Overview and Tutorial
Virtualization and Cloud
Outline

- Overview of virtualization
- Virtualization via virtual machines
- Virtualization via Linux containers (LXC)

- Overview of cloud (cloud is NOT virtualization)
- IaaS vs PaaS vs SaaS
virtualization

- Emulation or mirroring hw/sw with hw/sw
  - For managing complexity, to facilitate isolated sharing and utilization
  - Can be exact (data center virtualization)
  - Or different (exports a system view different from underlying)

- For high-level languages: Java, C#, Python, Ruby, JS...
  - Virtual machines, virtual execution environments, managed runtimes, common language runtime

- An operating system facilitates virtualization
  - Enabling resource sharing and abstraction for processes
- Modern virtualization abstracts entire systems (hw/os)
virtualization terminology

- **Host**: underlying physical machine
  - Running zero or more virtual machines (OSx, Windows, Linux)

- **Guest**: A running virtual machine
  - Aka GuestVM, Instance

- **Virtual machine image**: a disk image (usually big) that holds all of the code/data needed to run a complete system

- **Virtual machine instance**: a running VM image
x86 virtualization

- Paravirtualization (Xen)
  - Change the OS (few thousand lines) to “help” out the hypervisor with privileged instructions
    - When there is no hardware support to help
    - Only worked/s for Linux

- Hardware assisted virtualization (VT-x, AMD-v)
  - Full virtualization of x86 (for x86 guests)
    - Without paravirtualization (no OS modification)
    - Using processor extensions
      - VM Control Structure (VMCS) – root/non-root mode in ring-0
        - Instructions behave differently in root/non-root mode
      - Hypervisor runs in root mode, guestOSs run in non-root
      - Gives guest OS direct access to system resources w/o emulation
        - Hypervisor can control guest resource access at a very fine grain

- Modern guests execute with HW virt at near native speeds
VirtualBox (VBox)

- https://www.virtualbox.org/wiki/Downloads
- An application that runs/instantiates VM images
VBox

- VBoxManage list vms
- VBoxManage list runningvms
- VirtualBox app: VM management console
- New VM:
  - CD/DVD/iso
  - Mem size
  - Disk size
  - Any OS

Vagrant

- Developer tool that automates VM management
- Automate the setup of your development environment
  - Have a base system ready to use/extend
  - Employ different OSs in the same system
  - Throw it all away and start over when bad things happen
  - Use your laptop

- Terminology
  - **Headless** VM: without a GUI (command line only)
  - Provision(ing): setting up a VM, installing packages, configuring users, etc.
  - (disk) mount: way of sharing files between host and guest
  - Vagrantfile: a file on disk in the host with instructions on how to build and configure a VM instance
**vagrant: getting started**

vagrant box add **precise** http://cloud-images.ubuntu.com/precise/server-cloudimg-amd64-vagrant-disk1.box

mkdir –p mydir

cd mydir

vagrant init **precise**

//edit Vagrantfile (first time only)

vagrant up

vagrant ssh

- Lots of different **base boxes** available:
  - https://atlas.hashicorp.com/boxes/search
  - http://www.vagrantbox.es/
  - Most boxes are 300-500MB
vagrant: getting started

- Editing the Vagrantfile

```ruby
Vagrant.configure("2") do |config|
  config.vm.box = "centos6" # set by the vagrant init parameter if specified
  config.vm.network :private_network, ip: "192.168.34.3" # private, unused IP
  # config.vm.network :public_network # if you need a public IP (access into VM)
  config.vm.provider :virtualbox do |vb|
    vb.customize ['modifyvm', :id, '--memory', "1024"]
  end
  config.vm.synced_folder "/Volumes/cjkdisk/ckrintz/images", "/root/images"
end

# you can also add puppet or chef manifests to simplify VM provisioning
```
vagrant: getting started

- **vagrant ssh**
  - Logs you in as the user `vagrant`  #uses the 10.0.X.X net
  - Sudo to root, run `ssh-keygen`, and add your host’s pub. key
    - Then you can exit out and ssh in as root
    - The IP you should use is the public/private IP in the Vagrantfile

- **Note:** if your host loses networking, sleeps, reboots bad things can happen to your VM if it is running
  
  //execute halt inside the VM (as root/sudo) or from host:
  vagrant halt       //halting persists VM state
  vagrant status     //to see status (running, aborted,poweroff)
  vagrant destroy    //this removes the VM and its state
vagrant destroy  //this does not effect the base box
//base boxes aren’t modified by VM instances that use them

