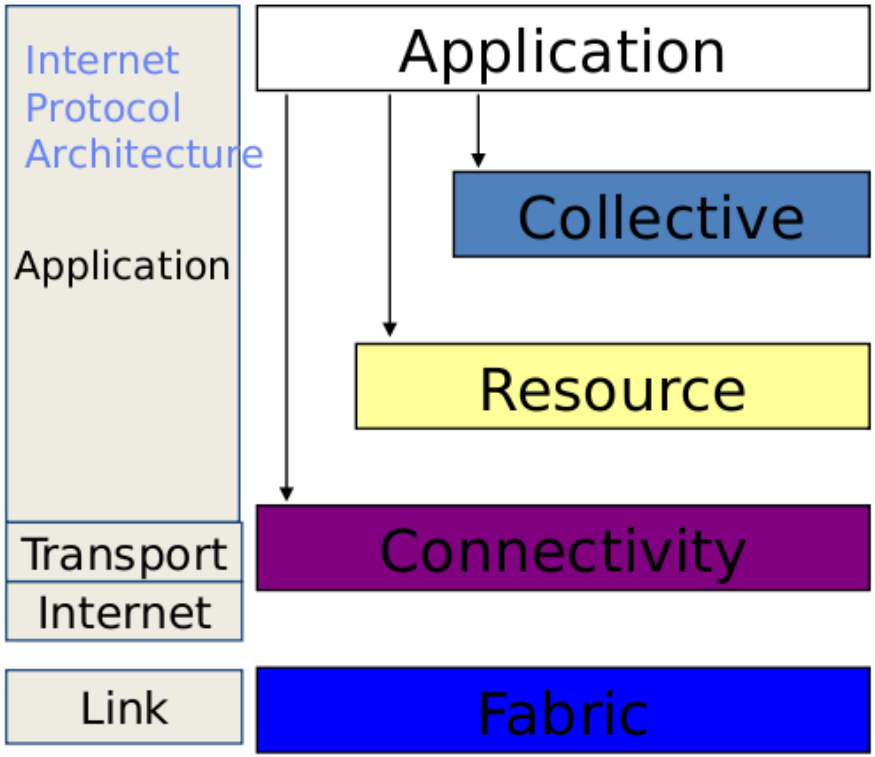

# Anatomy of the Cloud

Salvatore Filippone, PhD

School of Aerospace, Transport and Manufacturing
salvatore.filippone@cranfield.ac.uk

# Anatomy of the Cloud




"Coordinating multiple resources": ubiquitous infrastructure services, app-specific distributed services

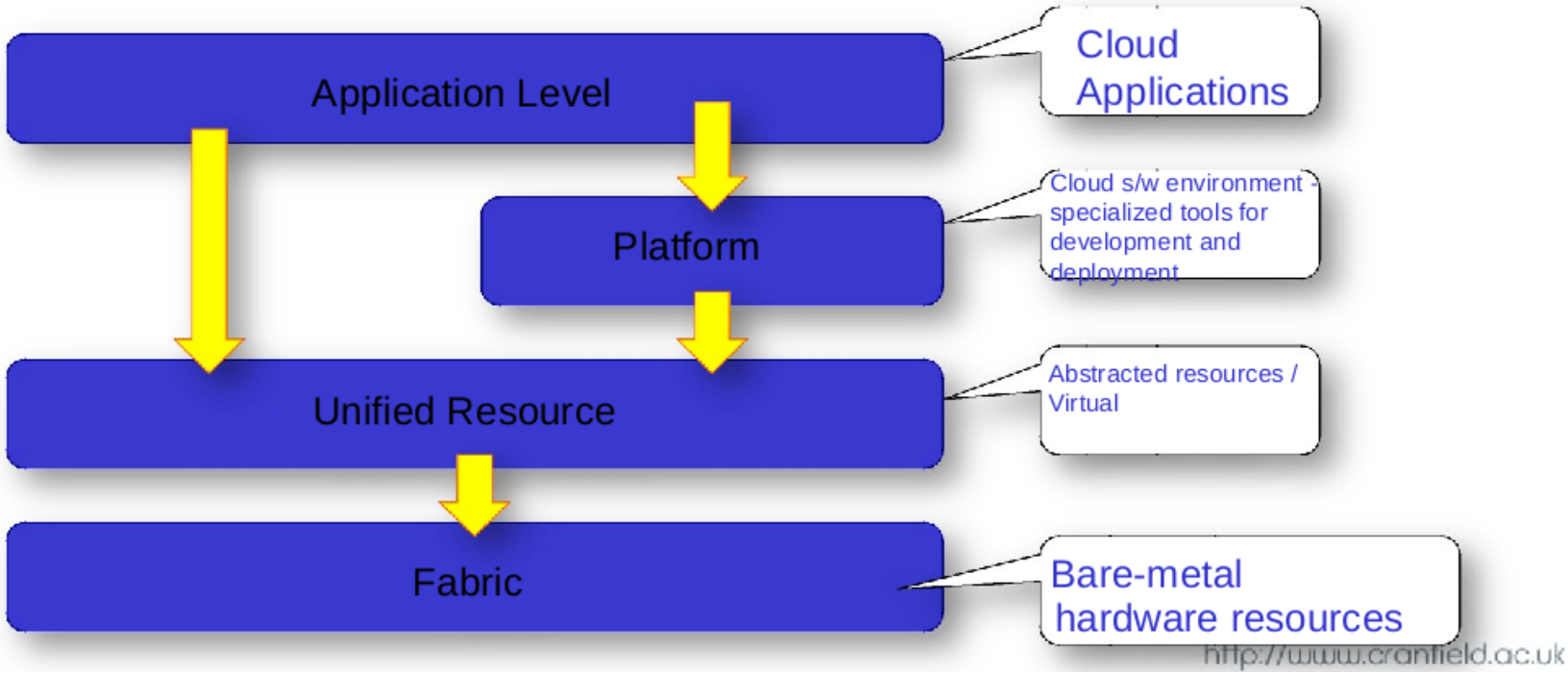
"Sharing single resources": negotiating access, controlling use

"Talking to things": communication (Internet protocols) & security

"Controlling things locally": Access to, & control of, resources

# Basic Architecture

Many implementations exist, but their general structure is usually quite similar



# Basic Architecture

## Fabric

- Bare metal hardware resources
- Compute, storage and communication resources
- Completely hidden from the application level
- An application can only access hardware through an intervening abstract layer, usually virtualised (UR layer)
- Developer cannot use resources, such as storage, directly
- Increases the security of the resources
- Compare with traditional compute systems, where the application communicates with the h/w through an API layer (via the OS)

# Basic Architecture

## Unified Resource

- Presents the abstracted hardware resources to the end user
- Resources are encapsulated and presented to developers as integrated resources
- For example, a storage system spread over many physical components may be presented as a single logical entity
- Manages resource usage between end user applications
- End user applications are unaware of other competing applications or system using the same physical hardware
- This can be an issue when optimising for performance because it is often more efficient to have all the resources on a single physical machine

# Basic Architecture

## Platform

The platform layer provides tools, frameworks and services to the application developer

- Collection of specialized tools, middleware, services on top of UR layer
- Allows the developer to access resources in a consistent or simplified way
- Typically a development language or framework (e.g., Ruby on Rails, Python, .NET, Java) is contained within this environment
- Provides a development platform for the application developer
- A further level of abstraction
- For example, a simplified API for storing data in a relational database, a web hosting environment, a scheduling service
- Can be used to present a familiar programming framework for development of Cloud applications, such as Microsoft .Net

# Basic Architecture

## Application Level

The uppermost level of the Cloud environment

- This is the only level that the user of the application can see
- Contains the applications that would run in the Clouds.
- It is the interface level between the user and the Cloud environment
- Generally served via the internet and this application typically does one thing. This could be email (e.g., Gmail) or CRM (e.g., SalesForce)

Two possible approaches available to the application developer

- Use the supplied platform framework to implement the application, or communicate directly with the underlying, virtualised environment
- The latter method is more flexible but requires more management of resources
- Usually the vendor restricts which method is available to the developer in their Cloud environment

