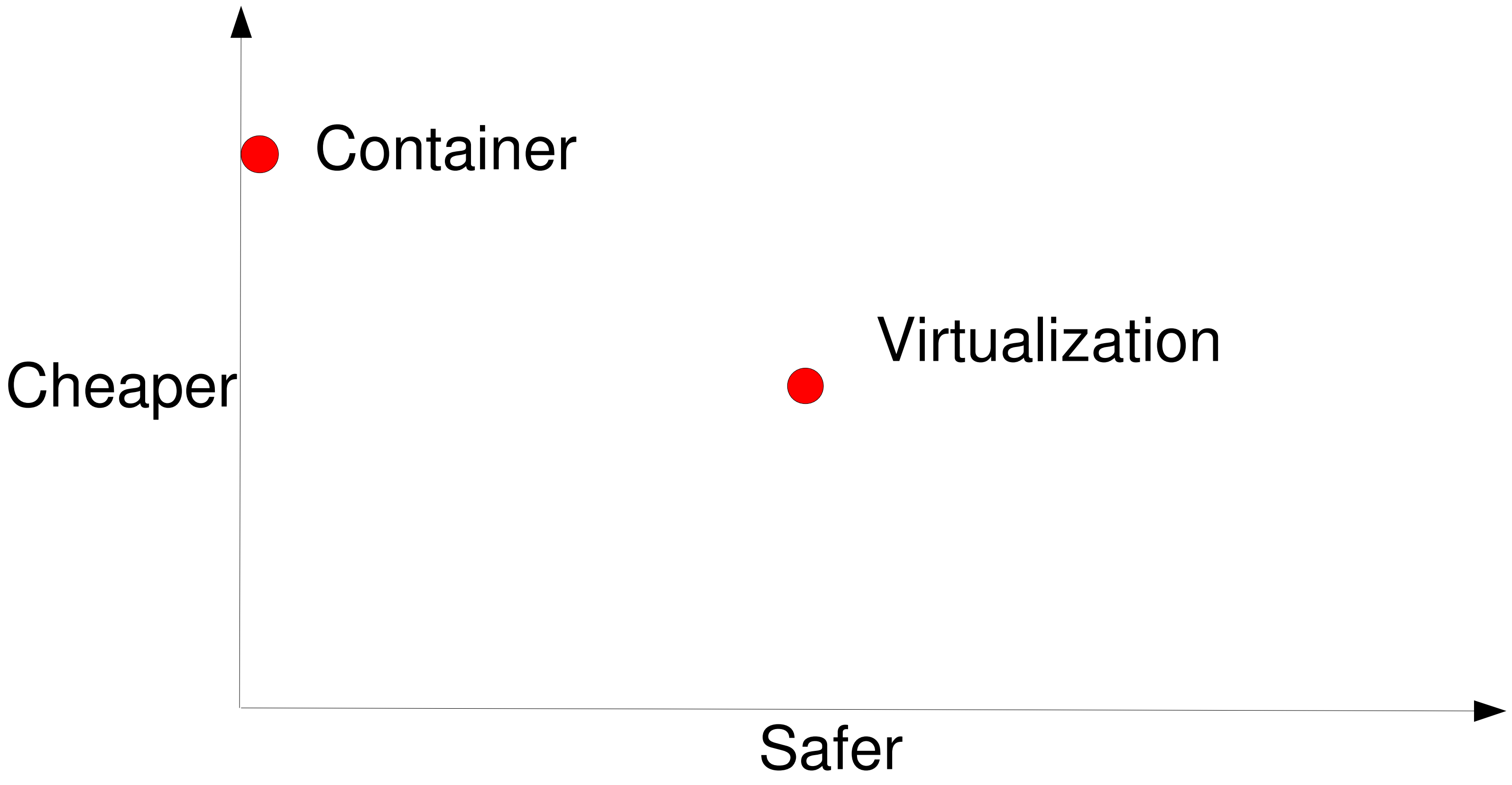
- Usually one container per process
- They all share the same kernel and base OS
  - so need to be compatible
- In practice most of the required libraries and helpers are put in the container
  - Minimize the level of dependancy and requirement
  - Raises the problem of updates



Containers for NFV

- ETSI NFV looking at containers
- Easier to give direct hardware access
- Better efficiency
  - Scheduling flow as steps in a pipeline has less overhead
  - Density, and a single kernel
- Isolation is not at the same level as with virt
  - SELinux and other kernel mechanisms
  - If a virt kernel crash it affects only one app
- Single kernel and base OS means standardization