• Vagrant VM instances are typically stored in ~/VirtualBox VMs/
  ▪ You should not mess with them, but if you blow away the directory with your Vagrantfile
    ▸ You can delete the files (which can take up lots of disk space)
    ▸ Names are prefixed with the base box – careful not to delete one you need
  ▪ You can also go into the VirtualBox GUI/app and delete them that way
    ▸ Right click + remove
vagrant: boxes

- Vagrant base boxes are typically stored in 
  ~/.vagrant.d/boxes/boxname
  - Added via `vagrant box add boxname dir_or_https`
  - Remove them via (use `vagrant box list //to see “provider”`)
    `vagrant box remove precise virtualbox`  //last arg is the “provider”

- Create your own boxes via
  ... `vagrant halt`
  `vagrant package`  //creates package.box from your configured VM
  //which can be added as a base box via `vagrant box add boxname package.box`
  //you can add puppet/chef manifests to these, to keep VM small
  (requires network connection upon instantiating)
Linux Containers: Light-weight virtualization

- Virtual machine alternative
- Linux host/guests only
- Requires Linux LXC
- Docker is a free tool
  - A user interface to LXC
  - LXC simplified

[Diagram showing differences between traditional virtualization and containers]

- Traditional Virtualization:
  - App
  - Guest-OS
  - Hypervisor
  - Host-OS

- Container:
  - App
  - Host-OS

- Guest 1:
  - Namespace Set 1
  - Container Management Tools

- Guest 2:
  - Namespace Set 2

- Kernel

API/ABI

P1

P2
Docker: Getting Started

• http://docs.docker.io/en/latest/installation/vagrant/

• Run it in a Vagrant VM (provided by Docker)
• Once you are able to instantiate the image, do an update on it (apt-get update; apt-get -y upgrade),
• exit and make a box of it for yourself (saves time):

• vagrant package
• vagrant box add docker package.box

• Then remove package.box
• Go to a new directory
• and start fresh:
• vagrant init docker
• //fix up the IP address (private), add more memory, mount a directory if needed
• //use this instance to play with docker
Docker: Basic Usage

- See: http://docs.docker.io/en/latest/examples/running_ssh_service/
- Note that there are a couple of bugs (missing commands, e.g. stop)
- //search for images here -- this is the public repo: https://index.docker.io/
- docker pull ubuntu //pull's a docker-ready ubuntu image version: latest
- docker pull ubuntu:12.04 //pull's a docker-ready ubuntu image version (tag)

- docker run -i -t ubuntu /bin/bash //create a container for a bash process using the ubuntu image
  //(the bash shell can fork other processes / executables)
  //note that when you exit, the process and thus the container are gone
  //along with anything you saved
Docker: Basic Usage

Install sshd in a container and save the container

docker run -i -t ubuntu /bin/bash
apt-get -y install openssh-server
passwd
//install other things if you like
exit

docker ps -a  //get the id of the process/container you just exited (top most in list)
//docker commit ID userID/name  -- ID is the id from above, userID is your userid, name is the name of the new image you are creating, ex:
docker commit b983ca83e673 ckrntz/sshd

//to remove (clean up disk space) your nonrunning containers
docker rm ID  //same ID as above

//to list your images
docker images

//to remove unneeded images
docker rmi ID  //get the ID from the command above
Docker: Basic Usage, continued

//push the container to your public repo (you need a login to index.docker.io)
docker login
docker push userID/name //same as above, must be a committed image ex:
docker push ckrintz/sshd
************************

//run your sshd container (in the background)
docker run -d -p 22 ckrintz/sshd /usr/sbin/sshd -D
//see that its running
docker ps -a

//get the port map for port 22
docker port ID 22 //ID comes from the ps command above or was output by docker run above

//ssh in
ssh root@192.168.34.4 -p 49153 //the -p arg is the port returned by the last command

//stop the running process/container
docker stop ID //use the ID from the ps command above, ex:
docker stop e5c03d1aaf12

//save any changes you made to the container
docker commit ID ckrintz/sshd
Questions on Virtualization?
Cloud Computing – is NOT virtualization

- Remote access to distributed and shared cluster resources
  - Potentially owned by someone else (e.g. Amazon, Google, ...)
    - Users **rent** a small fraction of vast resource pools, on-demand
    - Advertised service-level-agreements (SLAs), pay-per-use
  - Resources are **opaque** and **isolated**
  - Offer high availability, fault tolerance, services, & extreme scale
- Relies on OS, network, and storage virtualization/isolation
- AND lots of highly scalable, distributed systems software for fault tolerance, HA, elasticity, multitenancy, services, billing
On-demand Rental of Isolated Virtual Servers (VMs)

Infrastructure as-a-Service (IaaS)

SLAs

Web Services

Data Center

Public cloud computing

-- or --

On premises or private cloud computing

Using IaaS software

Operating System

Applications, Services, ...

