Virtualization and Containers

Open Source Software

Wesley Turner

Rensselaer Polytechnic Institute

Department of Computer Science

Reading Material

Reading Material

- **Virtualization**
  - https://en.wikipedia.org/wiki/Virtualization
  - https://en.wikipedia.org/wiki/Hardware_virtualization
- **Using Virtual Machines and Docker:**
  - https://goo.gl/sRVT7o
- **Oracle VirtualBox VM (Reference)**
  - https://www.virtualbox.org/manual/ch01.html#virt-why-useful

Introduction

What is Virtualization?

We will consider two main types:

- Virtual Machines
  - Software to allow a piece of hardware to run multiple operating system images at the same time
  - Eg. *VirtualBox*
- Containers
  - A lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it
  - Eg. *Docker*

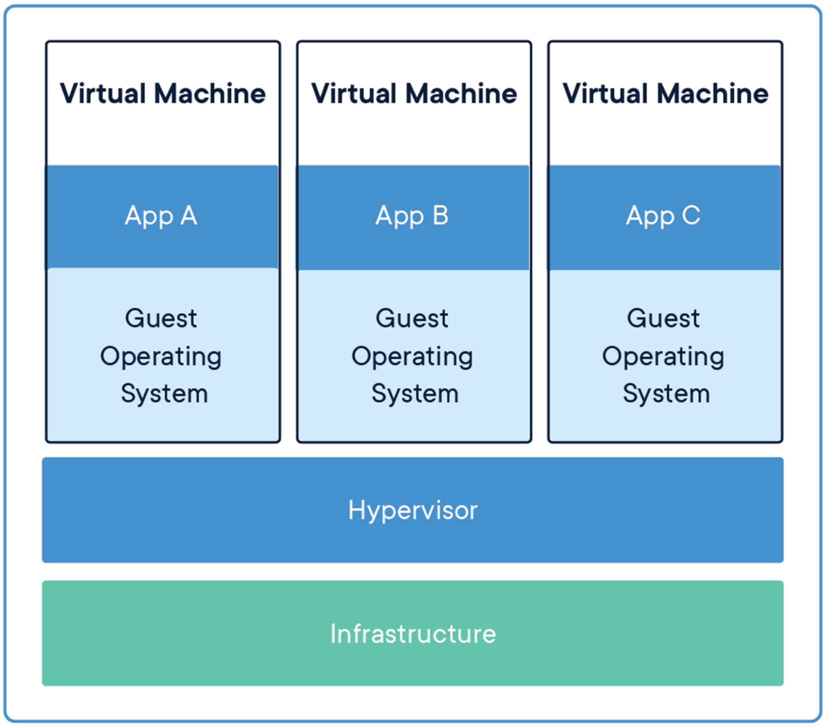
Virtual Machines

Concept

In a Virtual Machine:

- A software layer *(hypervisor)* emulates the underlying hardware
- An additional operating system runs on the virtual hardware
- Users interact with the virtualization layer as if it were the underlying machine

Architecture



https://www.docker.com/resources/what-container/#/virtual_machines

Advantages

- Virtual Machines share resources
  - Memory, disk, network bandwidth
  - Better resource utilization
- They run in isolated environments
  - Better security
- The same hardware can run multiple operating systems
  - Eg. Windows and Linux can coexist on the same machine at the same time
- An entire environment can be distributed
  - Less time wasted setting up environments
  - Less confusion about versions or compatibilities
- Machine states can be checkpointed
  - Can be migrated to a more or less powerful machine as needed
  - Rapid return of services when hardware goes down

Disadvantages

- Virtual Machines share resources
  - Memory, disk, network bandwidth
  - Poor planning and multiple VMs can drown a machine
- The same hardware can run multiple operating systems
  - May require multiple licenses
  - A problem with one set of hardware can affect hundreds of VMs
- An entire environment can be distributed
  - The entire guest environment -- including the entire guest OS, disk and memory -- needs to be represented in the VM file
    - VMs are large
  - Each environment communicates to resources through the hypervisor
    - Guest OS <-> hypervisor <-> native OS
      - Can significantly impact performance
- Machine states can be checkpointed
  - Checkpoint files can be huge

