



9-cloud-computing - Soft cloud computing file

soft computing (Savitribai Phule Pune University)



ROYAL INSTITUTE
OF TECHNOLOGY

Introduction to Cloud Computing

ID2210

Jim Dowling

Cloud Computing

- **Cloud computing** is the delivery of hosting services that are provided to a client over the Internet.
 - Enable large-scale services without up-front investment.

WHERE THE HECK
IS MY DATA?



ITS THERE, UP
IN THE CLOUDS.



Brainstuck.com

THE #1 PROGRAMMER EXCUSE
FOR LEGITIMATELY SLACKING OFF:

~~"MY CODE'S COMPILING."~~
VM'S LAUNCHING



[XKCD Comic 303]

Clouds are Elastic

- NIST Definition of Cloud Computing

"Cloud computing is a model for enabling 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."

Supporting Technologies

- Enormous computer data-centres containing **commodity hardware**.
- **Virtualization** of computation, storage, and communication.
 - Turn hardware and networking into software!
- Achieve **economies of scale**.
 - Reduce costs of electricity, bandwidth, hardware, software and use low-cost locations.
 - Lower-cost than provisioning own hardware.
- **Large-scale distributed systems services**, such as NoSQL datastores, object stores, and distributed filesystems, have enabled developers to build scalable cloud computing applications.

Cloud Computing Essentials

- Cloud computing is Utility Computing
 - Cloud services are controlled and monitored by the cloud provider through a pay-per-use business model.
- An ideal cloud computing platform is:
 - efficient in its use of resources
 - scalable
 - elastic
 - self-managing
 - highly available and accessible
 - inter-operable and portable

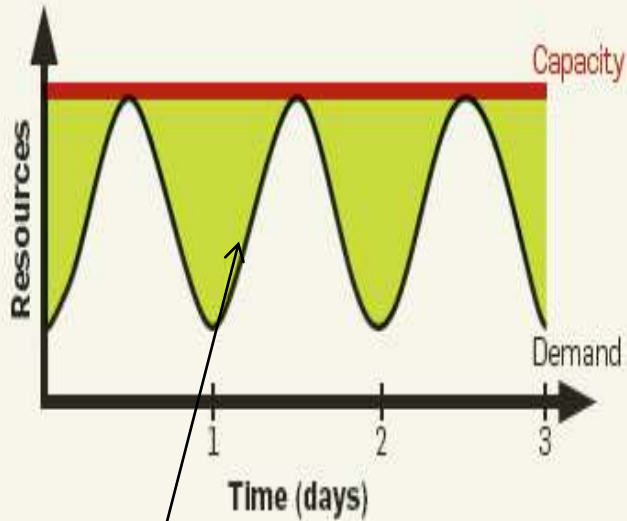
Cloud Properties

- **Resource efficiency:** computing and network resources are pooled to provide services to multiple users. Resource allocation is dynamically adapted according to user demand.
- **Elasticity:** computing resources can be rapidly and elastically provisioned to scale up, and released to scale down based on consumer's demand.

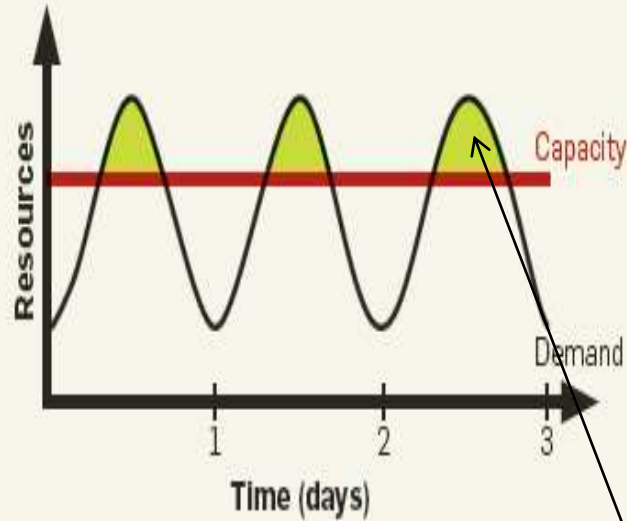
Cloud Properties

- **Self-managing services:** a consumer can provision cloud services, such as web applications, server time, processing, storage and network as needed and automatically without requiring human interaction with each service's provider
- **Accessible and highly available:** cloud resources are available over the network anytime and anywhere and are accessed through standard mechanisms that promote use by different types of platform (e.g., mobile phones, laptops, and PDAs).

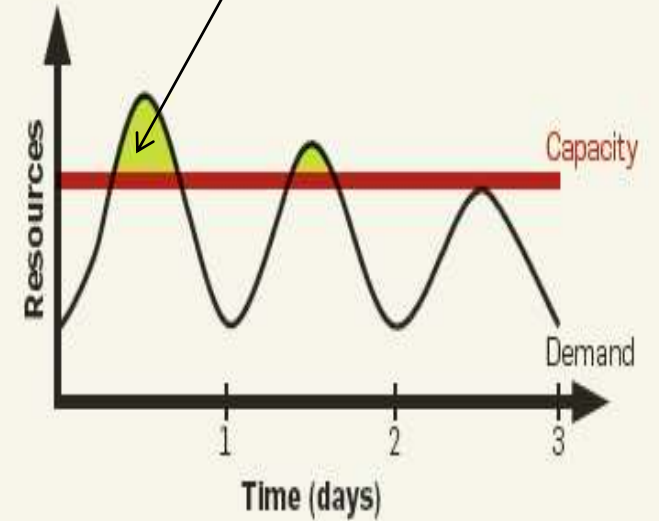
Over or Under-Provisioning



(a) Provisioning for peak load



(b) Underprovisioning 1



(c) Underprovisioning 2

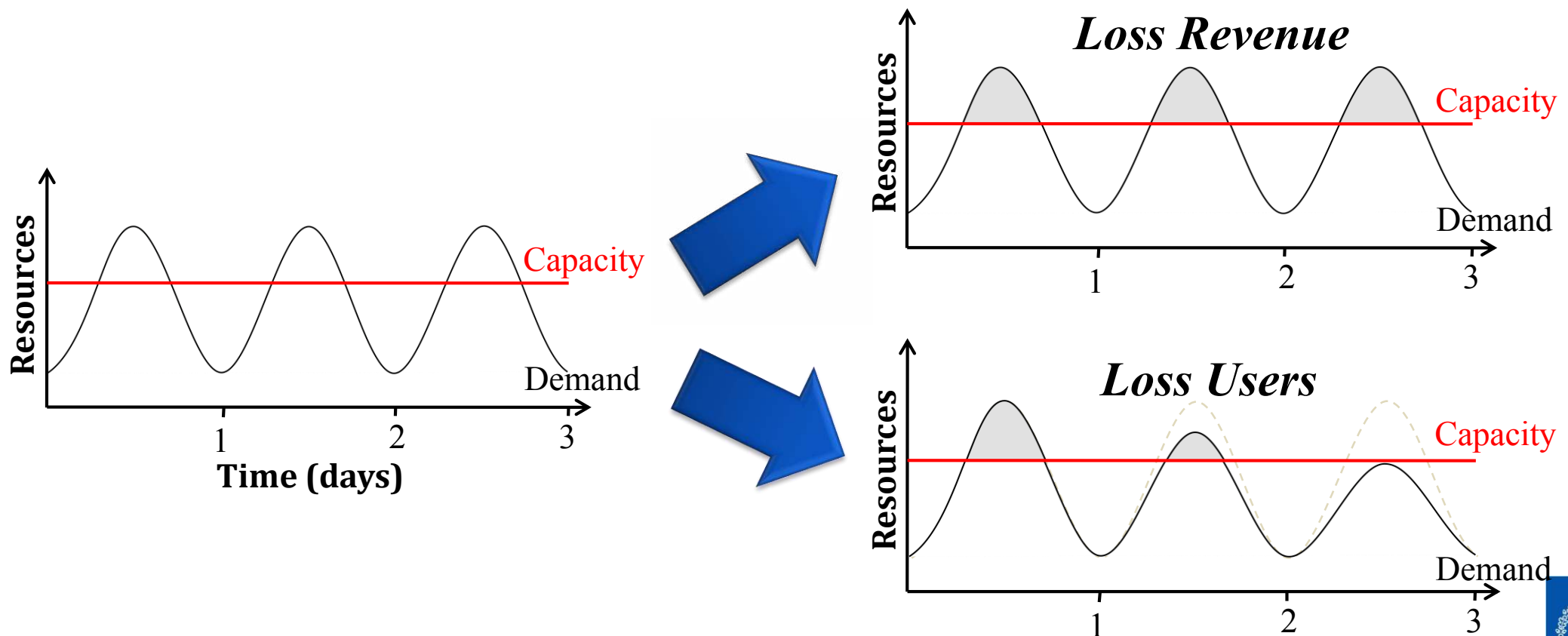
Less and
less
demand.

