Cluster and Cloud Computing – Lecture 5
Cloud Computing & Getting to Grips with NeCTAR

Professor Richard O. Sinnott *& Farzad Khodadadi*

30th March 2017

rsinnott@unimelb.edu.au

Assignment 1

- Access?
  - Googledocs for anyone left?
  - ssh "yourkaraageusername"@spartan2.hpc.unimelb.edu.au
  - Password: yourkaraagepassword
  - *Not your UniMelb username/password*
- MPI
  - module load OpenMPI/1.10.2-GCC-6.2.0 (or similar)
    - vs
  - module load Python/3.4.3-goolf-2015a (mpi4py)
- Solution only needs to work on the data given

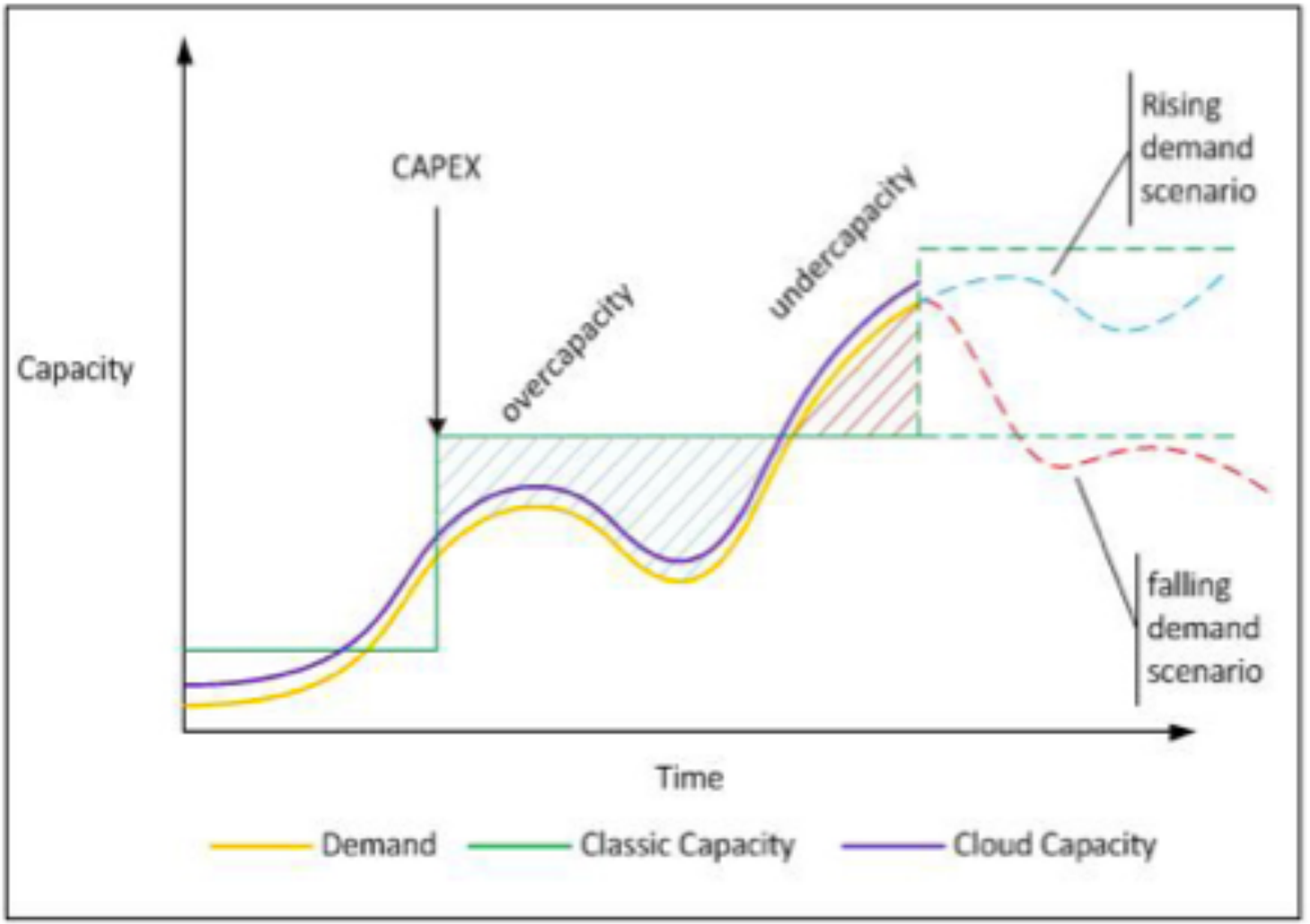
Assignment 1

- Pseudo solution…
  - Preprocessing
    - JSON and downsize data
  - Master/slave configuration
    - Master solely responsible for *reading/processing lines* in the file and then broadcasting to workers
    - Avoid loading whole file into memory (especially for bigTwitter)
      - Process line by line
        - » You can have your own libraries installed/configured in your environment (just for you)
      - Pre-existing knowledge of file size and #cores can be used
  - At end of master's processing, master node requests slaves to return counts for final processing
  - Key to solution is data structures
    - e.g. key:values, hashmaps, …
  - Re-use libraries that help…
    - JSON, REGEX, …

Overview

- Cloud benefits
  - Cloud marketing!?
- The various flavours of cloud computing
  - Introduction to #aaS?
- Break
- Demonstration of NeCTAR Research Cloud
- Break
- Scripting the Cloud