Virtualization and the Cloud

Virtual machine are the *de facto* way of using cloud compute resources

- **You** are protected by the virtual environment by interference from other VMs
- Other VMs, and the provider, are protected from **you** as well
- You can easily add VMs or migrate to more/less powerful VMs as your needs change

You do not necessarily know what hardware you are running on

- Performance can be different on different systems
- Performance can be different on different clusters
- Performance can be different in different locations (proximity sometimes matters)

An Example

The VA Scheduling Competition

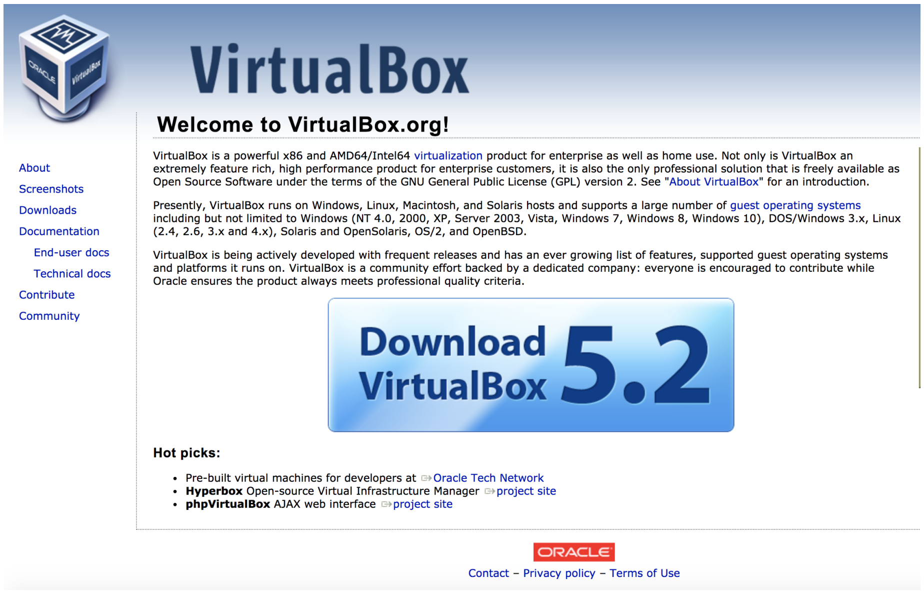
An interesting cloud use for Virtual Machines: https://vascheduling.devpost.com/

- The VA was investigating a new scheduler for their Electronic Health Recorder (EHR) system
  - Contestants were each given a Virtual Machine with access to nominal (simulated) patient data and a version of a running VISTA EHR
    - The machines ran in the cloud on Rackspace https://www.rackspace.com/

- The VA scheduler (continued):
  - They developed their scheduling solution on the system
  - At the end of the competition, the machines were checkpointed and submitted to OSHERA (Open Source Electronic Health Record Agent) for evaluation
- Three awards were given:
  1. https://devpost.com/software/health-etime
  2. https://devpost.com/software/oh-scheduler
  3. https://devpost.com/software/hp-open-community-team-submission

VirtualBox

VirtualBox



Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Hot picks:

- Pre-built virtual machines for developers at ⇨Oracle Tech Network
- **Hyperbox** Open-source Virtual Infrastructure Manager ⇨project site
- **phpVirtualBox** AJAX web interface ⇨project site

https://www.virtualbox.org/

Virtualbox is supported by Oracle and comes with a full suite of documentation and other useful information

- Licensing:
  - GNU General Public License (GPL) version 2
- Documents:
  - https://www.virtualbox.org/manual/UserManual.html
- Getting VirtualBox:
  - https://www.virtualbox.org/wiki/Downloads