Shaded area is unused capability.

Shaded area represents requests not served.

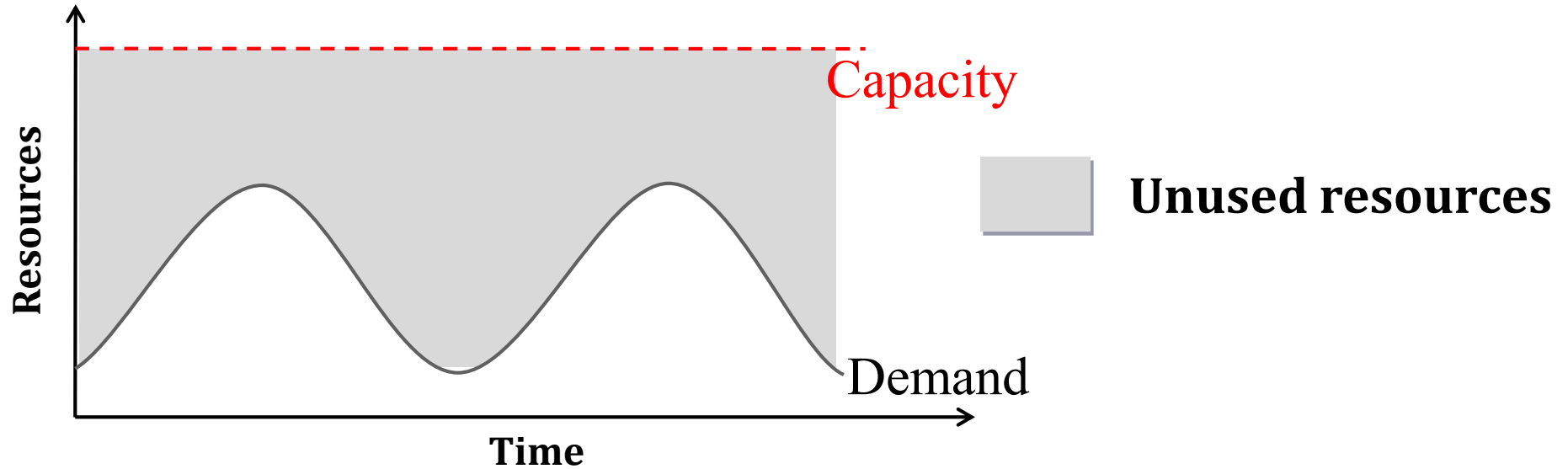
Dynamic Provisioning

- In traditional computing model, two common problems :
 - Underestimate system utilization which result in under provision



Dynamic Provisioning

- Overestimate system utilization which results in low utilization



- How do we solve this problem?
 - Dynamically provision resources

Real world estimates

- Average server utilization is 5% to 20%.
- Peak workload exceeds the average by factors of 2 to 10.
- Users provision for the peak.
- Peak loads may occur based on the time of day or based on other factors (e.g. photo sharing after the holidays, drop/add within two weeks of start of term, etc.)

Public Clouds, Private Clouds

This document is available free of charge on

StuDocu.com

Downloaded by Rahul Vashistha (rahulvashistha1412@gmail.com)

Deployment Model

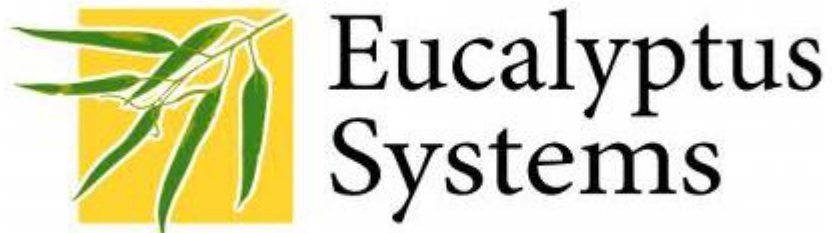
- There are four primary cloud deployment models :
 - Public Cloud
 - Private Cloud
 - Community Cloud
 - Hybrid Cloud

Public Clouds

- Public clouds are owned by cloud service providers who charge for the use of cloud resources.
- Basic characteristics:
 - Homogeneous infrastructure, Common policies
 - Shared resources and multi-tenancy
 - Leased or rented infrastructure
 - Economies of scale
- **AWS/EC2** (Amazon)
- **Azure** (Microsoft)
- **Google Cloud Platform.**
- **Rackspace.**

Private Clouds

- The cloud infrastructure belongs to and is operated by only one organization.
- Basic characteristics :
 - Heterogeneous infrastructure; Customized policies
 - Dedicated resources
 - In-house infrastructure; End-to-end control
- Examples include:



OpenNebula



Other types of Clouds

- Community cloud

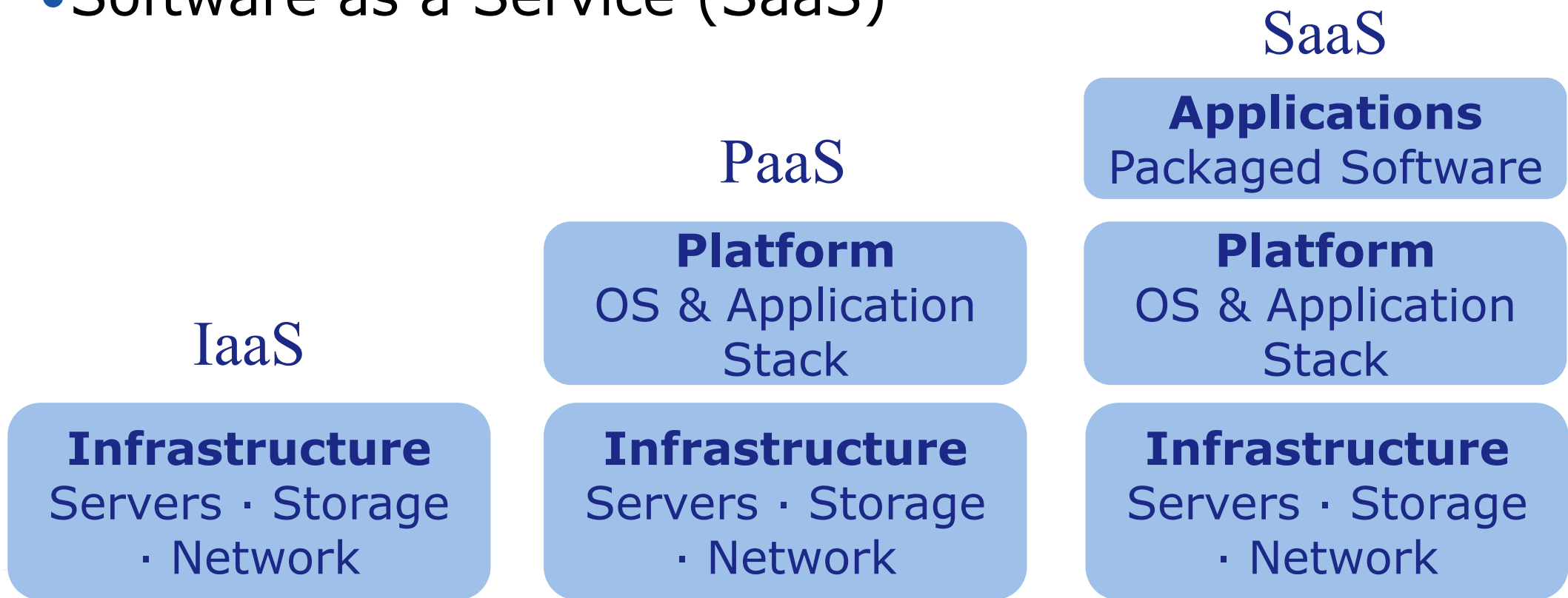
- The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations).