# Service Provision

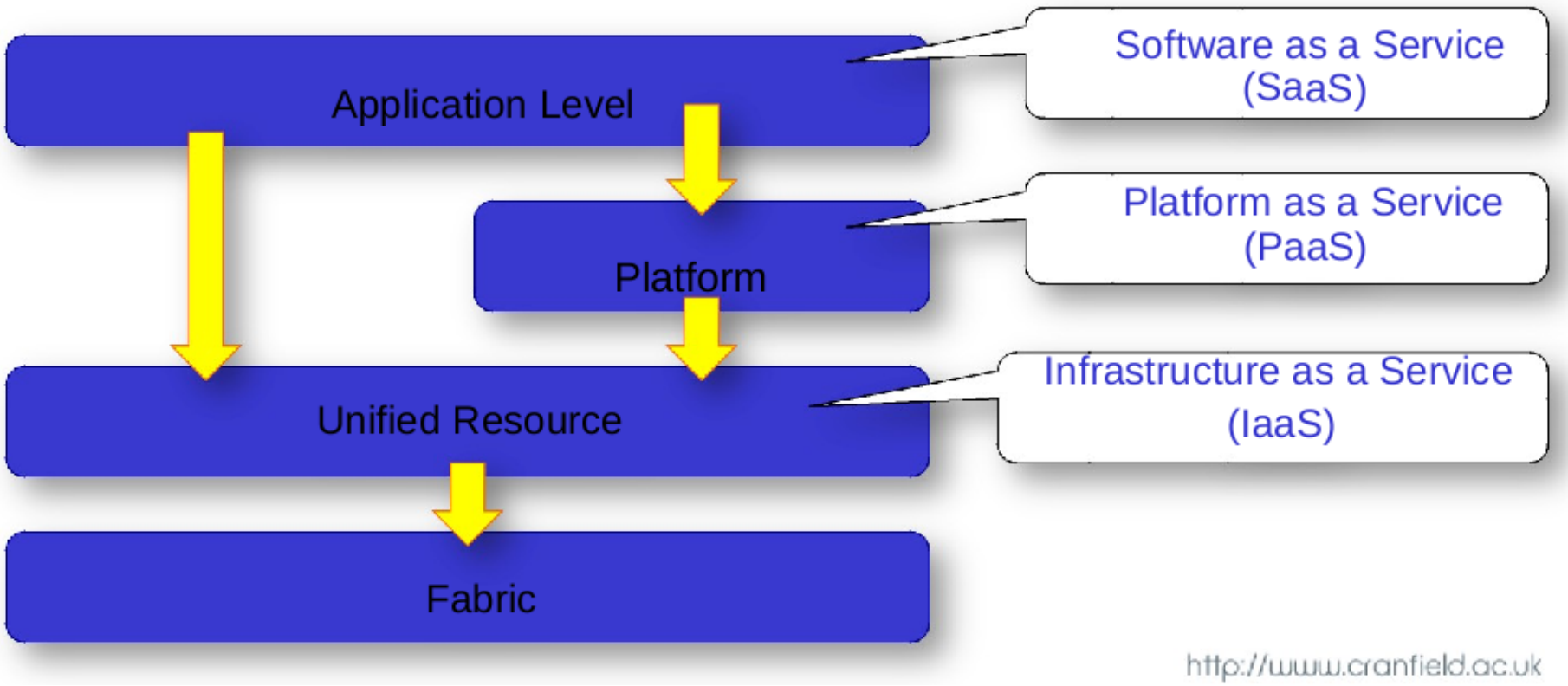
Cloud vendors present different levels of abstraction to the end-user and developer These are packaged as services: (cloud service models):

Infrastructure as a Service

- Platform as a Service
- Software as a Service

These can be roughly mapped onto the basic architecture, although precise details vary between vendors

# Basic Architecture



# Cloud Service Categories

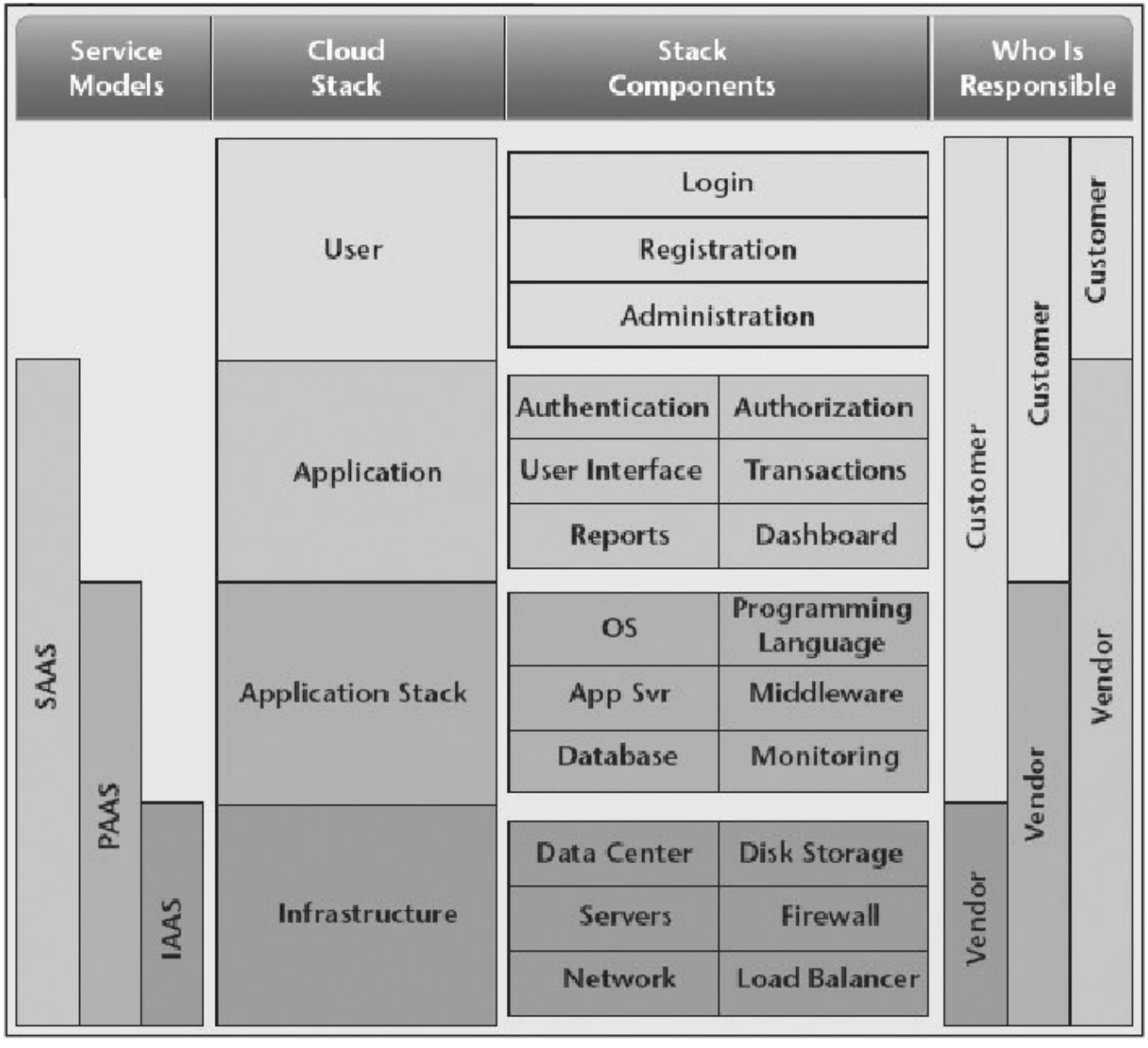
Clouds in general provide services at three different levels –

**SaaS:** High-level application frameworks.
⇒ Google App (Business Applications, Web Services, Multimedia) Salesforce.com, NetSuite

**PaaS:** Intermediate software development tools
⇒ Google App Engine, (Java, Python, .Net), Microsoft Azure

**IaaS:** Low-level hardware and virtual machines
⇒ Amazon EC2+S3, 3tera

# Cloud Services Comparison



# Basic Architecture

- Cloud applications can be created for a range of uses
- Potential range of application types requires different approaches to application design
- CloudApp (allows subscribers to share files, images, links, music, videos)
- AppStore by Amazon Web Services (facilitates quick and easy deployment of programs and applications stored in the cloud)
- AppEngine by Google (allows users to develop and run their own applications on Google's infrastructure)
- High performance data processing
- Limited or basic user interface
- Most interested in performing a single task - May be a one-off project
- 100% capacity some of the time, some capacity 100% of the time
- User facing applications
- Usually accessed via web based interfaces
- Utilized by many users concurrently