- Source

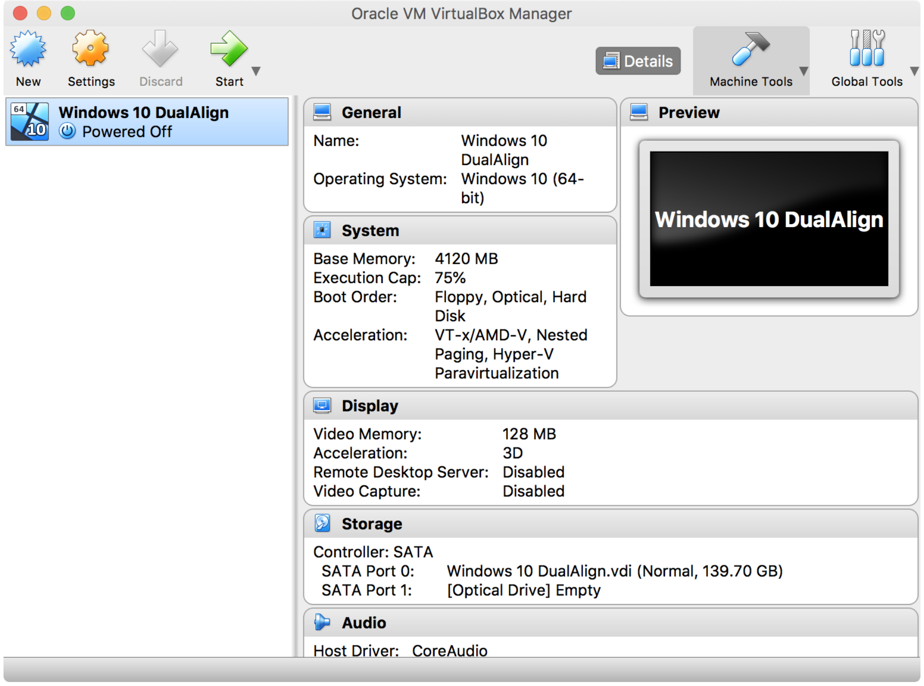
```
svn co https://www.virtualbox.org/svn/vbox/trunk vbox
```

- Contributing
  - https://www.virtualbox.org/wiki/Contributor_information
- Community
  - https://www.virtualbox.org/wiki/Community
- Issue Tracker
  - https://www.virtualbox.org/ticket/20939

- Runs on:
  - Windows, Linux, Macintosh, and Solaris
- Guest OS:
  - Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10)
  - DOS/Windows 3.x
  - Linux (2.4, 2.6, 3.x and 4.x)
  - Solaris

- Guest OS continued:
  - OpenSolaris
  - OS/2
  - OpenBSD
  - Other ... https://www.virtualbox.org/wiki/Guest_OSes
- Licensing:
  - VirtualBox is Open Source, but you still need permission for any guest operating system you use
  - In particular, Windows requires appropriate licensing

A simple interface can be used to define your virtual machine



General

Name: Windows 10 DualAlign
Operating System: Windows 10 (64-bit)

System

Base Memory: 4120 MB
Execution Cap: 75%
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, Hyper-V Paravirtualization

Display

Video Memory: 128 MB
Acceleration: 3D
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage

Controller: SATA
SATA Port 0: Windows 10 DualAlign.vdi (Normal, 139.70 GB)
SATA Port 1: [Optical Drive] Empty

Audio

Host Driver: CoreAudio

- The image can then be launched using the Virtualbox program
  - It gives a window where the entire guest operating system can be accessed
- Other buttons allow you to stop and checkpoint (save) the current state

- Be careful provisioning your image:
  - There is an inherent trade-off between image capabilities (size/memory) and VM compactness
  - That said, changing the size can be a pain
  - In general, you want to use the smallest VM that meets your needs

Containers

Concept

In a Container, the abstraction is to the application layer instead of the hardware layer:

- The application layer interacts directly with a *Container* layer
  - There is no Guest OS layer
  - The guest layer capabilities come in as shared services in the docker layer
  - The containers share the operating system kernel
- Each container runs as an application in user space

- Containers are stateless and use the existing system resources to store data and state