- Hybrid cloud

- The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.

IaaS, PaaS and SaaS

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



Spectrum of Cloud Users

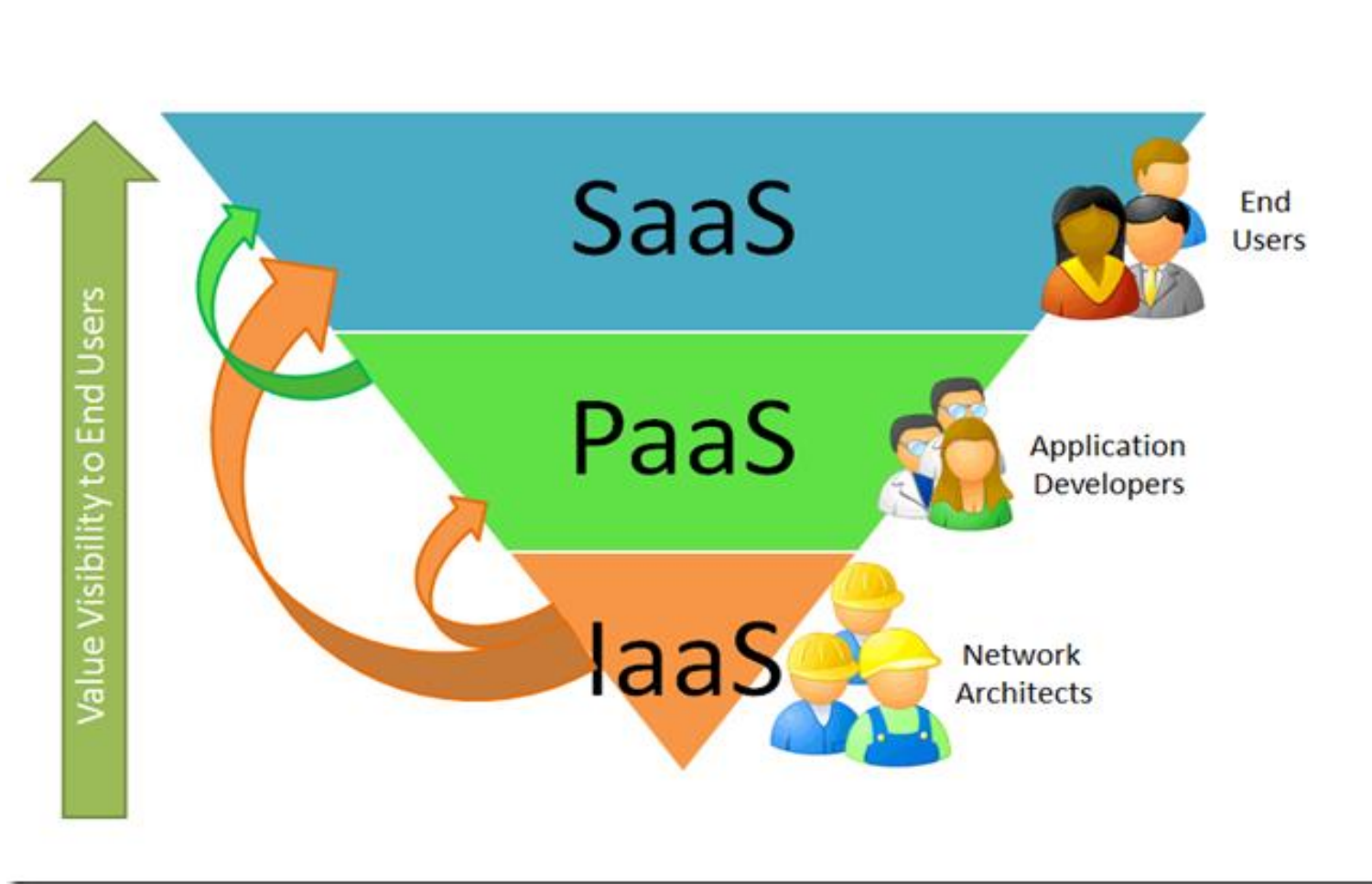


Image credit:

<http://blogs.msdn.com/b/seliot/archive/2010/03/04/what-the-heck-is-cloud-computing-another-re-look-with-pretty-pictures.aspx>

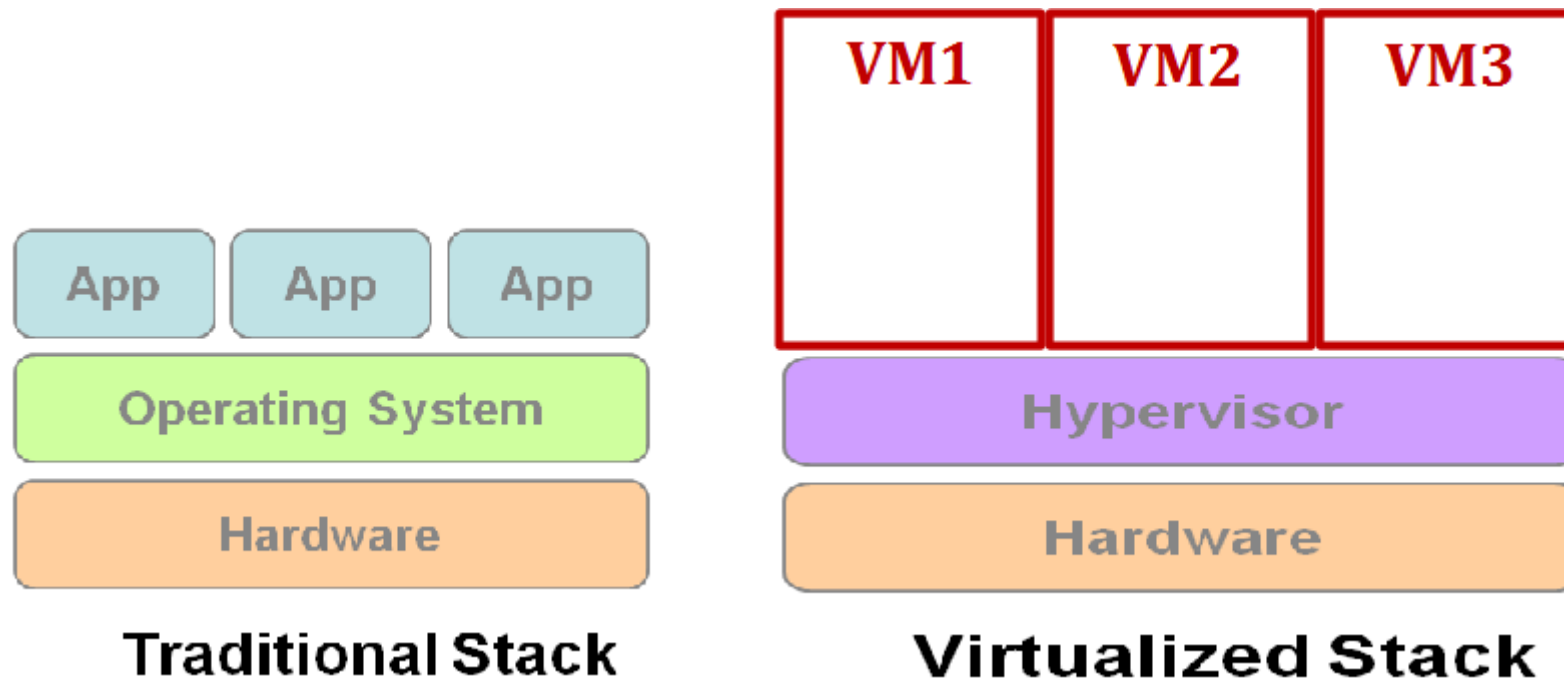
This document is available free of charge on