# Basic Architecture

## Different architectural abstractions offer different levels of flexibility to the application developer

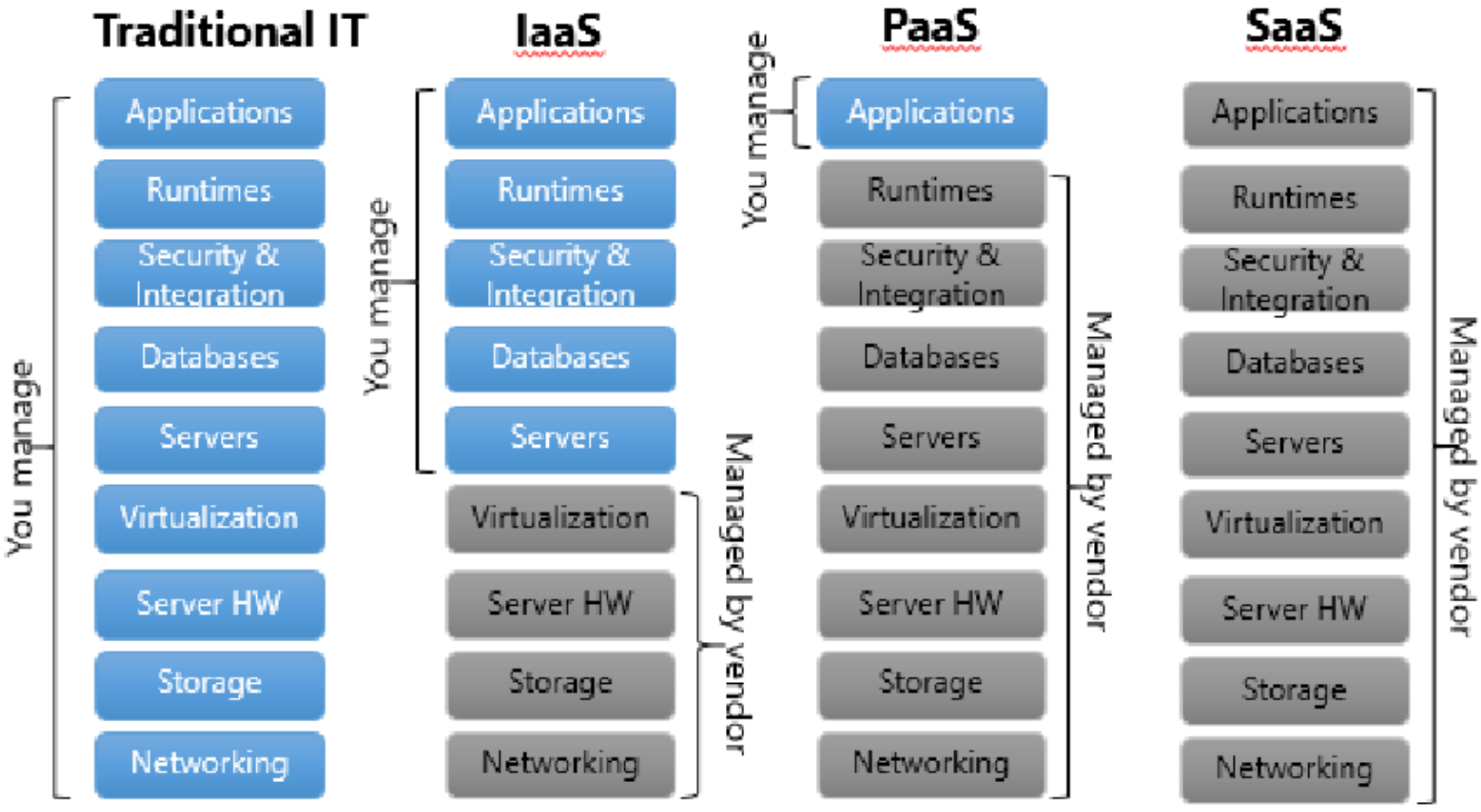
- The choice is wholly dependent upon the application and developer requirements
- For a web based application (SaaS), the developer could use either IaaS or PaaS based development environments
- Some applications are not well suited to particular environments
- Although the environments differ significantly, some of the design patterns used to create applications are very similar
- For example, creating dynamically scalable applications requires a flexible approach to processing data regardless of abstraction level
- Some patterns presented with IaaS are just as applicable to PaaS

# Architectural Abstractions

## IaaS — Infrastructure as a Service

- Infrastructure like CPU, disk space, servers, software, datacentre space, or network equipment is delivered as a fully outsourced service as a complete environment for development
- Highly configurable environments
- Resource usage metered on a pay-per-use basis
- Infrastructure can scale up or down dynamically based on application resource needs, although it is usually up to the developer to make best use of the extra flexibility
- Appropriate when there are existing SaaS services that fit user's needs (i.e., email, document management, CRM, etc.) and a minimum level of customization is needed.
- Mainly used by full-time developer
- No need to buy and maintain server infrastructure as these are provided by IaaS vendor

# IaaS


Traditional IT
IaaS
PaaS
SaaS
You manage
You manage
You manage
Managed by vendor
Managed by vendor
Managed by vendor
Applications
Runtimes
Security & Integration
Databases
Servers
Virtualization
Server HW
Storage
Networking

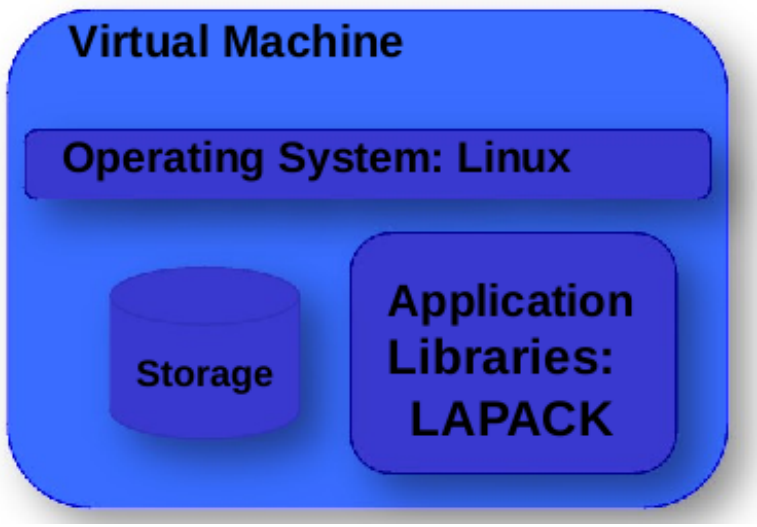
# Cloud Services Comparison

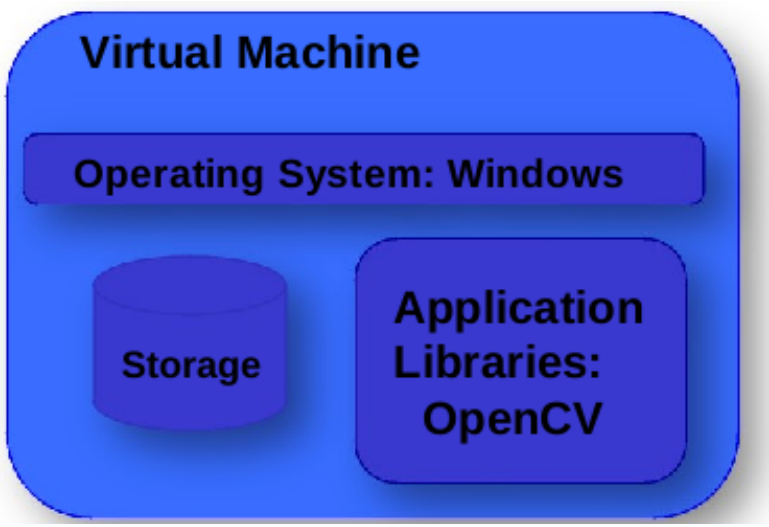
| = Managed for You | Standalone Servers | IaaS | PaaS | SaaS |
|---|---|---|---|---|
| Applications | ✗ | ✗ | ✗ | ✓ |
| Runtimes | ✗ | ✗ | ✓ | ✓ |
| Database | ✗ | ✗ | ✓ | ✓ |
| Operating System | ✗ | ✗ | ✓ | ✓ |
| Virtualization | ✗ | ✓ | ✓ | ✓ |
| Server | ✗ | ✓ | ✓ | ✓ |
| Storage | ✗ | ✓ | ✓ | ✓ |
| Networking | ✗ | ✓ | ✓ | ✓ |

# IaaS

## Allows the developer to create a custom environment

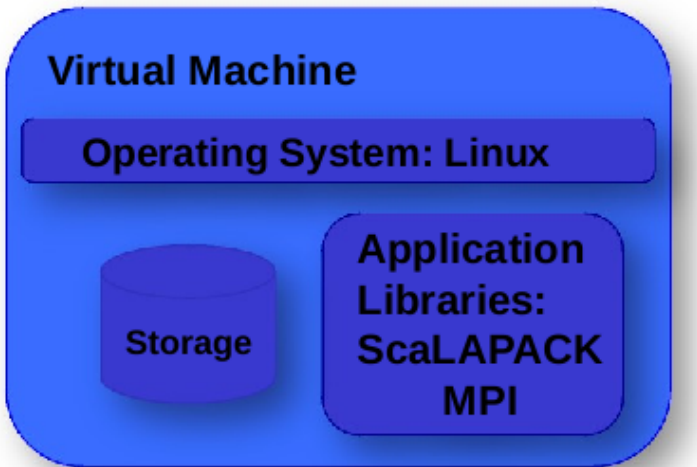
- Multiple such environments can be hosted by the vendor without interaction between them
- Gives total control to the developer
- Environments with specific libraries or frameworks can be created
- Unnecessary software can be removed to create a slimmed down environment