Life before cloud computing

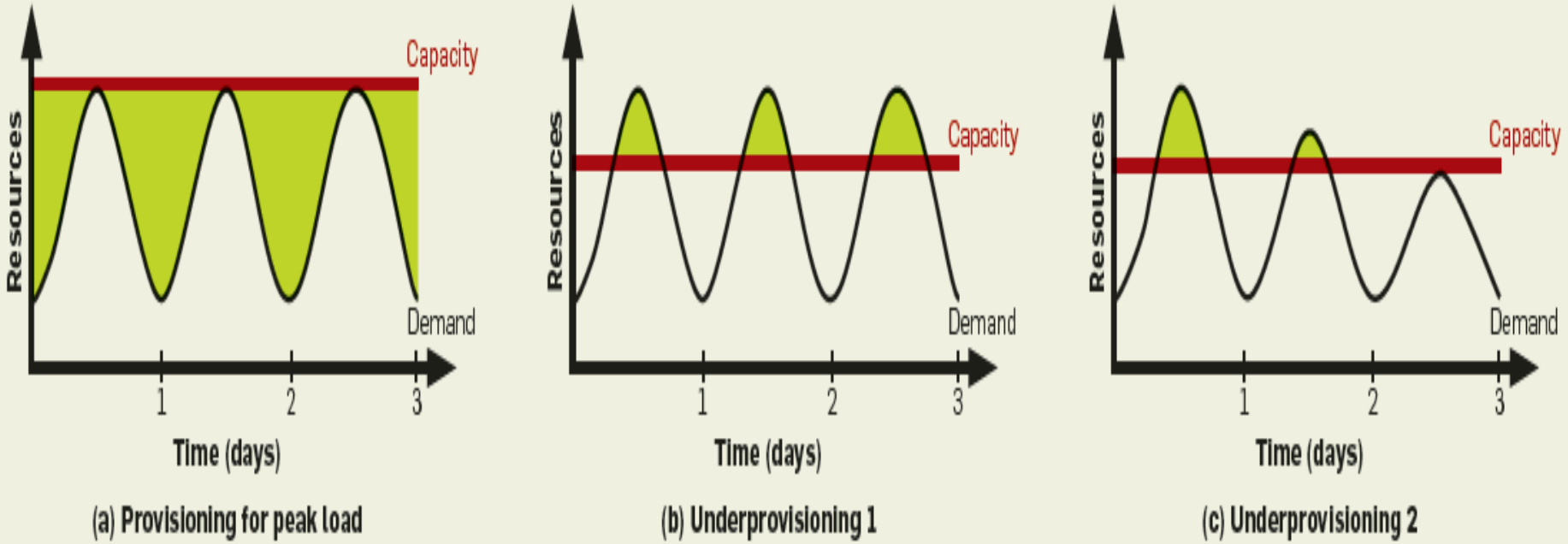


Capacity vs Utilization curves [8]

source: http://www.rackspace.com/knowledge_center/whitepaper/revolution-not-evolution-how-cloud-computing-differs-from-traditional-it-and-why-it

Life before cloud computing…ctd

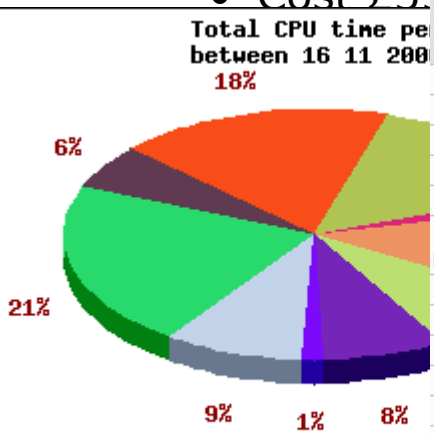
Figure 2. (a) Even if peak load can be correctly anticipated, without elasticity we waste resources (shaded area) during nonpeak times. (b) Underprovisioning case 1: potential revenue from users not served (shaded area) is sacrificed. (c) Underprovisioning case 2: some users desert the site permanently after experiencing poor service; this attrition and possible negative press result in a permanent loss of a portion of the revenue stream.



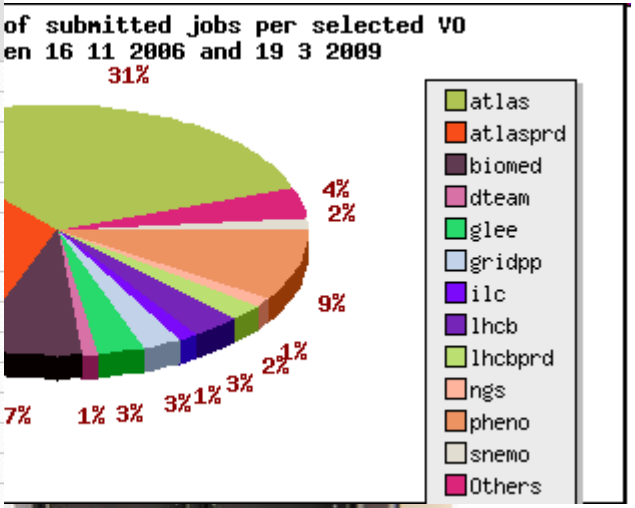
source: Ambrust et al, doi:10.1145/1721654.1721672