StuDocu.com

Downloaded by Rahul Vashistha (rahulvashistha1412@gmail.com)

Virtualization

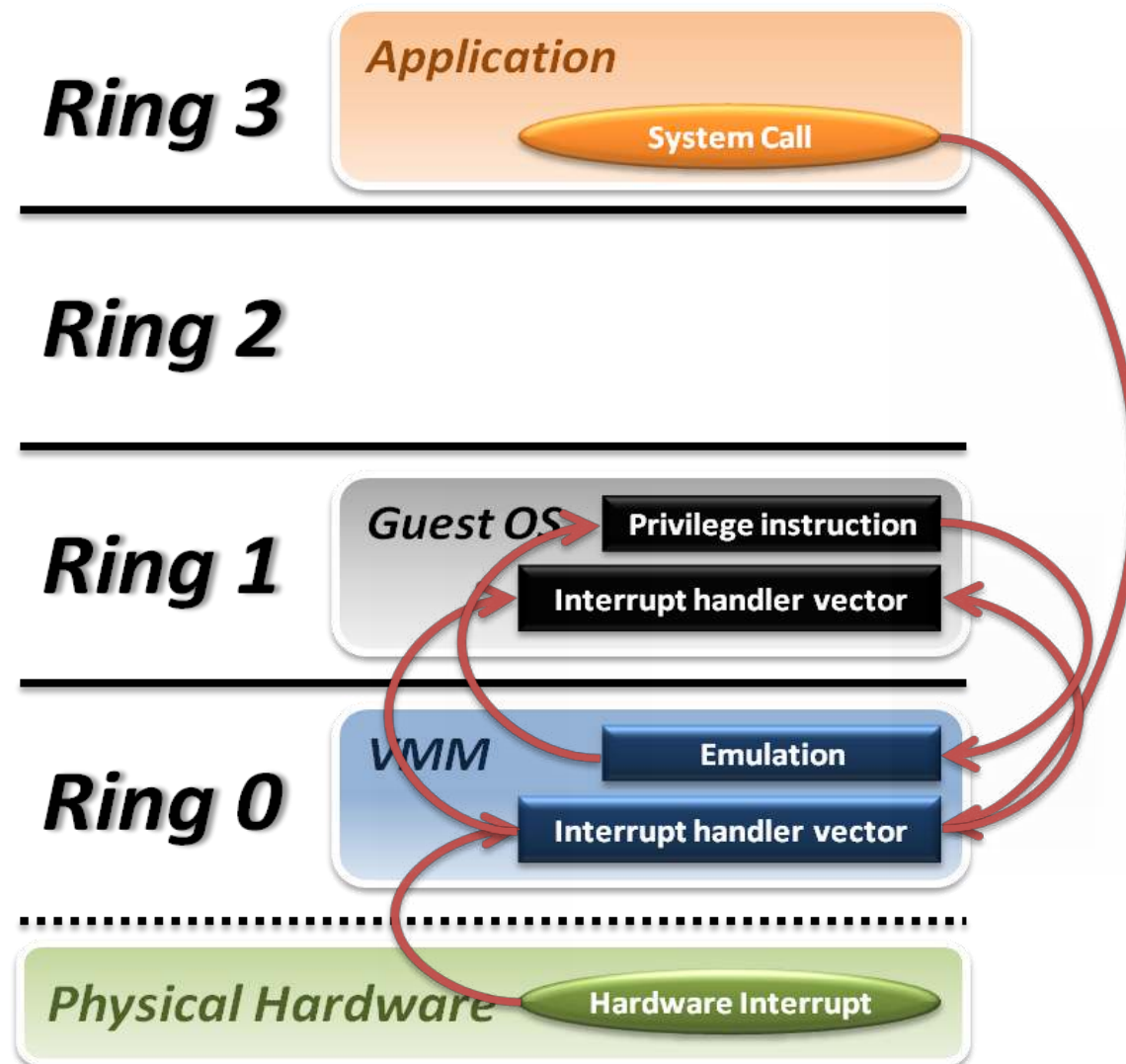
- **Virtualization** is the abstraction of logical resources away from underlying physical resources.
- A **hypervisor** (or **Virtual Machine Monitor** (VMM)) virtualizes a platform's operating system.
 - The hypervisor manages OS' as virtual machines (VMs) , enabling multiple OS' to share the same physical hardware.



Hypervisor's Trap and Emulate Model

- The hypervisor's virtualization paradigm is *trap and emulate* :
 - Normal instructions of guest OS
 - run directly on processor in user mode.
 - System Calls
 - CPU will trap to interrupt handler vector of Hypervisor.
 - Hypervisor jump back into guest OS.
 - Hardware Interrupts
 - Hardware makes CPU trap to interrupt handler of Hypervisor.
 - Hypervisor jumps to corresponding interrupt handler of guest OS.
 - Privilege Instructions
 - Running privilege instructions in guest OS will be trapped to Hypervisor for instruction emulation.
 - After emulation, the Hypervisor jumps back to guest OS.

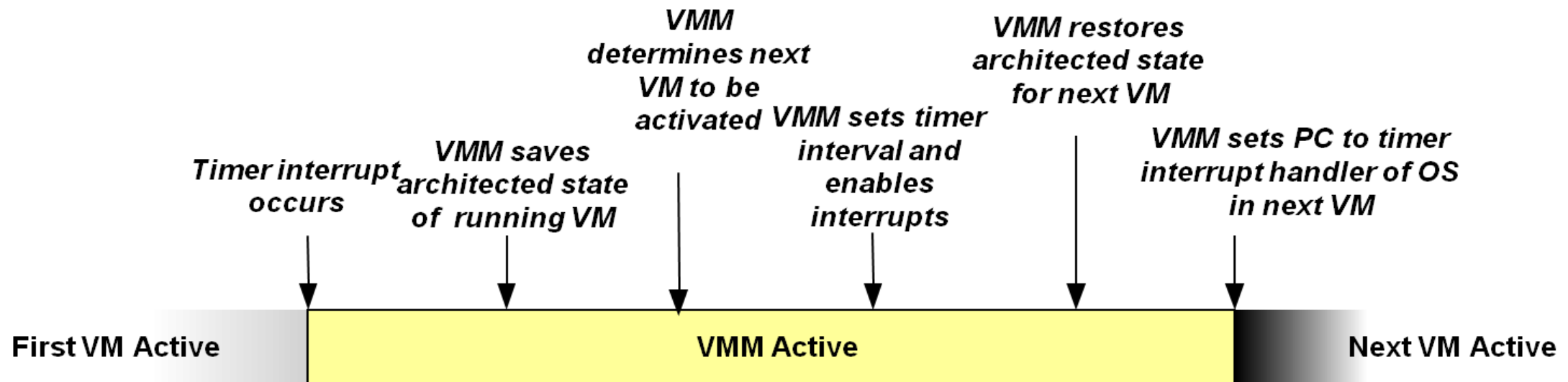
Trap and Emulate Model (VMM=Hypervisor)