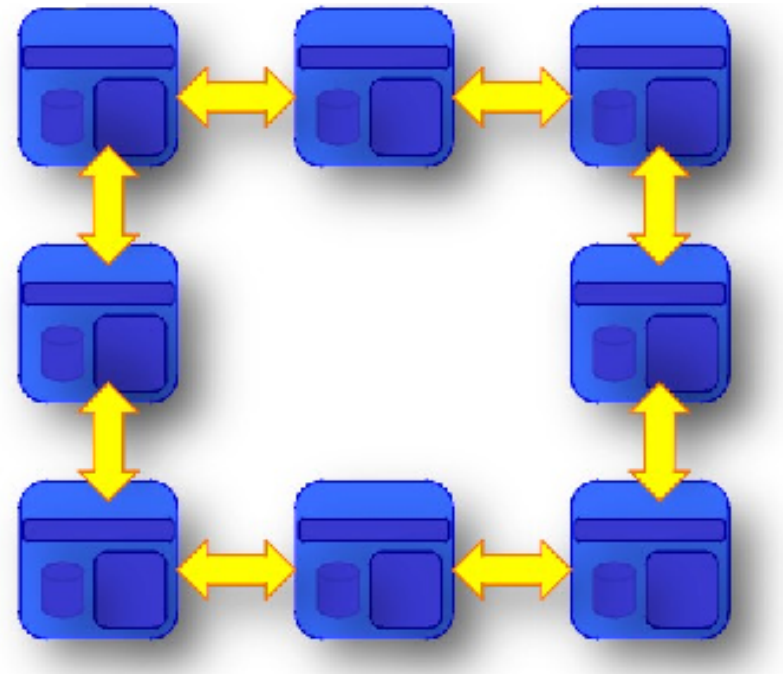


## Multiple component environment

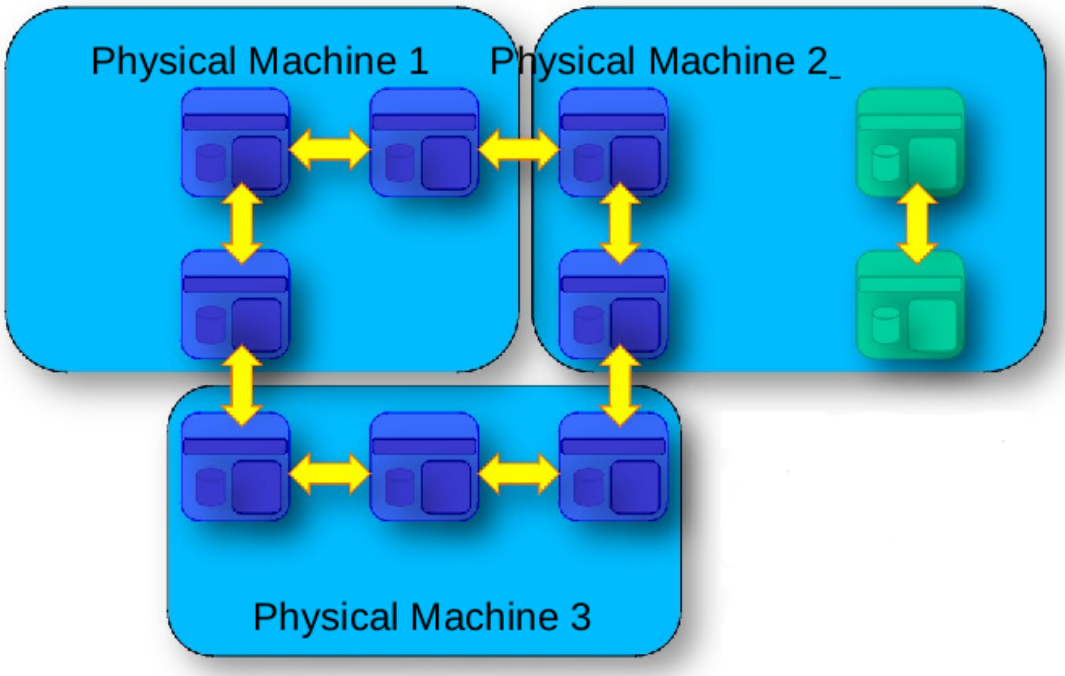
- IaaS gives you a server in the cloud (virtual machine) that you have complete control over.
- We are not limited to single custom VMs
- Multiple VMs can be created and configured to communicate



# IaaS Environments

Multiple component environments: VMs may or may not execute on the same physical platforms

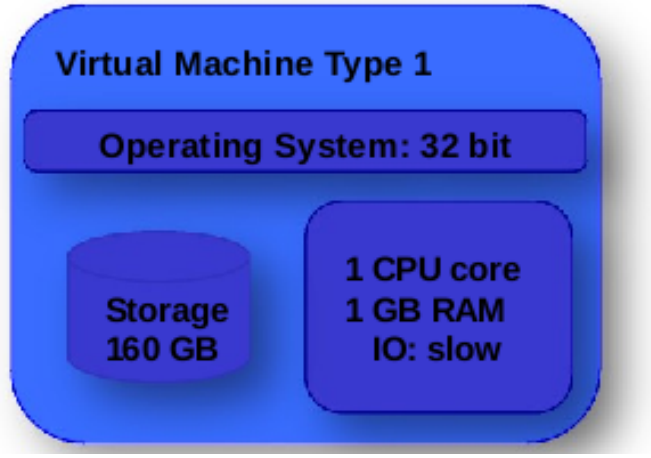


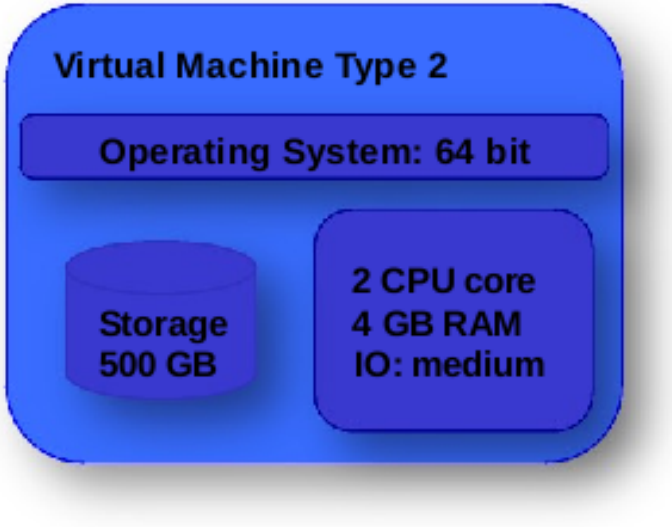

- This may lead to unexpected performance issues
- Data security may also need to be considered

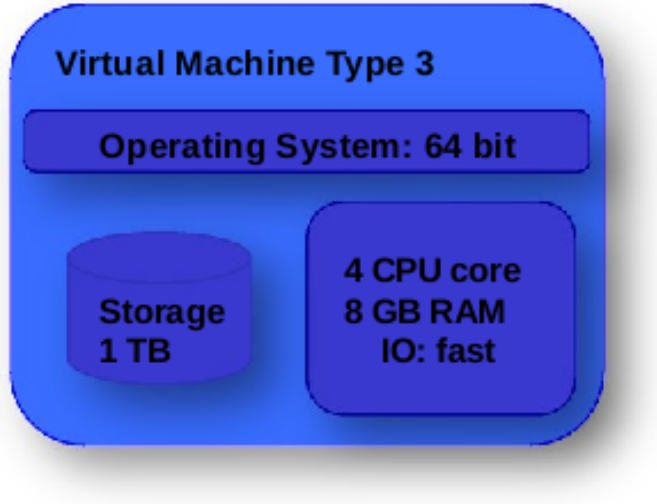
## Amazon Virtual Private Cloud (Amazon VPC)

- lets you set up a private cloud (a private, isolated section) within the Amazon Web Services, where you can launch AWS resources in a virtual network that you define.
- You can create a Hardware Virtual Private Network (VPN) connection between your corporate datacentre and your VPC and leverage the AWS cloud as an extension of your corporate datacentre.
- Nodes launched within a VPC aren?t addressable via the global internet
- Much more granular control over security.
- have a wall between Internet and your instances, you can decide to expose only a part of your infrastructure, basically the only one that needs to speak with the rest of the world