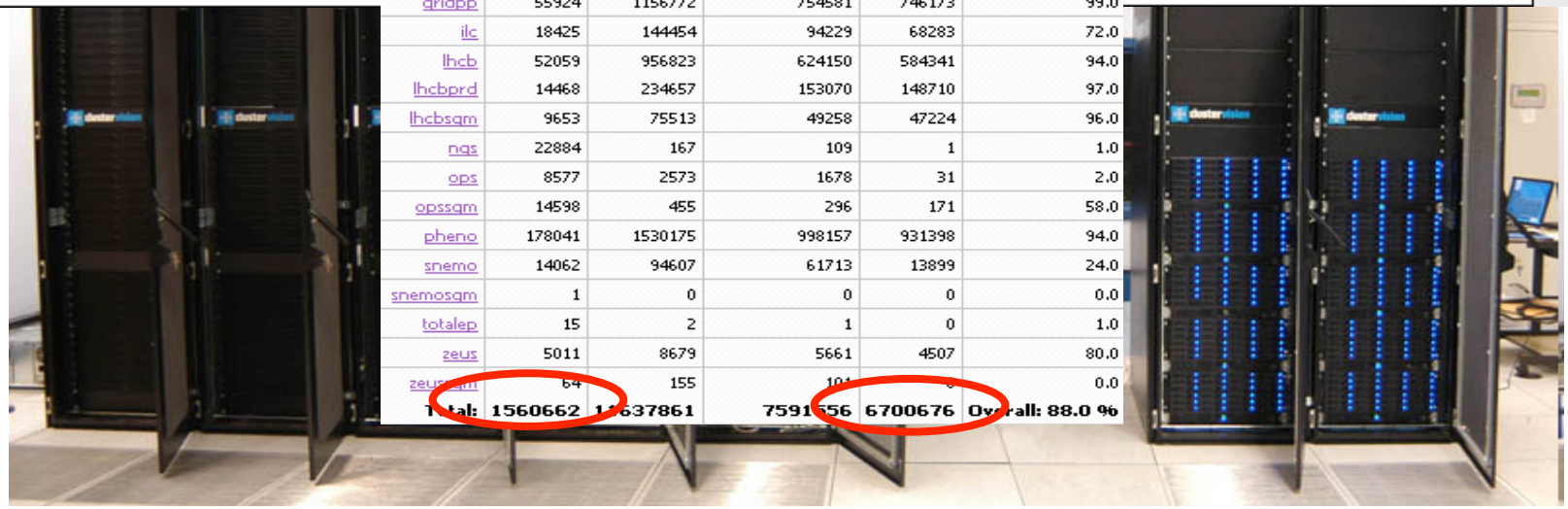
Cloud-busting? (~2009)

- To buy or not to buy that is the question…?
  - ScotGrid (www.scotgrid.ac.uk)
    - Cost £5



Group	Jobs	Kilo SPEC hours	WallClock Hours	CPU Hours	Eff (%)
alice	5	1	1	0	1.0
atlas	466733	2179463	1421698	1050451	74.0
atlasprd	497669	1808452	1179682	1091466	93.0
atlassgm	6420	930	606	465	77.0
babar	2	0	0	0	1.0
biomed	106413	1023678	667761	458138	69.0
biomedsgm	15	0	0	0	5.0
camont	9164	35510	23164	20070	87.0
cms	4857	18557	12105	6686	55.0
cmsprd	677	365	238	138	58.0
dteam	16494	3066	2000	344	17.0
dzero	92	141	92	75	82.0
glbio	33	694	453	446	99.0
glee	58306	2361972	1540752	1527659	99.0
gridpp	55924	1156772	754581	746173	99.0
ilc	18425	144454	94229	68283	72.0
lhcb	52059	956823	624150	584341	94.0
lhcbprd	14468	234657	153070	148710	97.0
lhcbsgm	9653	75513	49258	47224	96.0
ngs	22884	167	109	1	1.0
ops	8577	2573	1678	31	2.0
opssgm	14598	455	296	171	58.0
pheno	178041	1530175	998157	931398	94.0
snemo	14062	94607	61713	13899	24.0
snemosgm	1	0	0	0	0.0
totalep	15	2	1	0	1.0
zeus	5011	8679	5661	4507	80.0
zeussgm	64	155	101	0	0.0
Total:	1560662	11637861	7591556	6700676	Overall: 88.0 %





Cloud-busting? (~2009)

On-Demand Instances

United States | **Europe**

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
1.7Gb memory/160GB disk	$0.11 per hour	$0.135 per hour
7.5Gb memory/850GB disk	$0.44 per hour	$0.54 per hour
15Gb memory/1690GB disk	$0.88 per hour	$1.08 per hour
High CPU On-Demand Instances	Linux/UNIX Usage	Windows Usage
Medium	$0.22 per hour	$0.32 per hour
Extra Large	$0.88 per hour	$1.28 per hour

Amazon Elastic Block Store

United States | **Europe**

Amazon EBS Volumes

- $0.11 per GB-month of provisioned storage
- $0.11 per 1 million I/O requests

Amazon EBS Snapshots to Amazon S3 (priced the same as Amazon S3)

- $0.18 per GB-month of data stored
- $0.012 per 1,000 PUT requests (when saving a snapshot)
- $0.012 per 10,000 GET requests (when loading a snapshot)

Data Transfer

Internet Data Transfer

The pricing below is based on data transferred "in" and "out" of Amazon EC2.