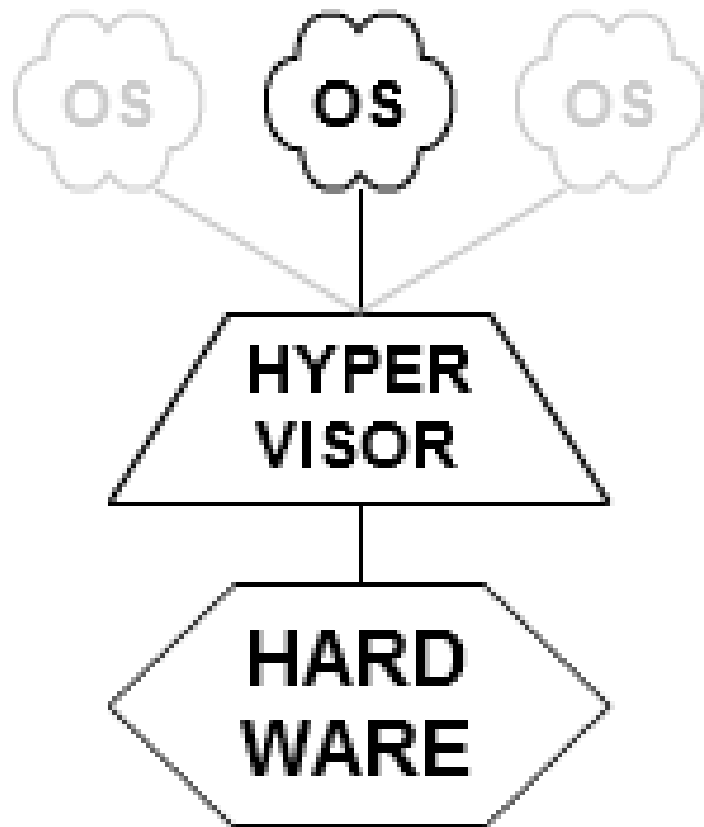
VM Context Switching

- The hypervisor context switches virtual machines:
 1. Timer Interrupt in running VM.
 2. Context switch to Hypervisor.
 3. Hypervisor saves state of running VM.
 4. Hypervisor determines next VM to execute.
 5. Hypervisor sets timer interrupt.
 6. Hypervisor restores state of next VM.
 7. Hypervisor sets the program counter to timer interrupt handler of next VM.
 8. Next VM active.

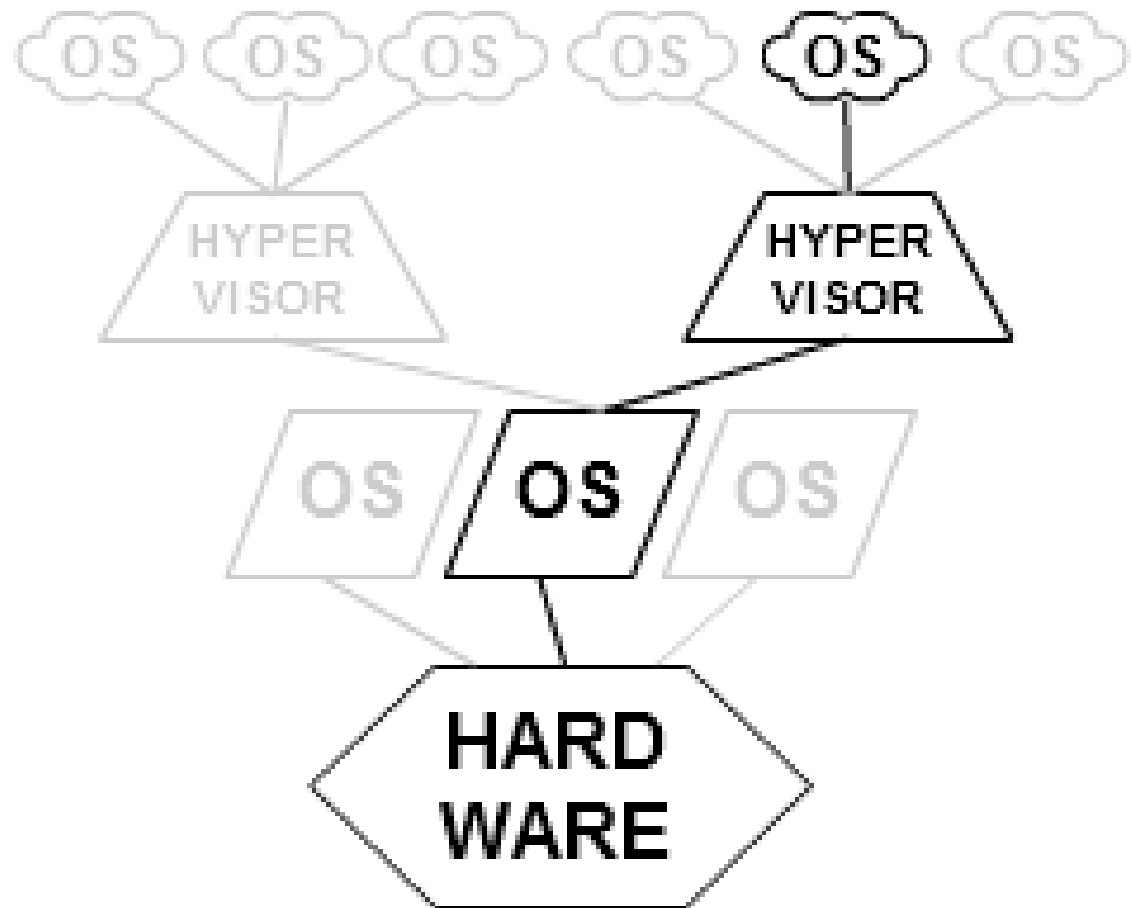
VM Context Switching



Hypervisor Models



TYPE 1
native
(bare metal)



TYPE 2
hosted

KVM (Kernel-based Virtual Machine)

- VMWare and Xen are the best-known virtualization platforms.
- KVM (Kernel-based Virtual Machine) is an open-source virtualization platform
 - Linux host OS
 - Run multiple virtual machines (Windows, MAC, etc) on your linux box
 - IO is virtualized using a device model in KVM
 - KVM requires a modified QEMU (open-source processor emulator) for its IO virtualization framework.
 - Type 1 Hypervisor, as it is a kernel-level module.

Virtual Machines are software – APIs to drive them.

OpenStack Compute REST API Features

- Authentication
- Servers
 - List Servers IPs
 - Create Server
 - Delete Server
 - Reboot Server
- Flavors (hardware config)
 - List Flavors
 - Get Flavor Details
- Images
 - List Images
 - Create Image/Snapshot
 - Get Image Details
 - Delete Image
- Backup Schedules
 - List Backup Schedules
 - Create/Update
 - Disable

Platform-as-a-Service (PaaS)