Virtual Servers (VMs)

Virtualization

Server HW

Storage

Networking

Using IaaS software
Cloud Infrastructure (as a Service)
- Applications
- Runtimes
- Security & Integration
- Databases
- Servers
- Virtualization
- Server HW
- Storage
- Networking

Cloud Platform (as a Service)
- Applications
- Runtimes
- Security & Integration
- Databases
- Servers
- Virtualization
- Server HW
- Storage
- Networking

Cloud Software (as a Service)
- Applications
- Runtimes
- Security & Integration
- Databases
- Servers
- Virtualization
- Server HW
- Storage
- Networking
Amazon Elastic Compute Cloud (EC2) and Simple Storage Service (S3)

- EC2 lets you start/stop remote servers (Linux and Windows)
  - Ssh into them, sudo to root, do whatever you like
  - Charged for by the hour (rounded up) when running
  - Spot instances: lower prices but you may be evicted and you must put in the price (market system)
  - Local disk goes away upon shutdown/reboot
    - Get persistence by making a **volume** (elastic block store (EBS)) and mounting it in your instance
- S3 lets you save files/data in buckets
  - Pay for the bytes you use and the put/get requests
- UI to control it all
  - Create account
  - Key pair, security group, start/stop instances
- CLI as well
EC2 tools

ec2-create-volume --size 10 -z zone_name
ec2-describe-volumes
ec2-describe-images
ec2-describe-instances
ec2-create-group and ec2-authorize (else use default)
ec2-add-keypair cjk.key > cjk.key
chmod 600 cjk.key
ec2-describe-keypairs
ec2-run-instances --n 1 --t m1.xlarge --k cjk.key emi-ABCDEF
   //use describe-instances until it says “running”
ec2-[attach/detatch]-volume --i i-XX --d /dev/sdf vol-YY
   //ssh in, format (fdisk) , mount/umount
ec2-terminate-instances i-XX
ec2-delete-volume vol-YY
• Infrastructure-as-a-Service (IaaS)
  - Public clouds: AWS, GCE

  ➤ Private clouds:
    ➤ Eucalyptus is an open source IaaS
      ➤ For on-premise or private clouds
      ➤ Mirrors the APIs and emulates the functionality of Amazon Web Services
        - We have a UCSB cloud if you need 1+ servers, contact Chandra
Euca2ools

euca-create-volume --size 10 -z zone_name

euca-describe-volumes
euca-describe-images
euca-describe-instances

euca-create-group and ec2-authorize (else use default)
euca-add-keypair cjk.key > cjk.key
chmod 600 cjk.key
euca-describe-keypairs

euca-run-instances –n 1 –t m1.xlarge –k cjk.key emi-ABCDEF
   //use describe-instances until it says “running”

euca-[attach/detatch]-volume –i i-XX –d /dev/sdf vol-YY
   //ssh in, format (fdisk) , mount/umount

euca-terminate-instances i-XX

euca-delete-volume vol-YY
PaaS: Simplified Resource/Cloud Use

- Frees developers to focus on their innovation/science/analytics/algorithms
  - Reduces complexity

- Automates
  - Deployment & configuration of software/systems
  - Resource provisioning
  - Elastic scalability
  - Accounting, auditing
  - Authentication
Cloud Platforms / Platforms as-a-Service (PaaS)

- Capstone to reimburse up to $50; Seek trials, student offers
- Google App Engine (python, java, go)
- Azure (C# with support for other languages)
- Elastic beanstalk in AWS (java)
- Heroku (ruby/rails)

- Backends as-a-service
  - Cloud endpoints in Google Cloud Platform
  - Parse, Kumulos, Kii, Kivney, many others

- Software as-a-service
  - Google bigquery, analytics, maps, SQL/NoSQL ...
  - AWS RDS, Redshift (SQL); elastic map reduce
  - **Hosted** services for databases, datastores (as-a-service)