Data Transfer In	
All Data Transfer	$0.10 per GB

Data Transfer Out	
First 10 TB per Month	$0.17 per GB
Next 40 TB per Month	$0.13 per GB
Next 100TB per Month	$0.11 per GB
Over 150 TB per Month	$0.10 per GB

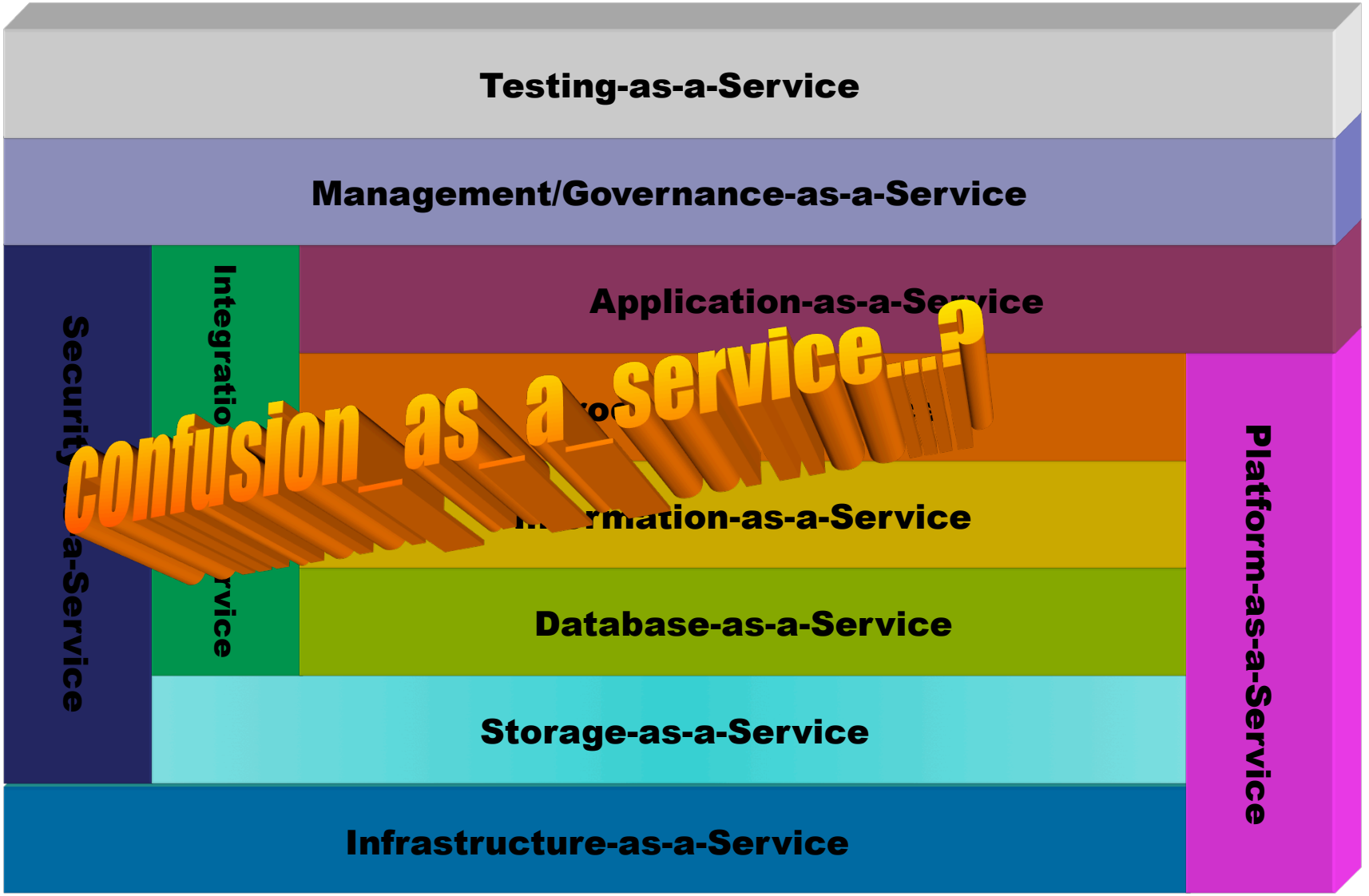
- $1=£0.69 (back then!)
  - £0.30*6,700,676 CPU hours
    - = £2,010,202 for just compute on-demand + data + networking + …?
- Now…???
  - https://aws.amazon.com/ec2/pricing/

Cloud Computing: A Definition

- NIST definition: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."
  - » National Institute of Standards and Technology (http://dx.doi.org/10.6028/NIST.SP.800-145)

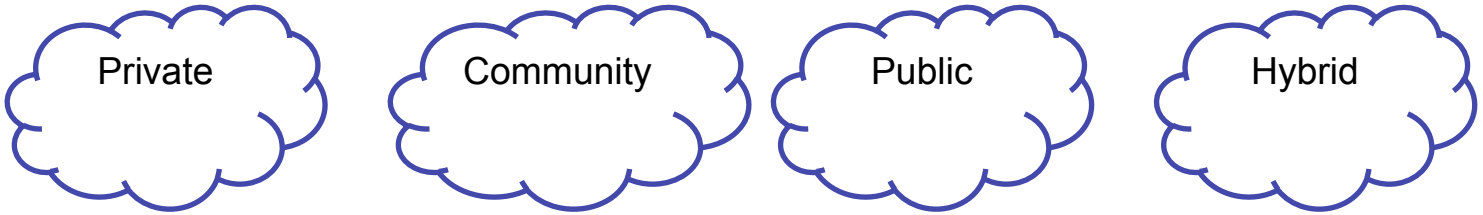
- Focus of today is to get you up and running on the Cloud and explore the technologies related to the underlined
  - Later lecture will do compare/contrast with AWS

Organizing the Clouds



The Most Common Cloud Models

Deployment Models



Delivery Models

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

Essential Characteristics

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Public Clouds

- Pros
  - Utility computing
  - Can focus on core business
  - Cost-effective
  - “Right-sizing”
  - Democratisation of computing
- Cons
  - Security
  - Loss of control
  - Possible lock-in
  - Dependency of Cloud provider continued existence