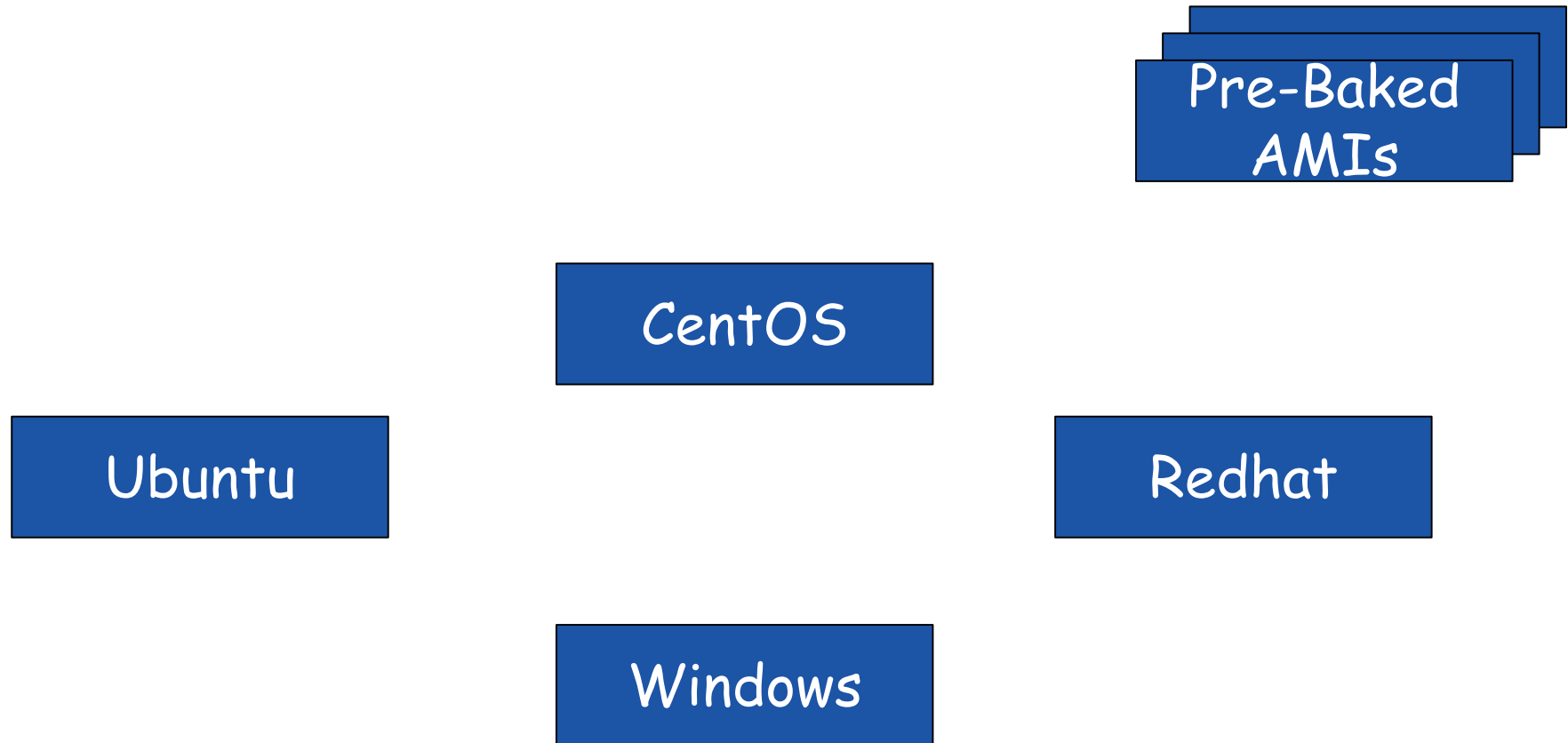
IaaS is not Enough

- IaaS provides virtual machines, but it cannot provide *elastic computing* by itself, where services scale up and down to meet user demand.
 - **Dynamic provisioning**
- Existing IaaS' do not provide support for the sharing middleware platforms among different VMs
 - **Multi-tenancy**

Multi-tenancy

- Multi-tenancy is where a single instance of the software runs on a server, serving multiple clients.
 - Think multiple users in a MySQL database
 - Java 9 should support multi-tenancy (many java programs running in the same JVM)
- The software should be able to provide a single service to all customers by setting configurations
 - More efficient use of server resources

IaaS - what you get

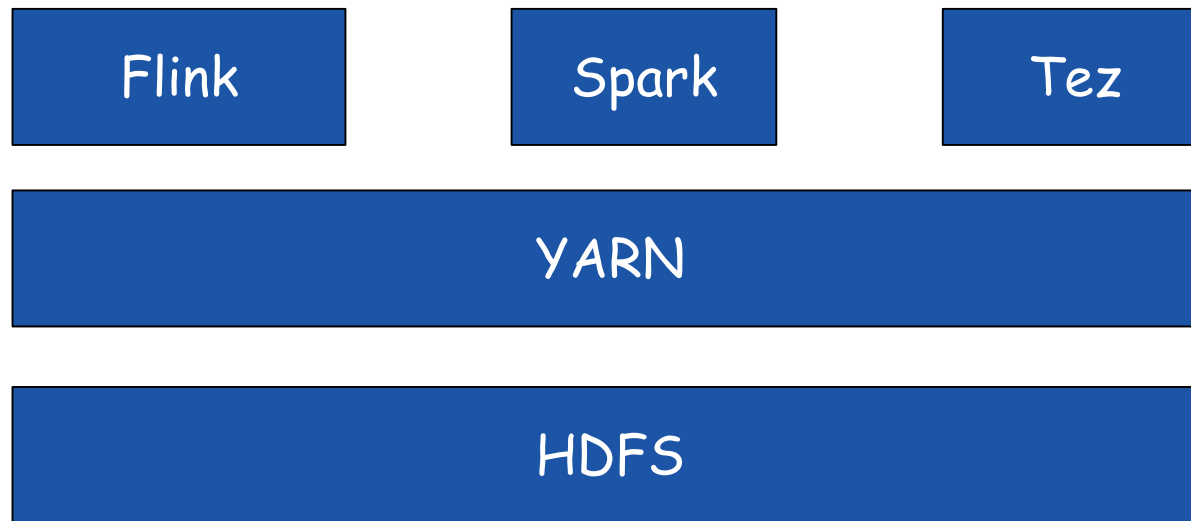


Infrastructure-as-a-Service

This document is available free of charge on **StuDocu.com**

Downloaded by Rahul Vashistha (rahulvashistha1412@gmail.com)

You might prefer this...



Configured stack of servers, dependencies, and firewalls and your app installed.

A Platform-as-a-Service

Running on lots of machines...

Data Center



PaaS



PaaS



PaaS



PaaS



PaaS



PaaS



PaaS



PaaS



PaaS

Platform-as-a-Service (PaaS)

- **Platform as a Service (PaaS)** is a computing platform that abstracts the infrastructure, OS, and middleware to drive developer productivity.
- PaaS leverages **dynamic provisioning**
- PaaS leverages **multi-tenancy**

Closed PaaS

- A **closed PaaS** provides a fixed set of services you can use. You cannot install your own services.
- They are typically hosted at some IaaS provider.

Closed PaaS	Supported Langs/Services
Heroku	Ruby, Node.js, JVM-langs, Python, SQL-DB, KV-Store
AppFog	PHP, Ruby, Node.js, Python, SQL-DB, KV-Store
AppEngine (Google)	Python, JVM-langs, GoLang
AWS Beanstalk, RightScale, EngineYard, CloudBees,