Drivers for technical evolution

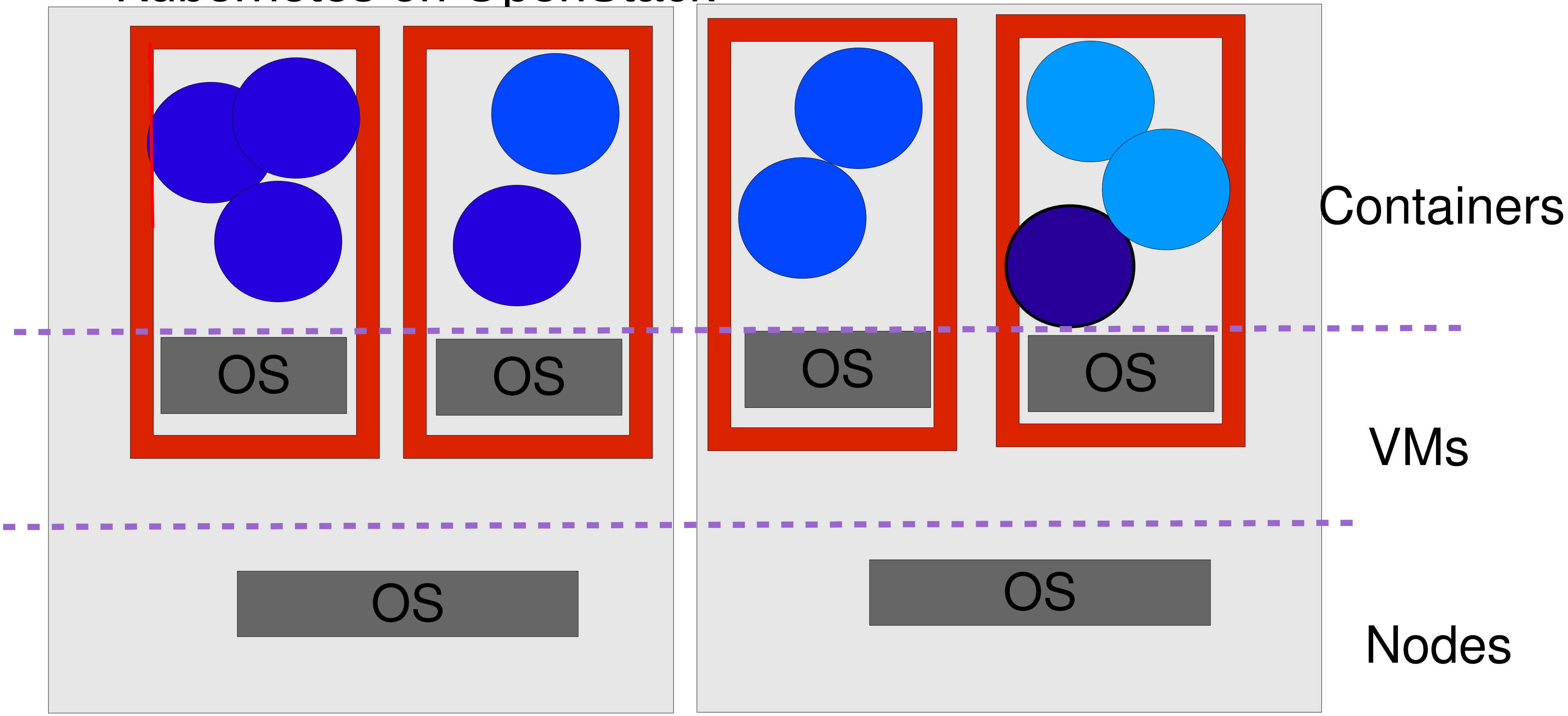


Containers vs. Virt environments

- The problem of APIs
- NFV picking up OpenStack
  - Not an ideal support for container directly
  - Libvirt has a container driver but not used much
- Containers have dedicated frameworks
  - OpenShift
  - Kubernetes
  - Mesos + marathon
  - …
- Virtual workloads are not scaled up/down as simply
- It's also an application problem

The 3 layers cake

- e.g. OpenShift on AWS or Openshift on OpenStack
- Kubernetes on OpenStack



Container software model: Docker

- Provide tools to provision the content of containers
- Define this as the model to build and deploy applications
- Make it independent of the base OS (mostly)

=> Suddenly containers become sexy

Rebuilding legacy apps

- Application are usually multi-processes
    - Splitting the apps into multiple containers
    - Define APIs
    - Build scaling in and out using container instances
- Break classic model of dependencies on a base OS
    - Bundle libs in the package
    - Container inheritance
- Need support from tools on how to redefine the apps

Common use case outside of NFV

- Content provider (Google, Seznam, BBC …)
- Speeding up the workflow and delivery of apps
  - From developper to live in hours
  - DevOps kind of workflow
  - Implementation of CI/CD workflows
- Web applications
- Data crunching

Certification of containers on the base OS

- Contrary to virt, the OS is not part of application delivery
- The 'surface of contact' between the application and OS
  - Is larger
  - Is beyond just the kernel APIs
  - Parts moving in the base OS can affect the container
- So applications need to be 'certified' against the OS
  - Vendor certification e.g. Red Hat certification
  - In-house certification
- The trend is to have minimal OS versions dedicated
  - Red Hat Enterprise Atomic
  - CoreOS
  - ...

Way forward and collaboration

- Look in the NFV catalog functions:
  - That are already service based
  - That do not require the protection of virt
- Modify the application to be container ready
  - Convert them to run in one container (automatable)
  - Split the containers at the service boundaries
- Define orchestration requirements for the app
- We can help with this !

Conclusions

- In the last 2 years containers moved from evolution
    - Cheaper application isolation
    - Integration in the virtualization stack
- To revolution
    - Define a new application format
    - New software delivery mechanisms
- Revolution for NFV as the traditional workloads are transitionned to the new model
- This will impact future definitions of NFV standards as done by ETSI
- This will impact the OS vendor relationship
- Some workloads will not change easilly, normal virtualization will still be available
- Be ready for a 3 layer cake: physical + virt + containers