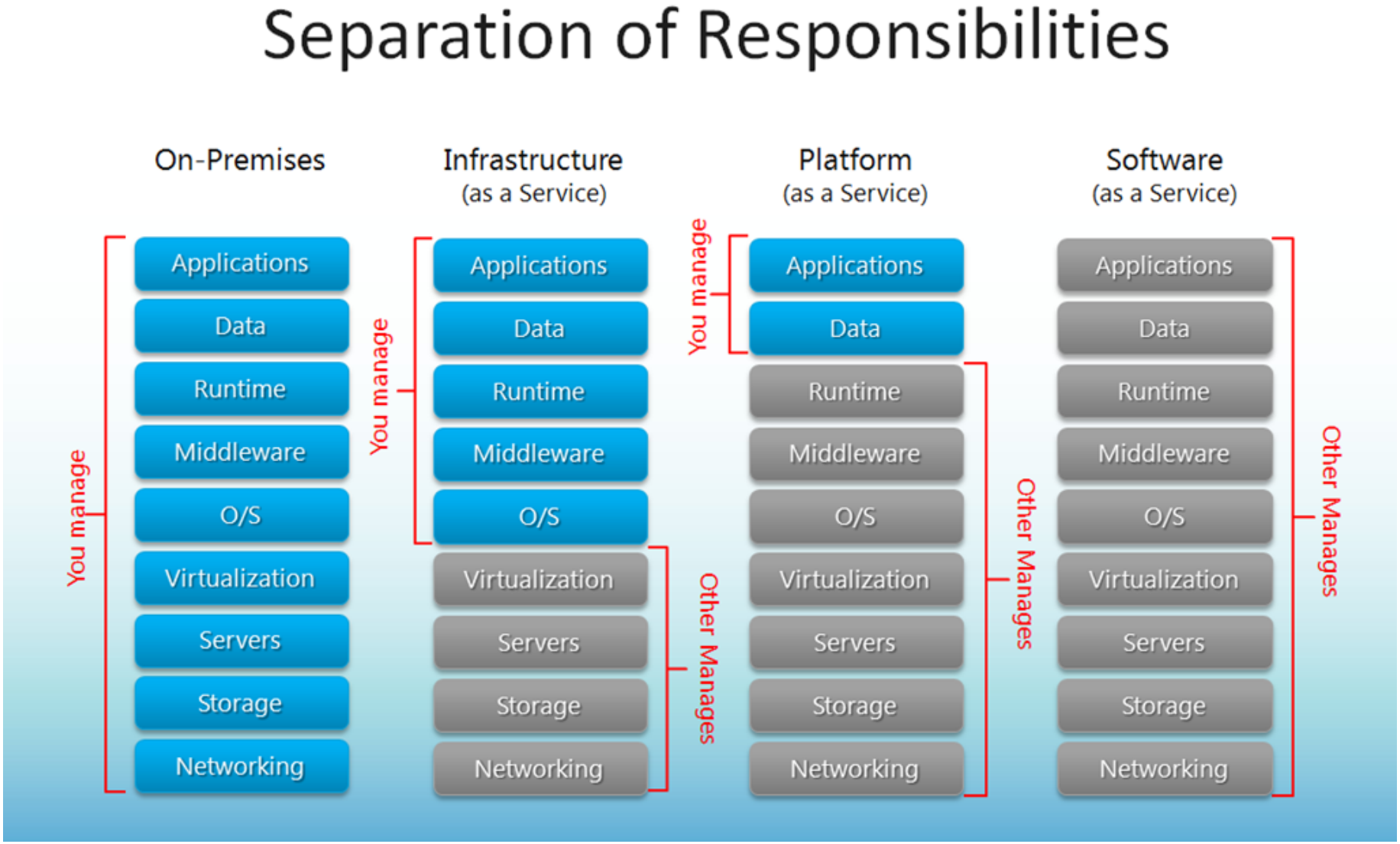
Private Clouds

- Pros
  - Control
  - Consolidation of resources
  - Easier to secure
  - More trust
- Cons
  - Relevance to core business?
    - e.g. Netflix moved to Amazon
  - Staff/management overheads
  - Hardware obsolescence
  - Over/under utilisation challenges

Hybrid Clouds

- Examples
  - Eucalyptus, VMWare vCloud Hybrid Service
- Pros
  - Cloud-bursting
    - Use private cloud, but burst into public cloud when needed
- Cons
    - How do you move data/resources when needed?
    - How to decide (in real time?) what data can go to public cloud?
    - Is the public cloud compliant with PCI-DSS (Payment Card Industry – Data Security Standard)?

Delivery Models



Public SaaS examples

- Gmail
- Sharepoint
- Salesforce.com CRM
- On-live
- Gaikai
- Microsoft Office 365
- Some definitions include those that do not require payment, e.g. ad-supported sites