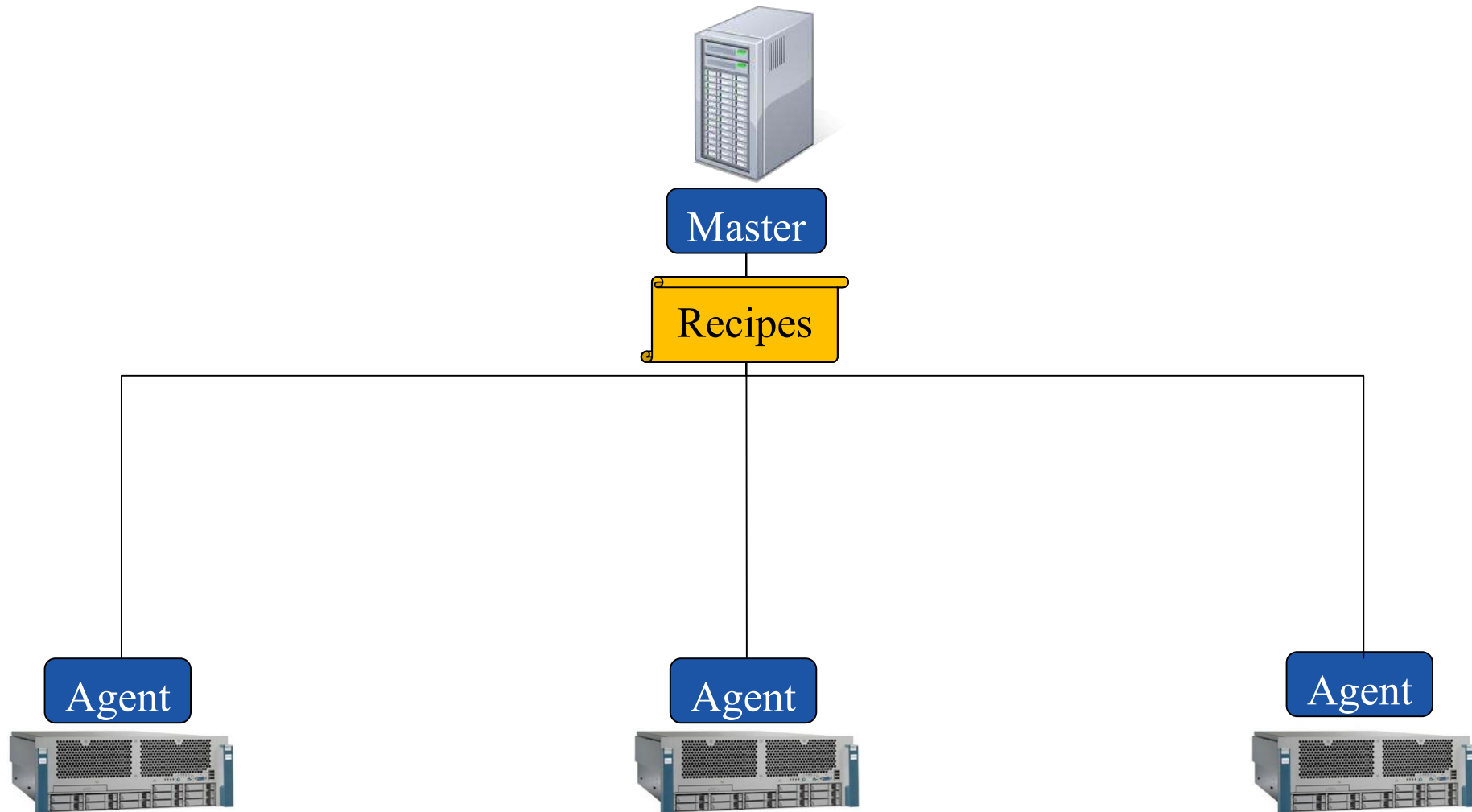
Open PaaS

- An **open PaaS** provides support for you to develop your own automated service deployments.



Kubernetes

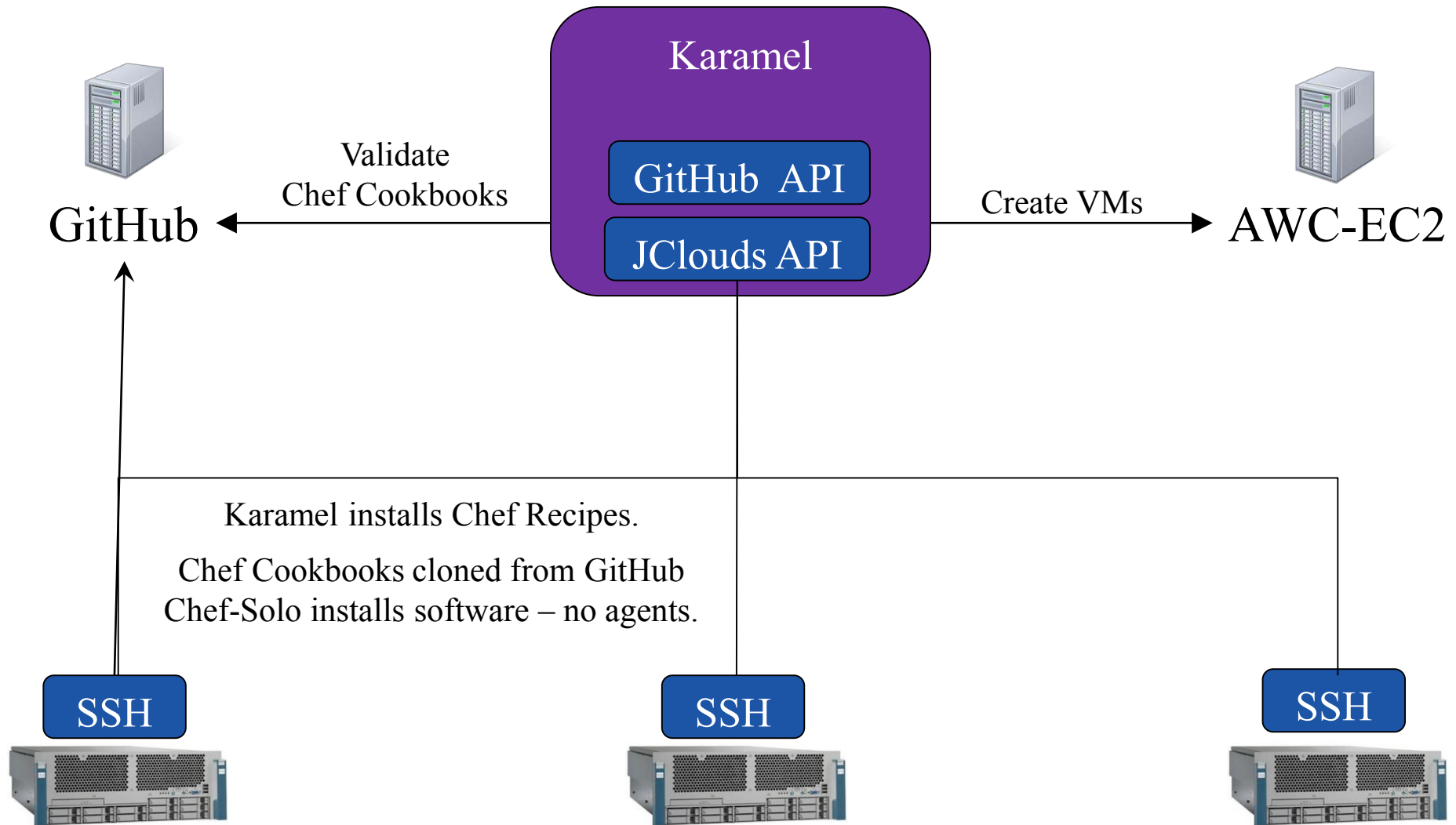
Automated Installation: Chef/Puppet/Salt/Ansible



Karamel/Chef

- Cluster definition in YAML
- Virtualization using JClouds
 - Support for AWS/EC2, Google Cloud Platform, OpenStack
- Karamelfile to Orchestrate Chef Recipes
- Chef-solo to execute recipes
- Standalone thick-client application
 - Ability to store user credentials
 - Ability to use discover the user's own ssh keys

Karamel/Chef



Case Study: Installing Hadoop

Cloudera Manager Cloud Express Wizard*

Abridged EC2-specific installation instructions*

Go to "EC2" in AWS web console and select "Instances"

Use the default "N. Virginia (us-east-1)" region.