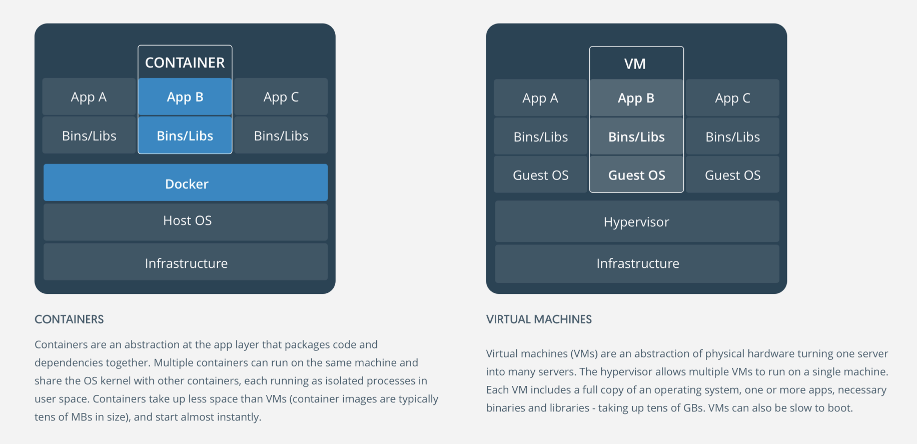
Advantages

- Containers were designed to solve some of the issues of VMs. Paticularly,
  - Large image sizes
  - High latency
  - Reduced performance
- By linking directly to the host kernel, containers:
  - Can have a reduced size
  - Can share resources
  - Can have improved performance

Disadvantages

Since containers are built from the bottom up, they can be very compact and light weight, but that means that you need to install/configure everything you want to use in your container.

Architecture



CONTAINERS

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.

VIRTUAL MACHINES

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

https://www.docker.com/what-container#/virtual_machines

https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

Docker

Docker Images

Docker maintains a library of images that you can run, i.e.:

```
$ docker images
REPOSITORY          TAG       IMAGE ID     CREATED       SIZE
python              3.5       99034fcd3ca6 2 weeks ago   917MB
python              latest    746a826ed9d7 2 weeks ago   922MB
redis               latest    f06a5773f01e 2 weeks ago   83.4MB
node                8         ed145ef978c4 2 weeks ago   673MB
postgres            9.5       a4da5bcea663 2 weeks ago   234MB
mongo               3.2       386ac407ef9f 5 weeks ago   300MB
hello-world         latest    e38bc07ac18e 3 months ago  1.85kB
ubuntu              16.04     20c44cd7596f 8 months ago  123MB
ubuntu              14.04     d6ed29ffda6b 8 months ago  221MB
fedora              latest    422dc563ca32 8 months ago  252MB
mysql               latest    5709795eeffa 9 months ago  408MB
ubuntu              latest    dd6f76d9cc90 9 months ago  122MB
node                7-alpine  4b72b56791f9 13 months ago 58.3MB
codenvy/ubuntu_jdk8 latest    4074bfc5705b 20 months ago 668MB
docker/whalesay     latest    6b362a9f73eb 3 years ago   247MB
```

You can spawn a container by running an image.

```
$ docker run -it docker/whalesay
root@40aa8aadc15f:/cowsay# cowsay squirt
 ________
< squirt >
 --------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
root@40aa8aadc15f:/cowsay# exit
```

You can find out what images you have running with

```
Wesleys-MacBook-Pro:Virtualization_Lecture wes$ docker ps
CONTAINER ID IMAGE    COMMAND     CREATED    STATUS    PORTS NAMES
eec278129a30 mongodev "/bin/bash" 5 days ago Up 5 days       zen_williams
```

You can create new images by:

1. Create a DockerFile to define your image
2. Run a docker build to turn your configuration into an image
3. Run just like a pre-canned image

4. Once you have build a docker image, it will appear when you run *docker images*
5. Alternately, you can use a yaml file to define multiple images that work together to create a system

We will explore all of these in the following example.

Example

Quick Cheat Sheet

```
## List Docker CLI commands
docker
docker container --help

## Display Docker version and info
docker --version
docker version
docker info

## Execute Docker image
docker run hello-world

## List Docker images
docker image ls

## List Docker containers (running, all, all in quiet mode)
docker container ls
docker container ls --all
docker container ls -aq
```

Example

As a class exercise, we will run through steps 1-4 of the Docker Tutorial at: https://docs.docker.com/get-started/

- Errata
  - Where the Dockerfile says 12-alpine, use 14-alpine
  - Where the Dockerfile says python2, use python3

For those of you running on WSL, there may be a few more steps:

- Follow the directions to download the Windows Desktop
- Then go to: https://docs.microsoft.com/en-us/windows/wsl/install-win10
- And Finally: https://docs.docker.com/docker-for-windows/wsl/

End