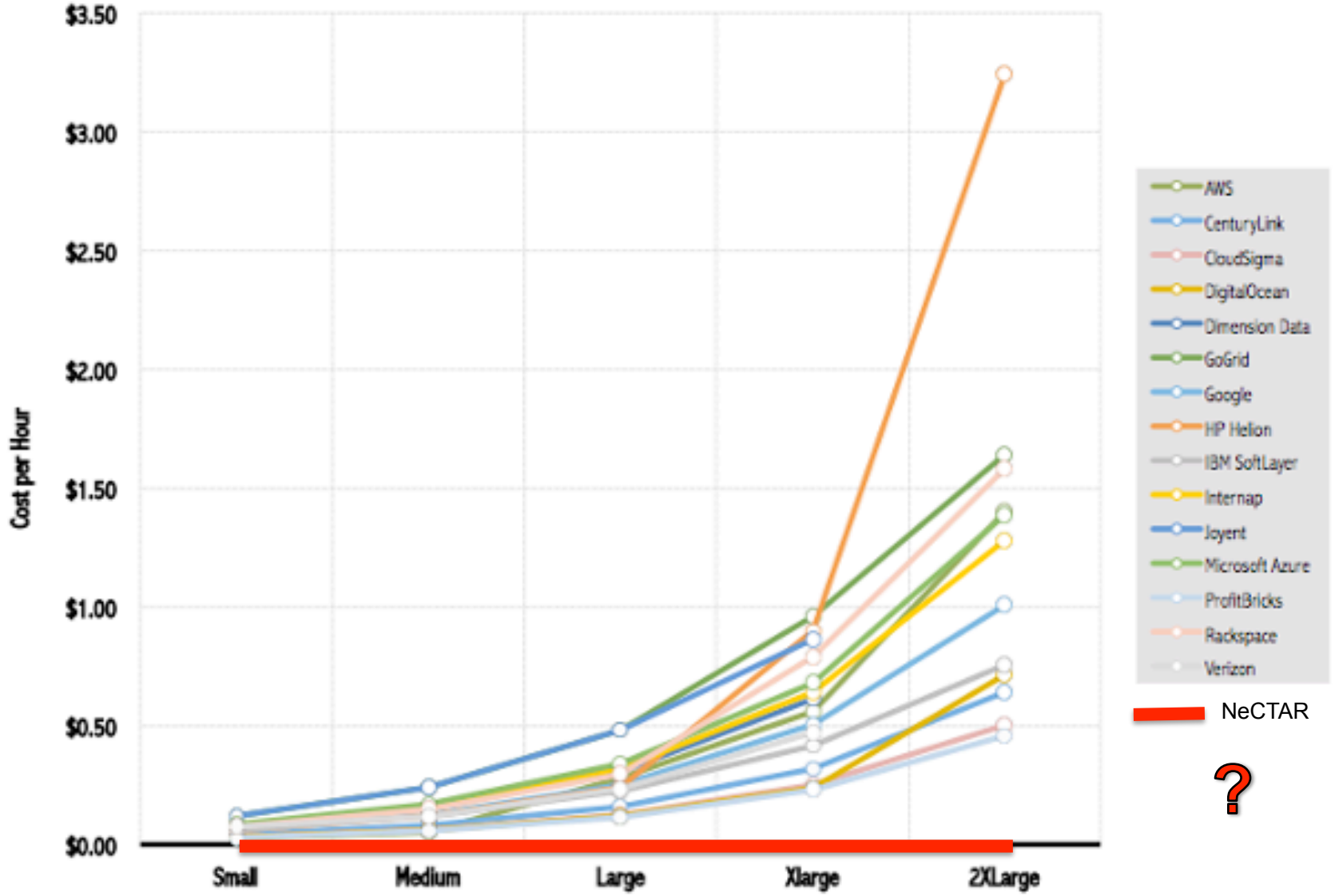
Public PaaS Examples

Cloud Name	Language and Developer Tools	Programming Models Supported by Provider	Target Applications and Storage Options
Google App Engine	Python, Java, Go, PHP + JVM languages (scala, groovy, jruby)	MapReduce, Web, DataStore, Storage and other APIs	Web applications and BigTable storage
Salesforce.com's Force.com	Apex, Eclipsed-based IDE, web-based wizard	Workflow, excel-like formula, web programming	Business applications such as CRM
Microsoft Azure	.NET, Visual Studio, Azure tools	Unrestricted model	Enterprise and web apps
Amazon Elastic MapReduce	Hive, Pig, Java, Ruby etc.	MapReduce	Data processing and e-commerce
Aneka	.NET, stand-alone SDK	Threads, task, MapReduce	.NET enterprise applications, HPC

Infrastructure As A Service (IaaS)

- Primary focus of this course…
- Many providers
  - Amazon Web Services (Market leader)
    - http://aws.amazon.com
  - Oracle Public Cloud
    - https://cloud.oracle.com/
  - Rackspace Cloud
    - www.rackspace.com
  - *CenturyLink, CloudSigma, DigitalOcean, DimensionData, GoGrid, Helio, Internap, Joyent, ProfitBricks, Verizon, …*
  - NeCTAR/Openstack Research Cloud
    - www.nectar.org.au

Cost Comparison



http://connect.cloudspectator.com/iaas-price-comparison-2015

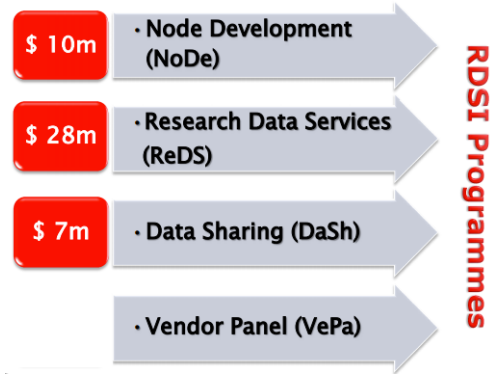
NeCTAR Research Cloud

- National eResearch Collaboration Tools and Resources (NeCTAR – www.nectar.org.au)
  - $50m+$10m+… federal funding
  - Lead by University of Melbourne
  - Had four key strands
    - ~~National Servers Program~~
    - Research Cloud Program
      - OpenStack IaaS
      - 4Gb-64Gb (linux flavours)
      - 30,000 physical servers available across eight availability zones
    - ~~eResearch Tools Program~~
    - Virtual Laboratories Program
      - Astro,
      - Genomics,
      - Humanities,
      - Climate,
      - Nano-,
      - …endocrine genomics

Data Infrastructures

- Research Data Services (RDS)
- $50m+$10m project to establish data storage resources across Australia
  - ~100 Petabytes national data
  - UniMelb, UniMonash for Vic-wide "nationally significant data sets"
- Used by many diverse communities

Sustainability/longevity?