Click on "Launch Instance"

On the next page, pick the "Ubuntu Server 12.04 LTS" 64-bit image.

select "Create a new Key Pair."

click "Create and Download your key pair."

save this file or you won't be able to SSH into the instance we're about to launch.

```
$ wget http://archive.cloudera.com/cm4/installer/latest/cloudera-  
manager-installer.bin
```

```
$ chmod +x cloudera-manager-installer.bin
```

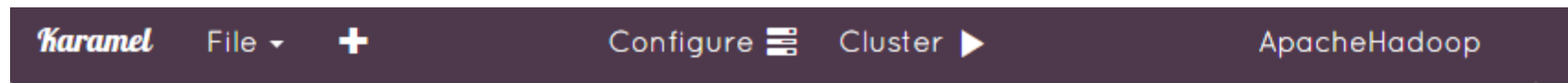
```
$ sudo ./cloudera-manager-installer.bin
```







*<http://blog.cloudera.com/blog/2013/03/how-to-create-a-cdh-cluster-on-amazon-ec2-via-cloudera-manager/>

Karamel Cluster Definition

```
name: ApacheHadoop
ec2:
  type: m3.medium
  region: eu-west-1
cookbooks:
  hadoop:
    github: "hopshadoop/apache-hadoop-chef"
    version: "v0.1"
groups:
  namenode:
    size: 1
    recipes:
      - hadoop::namenode
      - hadoop::resourcemanager
datanodes:
  size: 2
  recipes:
    - hadoop::datanode
    - hadoop::nodemanager
```

Karamel Hadoop Cluster - WebUI



 namenodes	 datanodes
hadoop::nn 	hadoop::dn 
hadoop::rm 	hadoop::nm 

Other Cluster-Definition Driven PaaSes

- Amazon Web Services OpsWorks
 - JSON cluster definition
 - Virtualization using EC2
 - Custom Orchestration
 - Chef-solo as provisioner
- Google Kubernetes
 - JSON cluster definition
 - “Virtualization” using Docker Containers
 - Extended Linux Containers
 - Orchestration support for Docker Containers
 - No built in support for orchestration

Software-as-a-Service (SaaS)

This document is available free of charge on

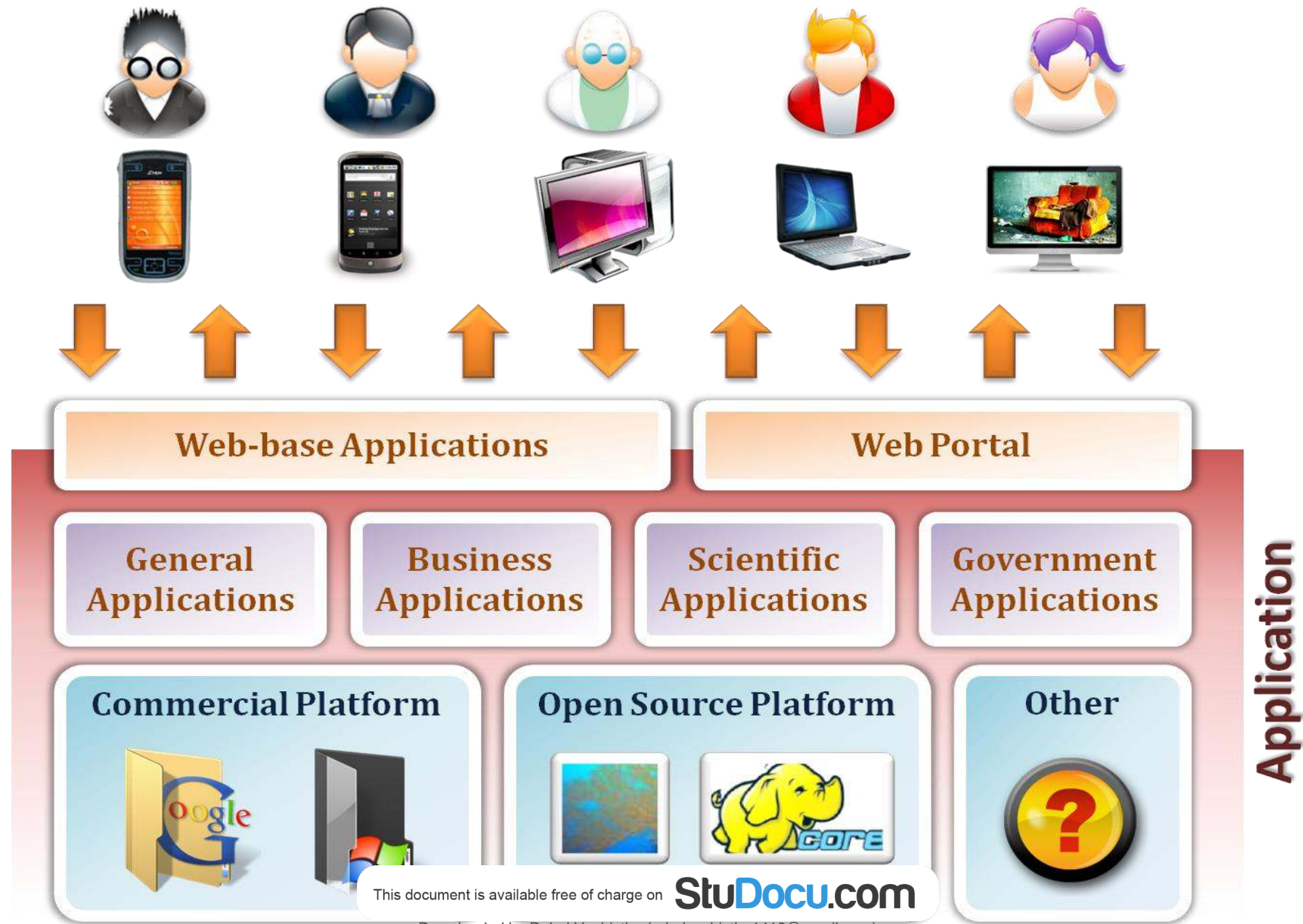
StuDocu.com

Downloaded by Rahul Vashistha (rahulvashistha1412@gmail.com)

Software as a Service

- Software as a Service - SaaS
 - Run applications on a provider's on a cloud infrastructure.
 - Applications are accessible from various client devices through a thin client interface such as a web browser.
 - User is oblivious to the underlying cloud infrastructure
- Examples
 - Dropbox
 - Google Apps (e.g., Gmail, Google Docs, Google sites,..)
 - Salesforce.com

Software as a Service



Obstacles To Cloud Computing

- Data Lock-in
- Data Confidentiality/Auditability
- Data transfer bottlenecks/costs
- Performance unpredictability for systems apps
- Legislative Compliance Concerns in Europe

Summary of Cloud Computing Architecture

Visual Model Of NIST Working Definition Of Cloud Computing

<http://www.csrc.nist.gov/groups/SNS/cloud-computing/index.html>



Conclusions

- Cloud computing has enabled an explosion in large-scale computing services and applications.
- Clouds provide services at three main levels: IaaS, PaaS, SaaS.
- New programming models enable easier development of large-scale applications.
- Hadoop is the open-source enabling technology for Big Data
 - Hadoop is rapidly becoming the operating system for the Data Center

References

- Dean et. Al, "MapReduce: Simplified Data Processing on Large Clusters", OSDI'04.
- Schvachko, "HDFS Scalability: The limits to growth", Usenix, :login, April 2010.
- Murthy et al, "Apache Hadoop YARN: Yet Another Resource Negotiator", SOCC'13.
- "Processing a Trillion Cells per Mouse Click", VLDB'12

References

Dean et al., MapReduce: simplified data processing on large clusters, *Comms of ACM*, vol 51(1), 2008.

Armburst et al., "Above the Clouds: A Berkeley View of Cloud Computing"

"Cloud Computing: Principles and Paradigms," R. Buyya et al. (eds.), Wiley, 2010.

"Cloud Computing: Principles, Systems and Applications," L. Gillam et al. (eds.) Springer, 2010.

Jeffrey Dean and Sanjay Ghemawat: "MapReduce: Simplified Data Processing on Large Clusters" in *OSDI* 2004

Senjay Ghemawat, : "The Google File System". *SIGOPS Operating Systems Review* 37(5), 2003

M. Isard et al.: "Dryad: Distributed Data-parallel Programs from Sequential Building Blocks" in *EuroSys* 2007