# IaaS Resource Use

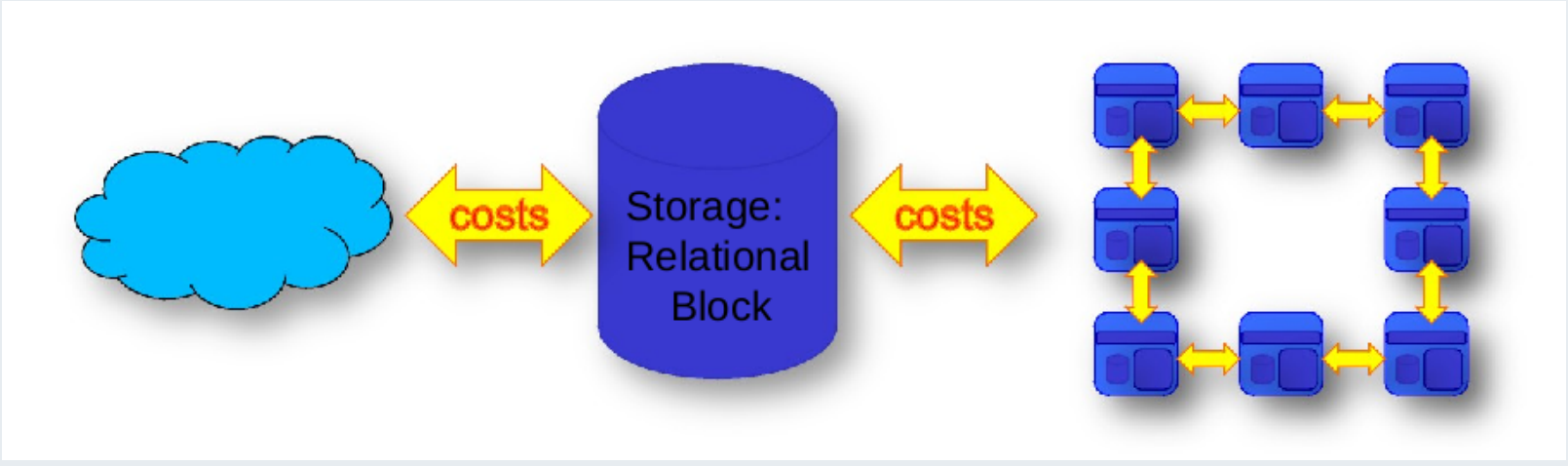


## Persistent storage costs

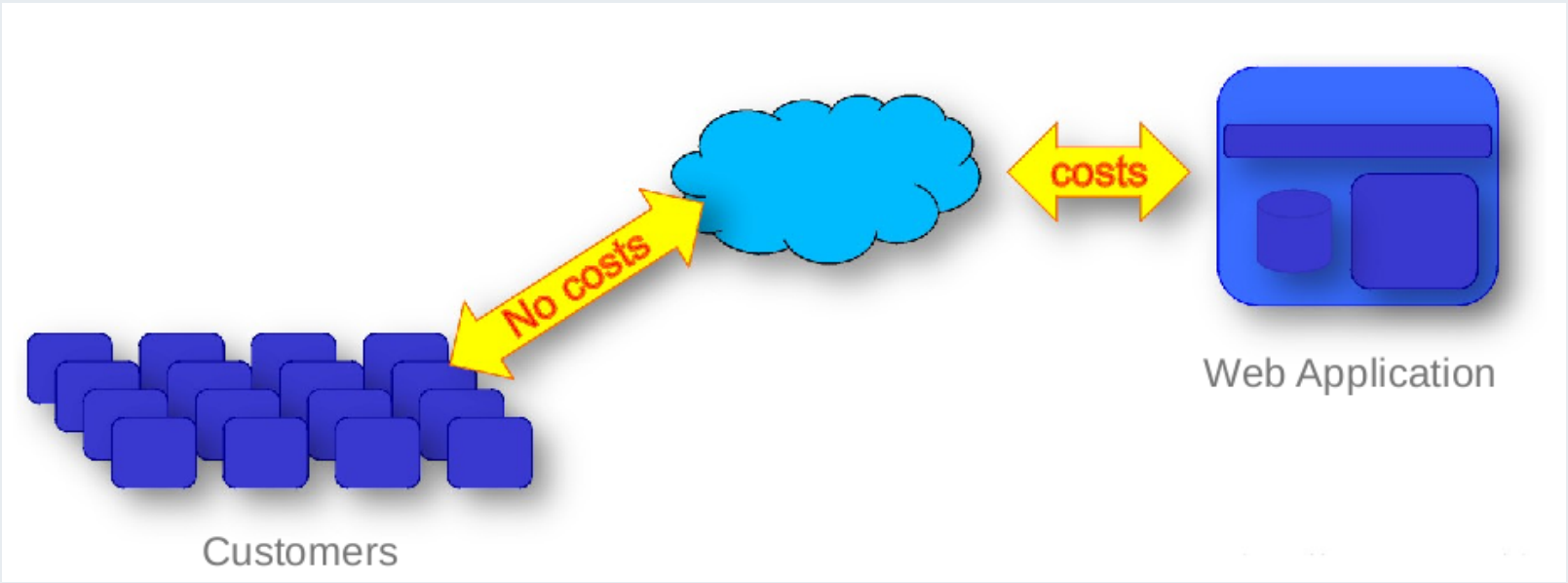
- Although each VM is assigned storage, the application may require persistent storage
- Allows data to be shared between VMs
- Data can be uploaded and stored in the Cloud without VM interaction
- Costs can be charged for data transfer between storage and VMs



# IaaS Resource Use

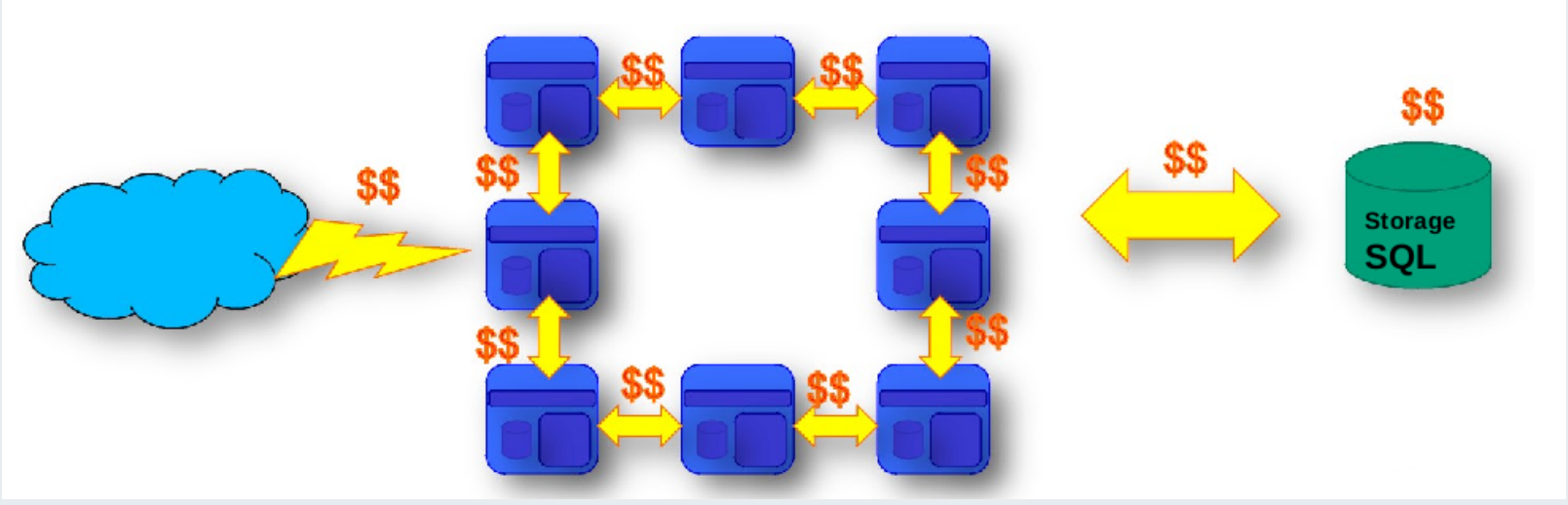
## Communication costs

- Communication can be charged for transactions between VMs
- Application transactions to and from the Cloud can be charged
- Customer transactions are paid for by the customer indirectly



## Designing applications for cost efficiency

- Unit costs are usually very small but the number of transactions can quickly increase the overall cost
- Applications that generate large data sets may be charged for moving the data off the Cloud infrastructure
- Applications need to be designed with these costs in mind!



# IaaS Resource Use

## Optimal design choice?

- 1 process node for 1000 hours?
- 1000 process nodes for 1 hour?

## Costs on AWS:

- Micro (single CPU) on-demand instance $0.0118 / hour
- Transferring data in/out of EC2:
- From internet $0.00/GB
- To internet: first GB free, then next 10TB / month $0.09/GB
- Additional storage charges
- Data transfer costs between instances $0.00 / GB
- Data transfer to other regions $0.02 / GB
- No penalties for concurrency!

Data from `https://aws.amazon.com/it/ec2/pricing/on-demand/`, Nov. 2018, EU London region.