Common Terms

- Machine Image: A stored image/template from which a a new virtual machine can be launched, e.g. Ubuntu
- Instance: A running virtual machine based on some machine image.
- Volume: Attachable Block Storage, which is the equivalent of a virtual disk drive.
- Object Store: A large store for storing simple binary objects + metadata within containers
- Security Groups: A means of specifying firewall rules
- Key-pairs: Public/private key pairs for accessing a virtual machine

NeCTAR Research Cloud

- Based on OpenStack
  - Open source cloud technology
- Many associated/underpinning services
  - Compute Service (code-named Nova)
  - Image Service (code-named Glance)
  - Block Storage Service (code named Cinder)
  - Object Storage Service (code-named Swift)
  - Security Management (code-named Keystone)
  - Orchestration Service (code-named Heat)
  - Network Service (code-named Neutron)
  - Metering Service (code-named Ceilometer)
  - …

http://docs.openstack.org/cli-reference/

NeCTAR Research Cloud Instance Types

Instance Type	VCPUs	Memory (GB)	Instance Storage (GB)
m1.small	1	4	10GB + 1 x 30
m1.medium	2	8	10GB + 1 x 60
m1.large	4	16	10GB + 1 x 120
m1.xlarge	8	32	10GB + 1 x 240
m1.xxlarge	16	64	10GB + 1 x 480

x4

+100Gb Object Store

+250Gb Volume Store

BREAK

NeCTAR Demo (Crib Sheets LMS)

- Launching a new VM
- Connecting to VM via ssh
  - ssh –i <key> hostname
- Installing an application
- Setting up security groups
  - Communicating between trusted nodes
- Attaching a Volume
  - Formatting
  - Mounting
- Creating a snapshot
- Uploading a file to the object store

Basic recipes to follow…

- ssh-keygen –t rsa –f cloud.key (on Unix/Linux/MacOS or see Putty for Windows)
- chmod 600 cloud.key
- ssh –i cloud.key ubuntu@<instance_ip> (Ubuntu)
- ssh –i cloud.key ec2_user@<instance_ip> (Amazon Linux, RHEL)
- sudo apt-get install apache2
  - apt-get installs/removes packages on Ubuntu installations
  - sudo = runs commands as another user (e.g. superuser)
- sudo fdisk –l
  - fdisk for partitioning disk
- sudo mkfs.ext4 /dev/vdb
  - mkfs = make file system
  - ext4 = type of file system (ext2, ext3, ext4)
    - See http://www.thegeekstuff.com/2011/05/ext2-ext3-ext4/ for details on differences
- cd /mnt
  - where volumes get attached
- sudo mkdir myVolume
- sudo mount /dev/vdb /mnt/myVolume
- cd myVolume
- df –h
  - Provides summary of drive space use
- Other useful commands rsync
  - File difference copies, …

BREAK

Automation

- Deploying complex cloud systems require a lot of moving parts
  - Easy to forget what software you installed, and what steps you took to configure system
  - Might be non-repeatable
  - Snapshots are monolithic – provides no record of what has changed
- Automation:
  - Provides a record of what you did
  - Codifies knowledge about system
  - Makes process repeatable
  - Makes it programmable – "Infrastructure as code"

Classification of Scripting tools

- Cloud-focussed
  - Apache JClouds (Java - supports multi-cloud)
  - Boto (Python – supports AWS and Openstack)
- Shell scripts
  - Bash
  - Perl
- Automation and configuration management tools
  - Chef (uses Ruby for creating cookbooks)
  - Puppet (uses its own configuration language)
  - Ansible (use YAML to express playbooks)
  - Fabric (Python library that uses SSH for application deployment and administration tasks)

Scripting Clouds with Boto

- Scripting Basic Setup of Cloud Resources
  - Launching an instance
  - Listing instances
  - Listing snapshots
  - Terminating an instance
  - Uploading data to a bucket using the swift client

- Installing Boto
  - On Ubuntu, use the following command and make sure python is already installed:

    sudo apt-get update && sudo apt-get install python-boto

Boto example

```
import boto
from boto.ec2.regioninfo import RegionInfo

region=RegionInfo(name='melbourne', endpoint='nova.rc.nectar.org.au')

ec2_conn = boto.connect_ec2(aws_access_key_id=<YOUR_ACCESS_KEY>,
aws_secret_access_key=<YOUR_SECRET_KEY>, is_secure=True,
region=region, port=8773, path='/services/Cloud', validate_certs=False)
```

Boto Scripting: List images

Example:

```
images = ec2_conn.get_all_images()
for img in images:
    print 'id: ', img.id, 'name: ', img.name
```

Boto Scripting: Launch image

Format:

ec2_conn.run_instances(<ami_id>, key_name=<key_name>,
instance_type=<instance_type>,
security_groups=[’<security_group>'])

Example:

ec2_conn.run_instances('ami-000007b9',
key_name=’your_key_pair', instance_type='m1.small',
security_groups=[’your_security_group'])

Boto Scripting: List instances

Get reservations:

```
reservations = ec2_conn.get_all_reservations()
```

Show reservation details:

```
for idx, res in enumerate(reservations):
    print idx, res.id, res.instances
```

Show instance details:

```
print reservations[0].instances[0].ip_address
print reservations[0].instances[0].placement
```

Boto Scripting: Terminate Instance

Format:

ec2_conn.terminate_instances(instance_ids=[…])

Example:

ec2_conn.terminate_instances(instance_ids=[‘i-1002a37c’])

Boto Scripting: Using Volumes

Request volume:

```
vol_req = conn.create_volume(50, ”monash-01”)
```

Check provisioning status:

```
curr_vol = conn.get_all_volumes([vol_req.id])[0]
print curr_vol.status
print curr_vol.zone
```

Attach volume:

```
conn.attach_volume (vol.id, inst.id, "/dev/vdc")
```

Boto Scripting: Creating Snapshots

Create a snapshot:

```
snapshot = conn.create_snapshot(vol.id, 'My snapshot')
```

Create a volume based on the snapshot:

```
new_vol = snapshot.create_volume('monash-01')
conn.attach_volume (vol.id, inst.id, "/dev/vdd")
```

Delete snapshot:

```
conn.delete_snapshot(snapshot.id)
```

Boto Scripting: Connect to Object Store

Amazon:

```
s3_conn = boto.connect_s3(
        aws_access_key_id='<access_key>',
        aws_secret_access_key='<secret_key>')
```

Nectar: Using S3 compatible API:

```
s3_conn =  boto.connect_s3(
        aws_access_key_id='<access_key>',
        aws_secret_access_key='<secret_key>',
        is_secure=True, host='swift.rc.nectar.org.au',
        port=8888, path='/')
```

Boto Scripting: List Buckets

Example:

```
buckets = s3_conn.get_all_buckets()
for bucket in buckets:
    print bucket.name
```

Boto Scripting: Storing Data via API

Setting a key-value pair:

```
from boto.s3.key import Key
k = Key(bucket)
k.key = ’my_key’
k.set_contents_from_string(’hello world')
```

Getting a key-value pair:

```
k = Key(bucket)
k.key = ’my_key’
k.get_contents_as_string()
```

Get/set content from file:

```
k.set_contents_from_filename(’hello.jpg’)
k.get_contents_to_filename(‘world.jpg')
```

Storing Data via Swift Client

1. Download openstack RC file:

In dashboard -> Access & Security -> API Access -> Download Openstack RC file

2. Execute file in current shell environment

source tenancy-openrc.sh

3. Get/set content from file:

swift upload <bucket_name> <file_name>

Ansible

- An automation tool for configuring and managing computers
  - Finer grained set up and configuration of software packages
- Initial release: Feb. 2012
- Combines Multi-node software deployment
- Ad-hoc task execution and configuration management
  - Configuring thousands of machines manually!?

Ansible: Structure

- Ansible scripts are called playbooks
- Scripts written as simple YAML files
- Structured in a simple folder hierarchy

```
Playbook folder
|- variables
|    |_ vars
|- inventory
|    |_ inventory.ini
|- roles
|    |- files
|    |- tasks
|    |   |- task1.yml
|    |   |_task2.yml
|    |_ templates
|_ playbook.yml
```

Ansible: Inventory

- Contains a list of machines to act upon
- Can be grouped
- Stored in .ini format

```
[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

Ansible: Playbooks

- Executed sequentially from a yaml file

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service: name=httpd state=started
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Ansible Demo

- Simple playbook to install Apache
- Required steps:
  - Install the Ansible client locally
  - Create and configure a fresh VM in NeCTAR
  - Setup the host (inventory) file
  - Test ssh connection
  - Create apache installation playbook
  - See the magic happen!

Ansible: Features

- Easy to learn
  - Playbooks in YAML, Templates in Jinja2, Inventory in INI file
  - Sequential execution
- Minimal requirements
  - No need for centralised management servers/daemons
  - Single command to install (*pip install ansible*)
  - Uses SSH to connect to target machine
- Idempotent:
  - Executing N times no different to executing once.
  - Prevents side-effects from re-running scripts
- Extensible:
  - Write your own modules

Ansible: More Features

- Supports push or pull
  - Push by default but use cron job to make it pull
- Rolling updates
  - Useful for continuous deployment/zero downtime deployment
- Inventory management
  - Dynamic inventory from external data sources
  - Execute tasks against host patterns
- Ansible Vault for encrypted data
- Ad-hoc commands
  - When you just need to execute a one-off command against your inventory
    - e.g. ansible -i inventory_file -u ubuntu -m shell -a “reboot”
- Ansible Tower: Enterprise mission control for Ansible (Dashboard, System Tracker, etc)

Ansible: Demo 2

- A role is a reusable set of tasks
  - Can be published in Ansible Galaxy - a centralised repository of roles
    - https://galaxy.ansible.com/
- Building playbooks out of roles
  - How to structure a playbook
  - How to define variables
  - How variables are used in templates
  - How to fix simple errors

References

- Hwang, Dongarra & Fox, 2011. Distributed and Cloud Computing, 1st Edition. Elsevier.
- Armbrust et al., 2010. A view of cloud computing. Communications of the ACM 53, 50-58. http://doi.acm.org/10.1145/1721654.1721672
- Revolution Not Evolution: How Cloud Computing Differs from Traditional IT and Why it Matters http://www.rackspace.com/knowledge_center/whitepaper/revolution-not-evolution-how-cloud-computing-differs-from-traditional-it-and-why-it
- The 10 Most Important Companies In Cloud Computing http://www.businessinsider.com.au/10-most-important-in-cloud-computing-2013-4?op=1#a-word-about-clouds-1
- Boto, https://github.com/boto/boto
- Ansible, https://www.ansible.com/

References

- Chef https://www.chef.io/
- Fabric http://www.fabfile.org/
- Puppet https://puppetlabs.com
- JClouds https://jclouds.apache.org/