# IaaS Scalable Provisioning

IaaS represents the most flexible development environment

- Complete control over VM configuration
- Complete application architectures can be created virtually with extensive choice over components
- However, developer usually responsible for starting new VMs as required
- Developer must monitor current processing load
- Compare with other abstractions (PaaS, SaaS)

# IaaS Scalable Provisioning

## Two ways of approaching scalable provisioning

1. The data set is fixed and processing nodes can be allocated at the start of processing
2. The data set is variable due to external factors

Each scenario may require a different approach to developing the application architecture

The details of application scalability will depend upon the level of abstraction

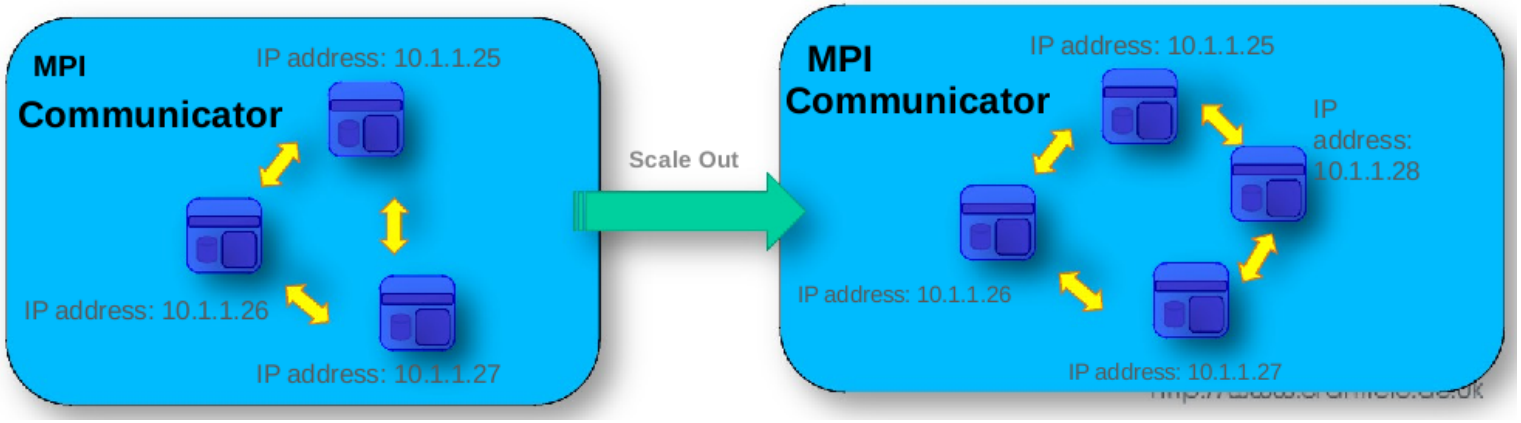
IaaS environments require the developer to explicitly handle the low level details

PaaS environments will offer some help to monitor workload and start new worker instances

# IaaS Scalable Provisioning

## Scalability will depend upon environment

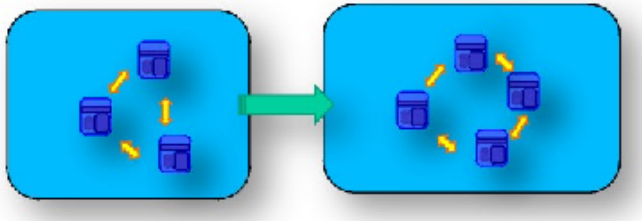
- Know your expected usage patterns
- Basic assumption still holds — unlimited resources
- But for private Clouds, some constraints may apply!
- Time to spin up a VM is of the order of minutes
- The developer needs to integrate the new instances into the current environment — networking and communications

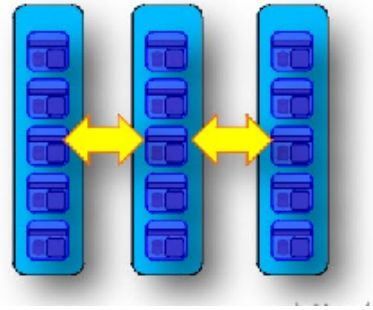


# IaaS Scalable Provisioning

Applications capable of dynamic load balancing can best take advantage of elasticity

- Architecture can be developed to take advantage of ad-hoc scalability of resources
- Requires more complete control over the design of the application
- May require different approach to traditional distributed computing
- How can we control load balancing?
- Two examples — web based transaction server and queue based pattern

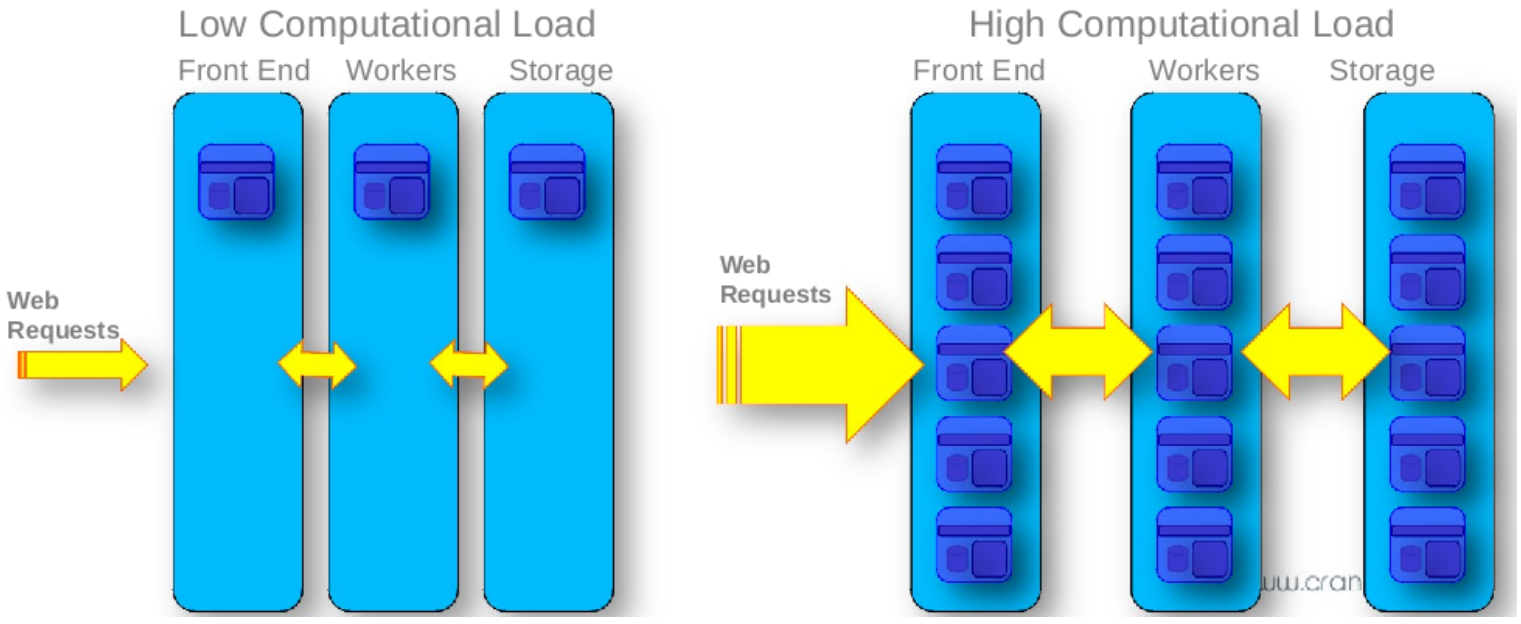




# IaaS Scalable Provisioning

## Web based transaction server — 3 tier system

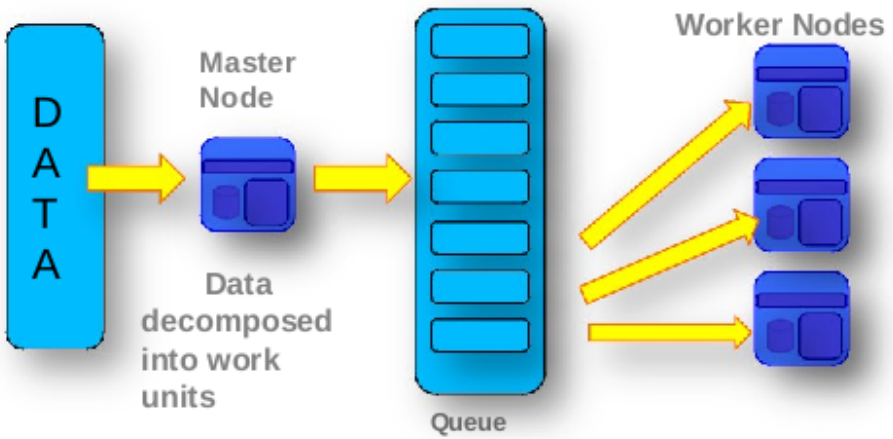
- Each transaction is discrete and independent — loosely coupled
- Front end nodes distribute jobs to workers
- Very scalable



# IaaS Scalable Provisioning

A more generalised approach might be to use the Master Worker pattern together with a Shared Queue pattern:

- A master node decomposes the data/work into smaller chunks and stores them in a shared queue
- A pool of worker nodes is created




- Each worker node retrieves the next work unit from the shared queue
- When the worker has completed, the result is sent back to the master node
- The worker retrieves a new work unit from the shared queue

# IaaS

## Providers

Amazon EC2, IBM, HP, Rackspace, Eucalyptus, Cisco, Joyent

## Properties

- Virtual machines offered to consumers
- Freedom of choice of operating system

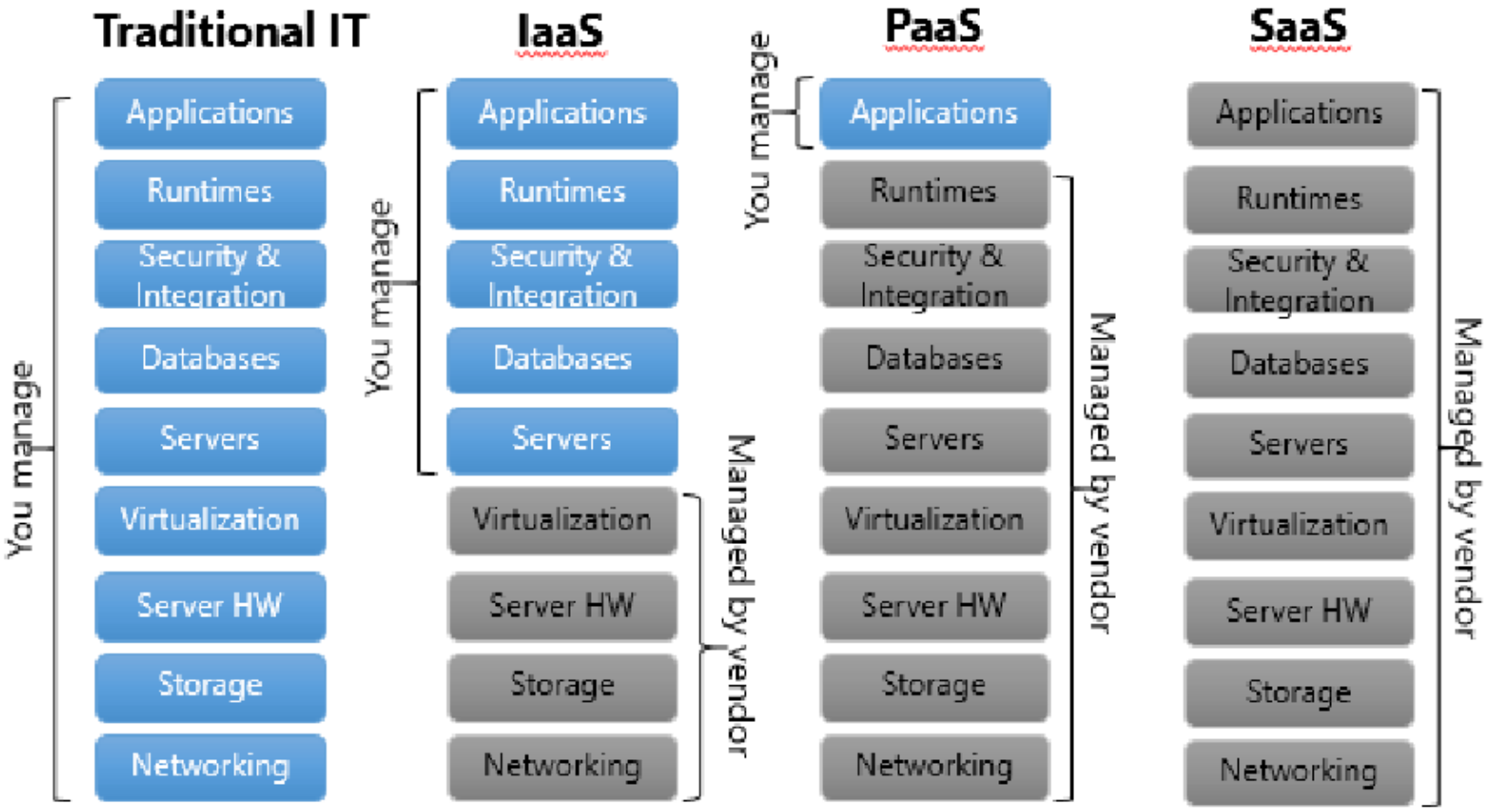
## Access to resources

Consumers have access to virtual machine instance, which can be configured to suite the opsys and application. No direct control over hardware resources.

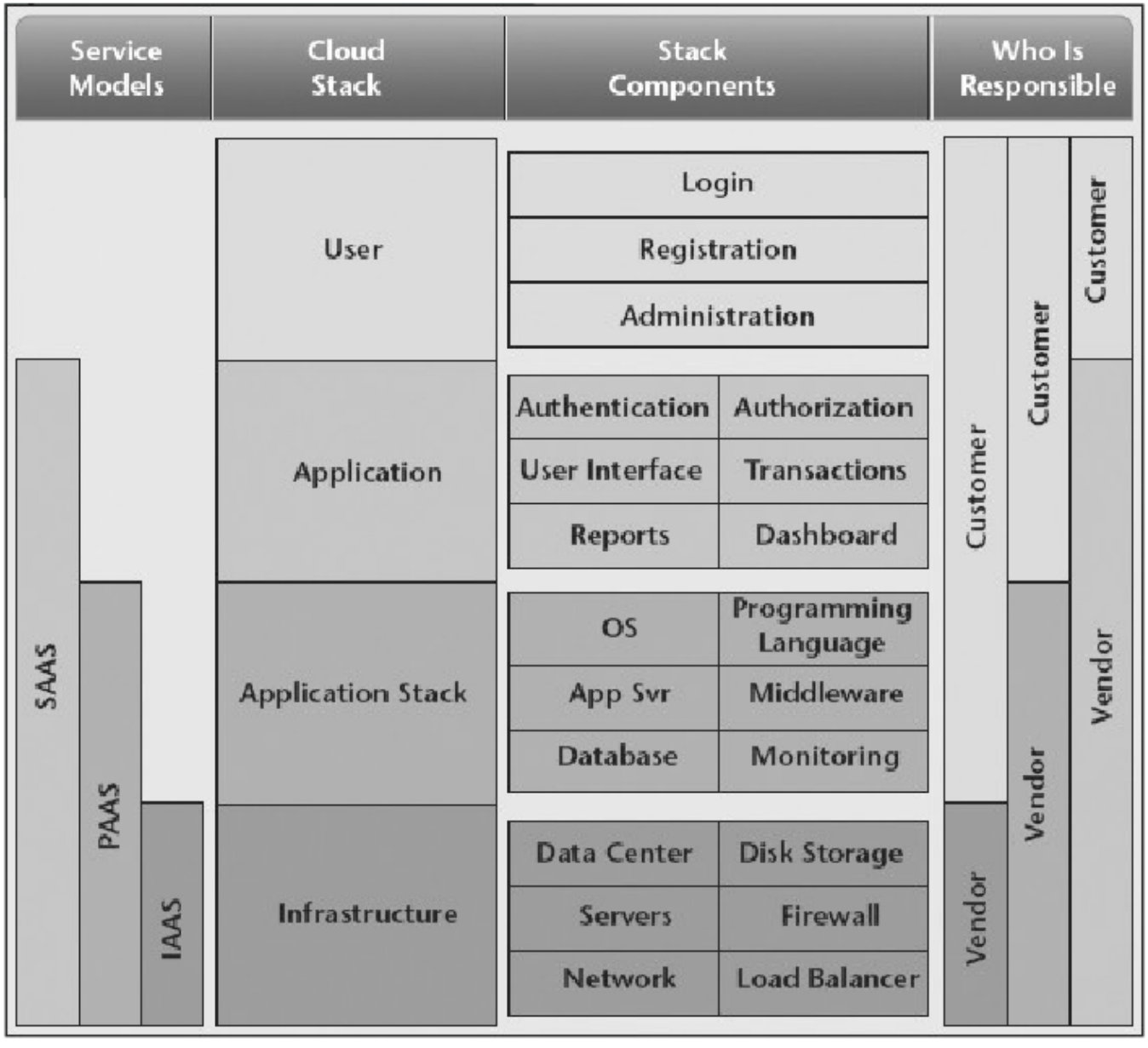
## Key challenges

- Governance;
- Data encryption;
- API Key Protection

# PaaS


Traditional IT
IaaS
PaaS
SaaS
You manage
You manage
You manage
Managed by vendor
Managed by vendor
Managed by vendor
Applications
Runtimes
Security & Integration
Databases
Servers
Virtualization
Server HW
Storage
Networking

# Cloud Services Paas



# PaaS

Platform as a Service: next level in architectural abstraction

- High level development environment
- Developer is presented with packaged resources available through APIs
- No need to pre-install or configure the resource as in the case of IaaS
- Different resources can be offered as a PaaS — compute, storage, etc.

Some crossover in definitions of PaaS and IaaS

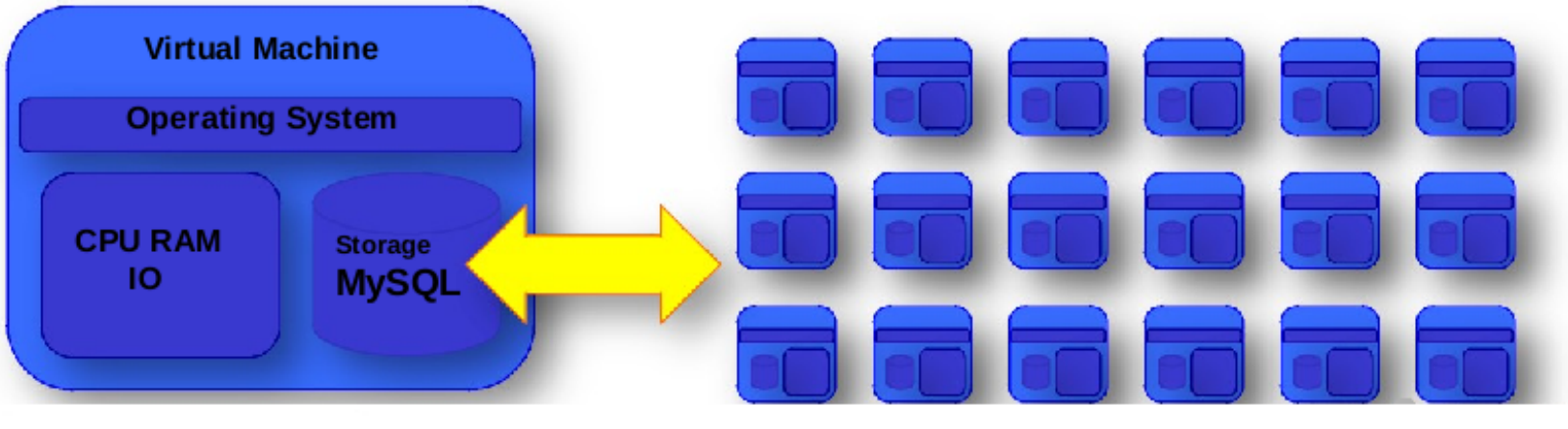
- Generally accepted as a complete development environment
- Useful to understand the differences between the two abstractions

## PaaS — Platform as a Service

- No direct access to virtualisation level
- High level integrated development environment for custom applications
- Less flexibility than IaaS, but many of the low level resources are packaged for ease of use
- Generally, the platform will handle scalability often without the developer needing to explicitly code for it.
- An ex. of this is the ability of Google AppEngine to spin up new instances of web applets as required.
- Vendors can present familiar development environments to aid with the software engineering process.
- An ex. of this is Microsoft Azure, which presents the .Net environment to the developer allowing them to harness programming languages

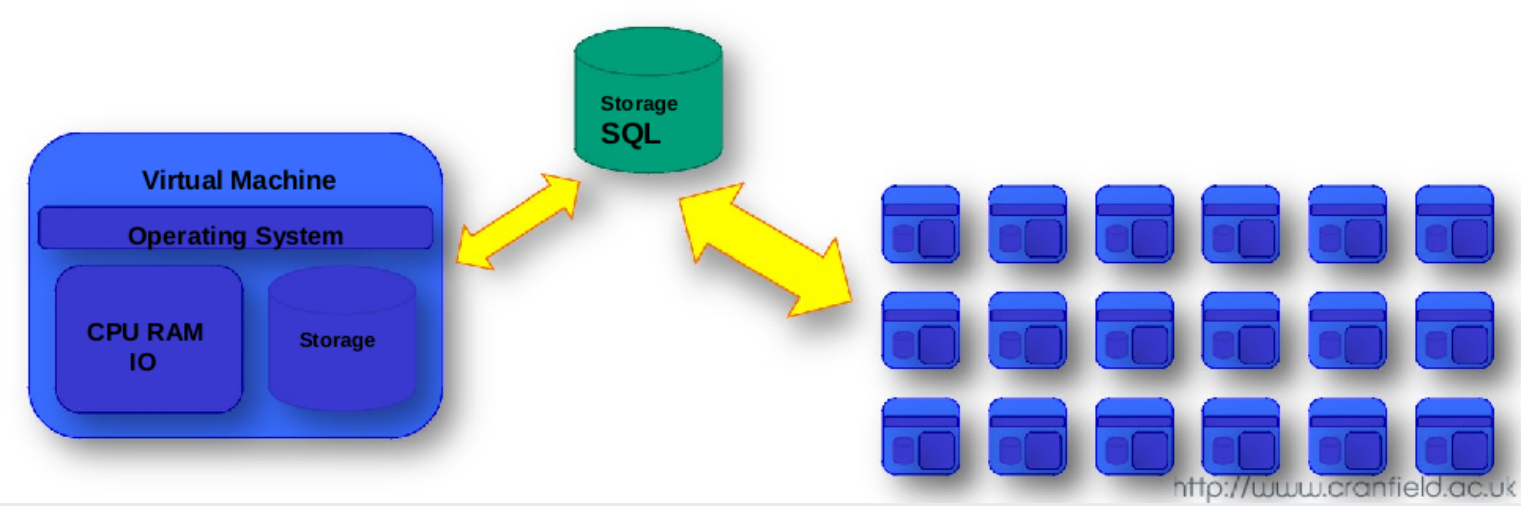
## Offering storage as a PaaS

- Consider a relational database, such as MySQL
- In an IaaS environment, a developer would need to create a VM with MySQL installed
- How can it be shared between multiple VMs?
- Developer would need to explicitly handle these interactions
- Bottlenecks may occur which would need to be optimised



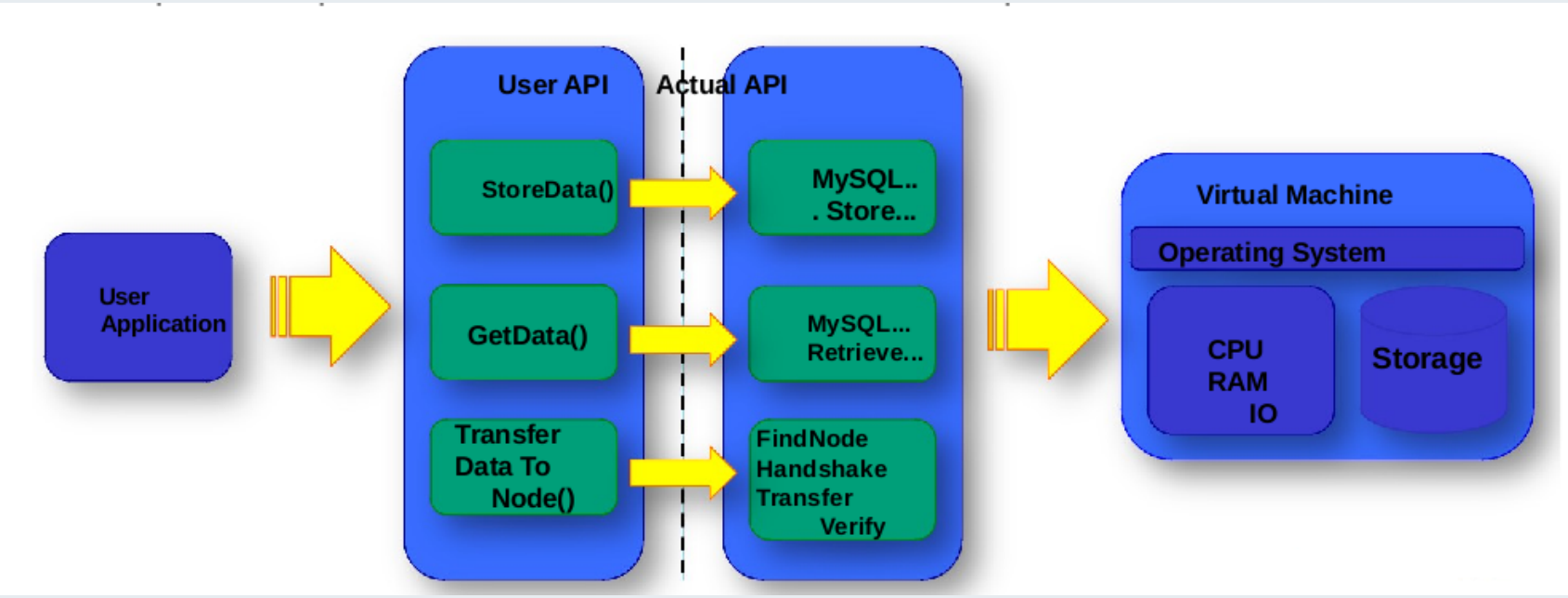
## Comparison of IaaS and PaaS resource usage

- Cloud provider offers a pre-configured, virtualized application server environment to which applications can be deployed by the development staff
- In a PaaS environment it is possible that the vendor has already installed a pre-packaged database application.
- The developer can access it through a well defined API
- Some vendors offer a structured storage solution in their IaaS (i.e.. AWS)



## Offering PaaS as complete compute environment

- Generally accepted definition (Azure, AppEngine)
- Includes storage and IO — no direct access to resources
- Specific implementation details hidden from developer



# PaaS

Typical development cycle:

**Requirements** Choose platform and resources

**Design** Choose application architecture suitable for the platform

**Implement** Write application

**Validation and Verification**

**Test locally** (as far as possible ... resources may not be available locally)

**Debug**

**Deploy** Upload application package to Cloud service

No different than other forms of development; need to apply same disciplines of software engineering

# PaaS service model

## Providers

Google App Engine, Microsoft Azure, Force.com, AT&T Synaptic

## Properties

- Platform for developing scalable applications;
- Test, deploy, host and maintain in one environment;
- Easy integration with Web services and databases.

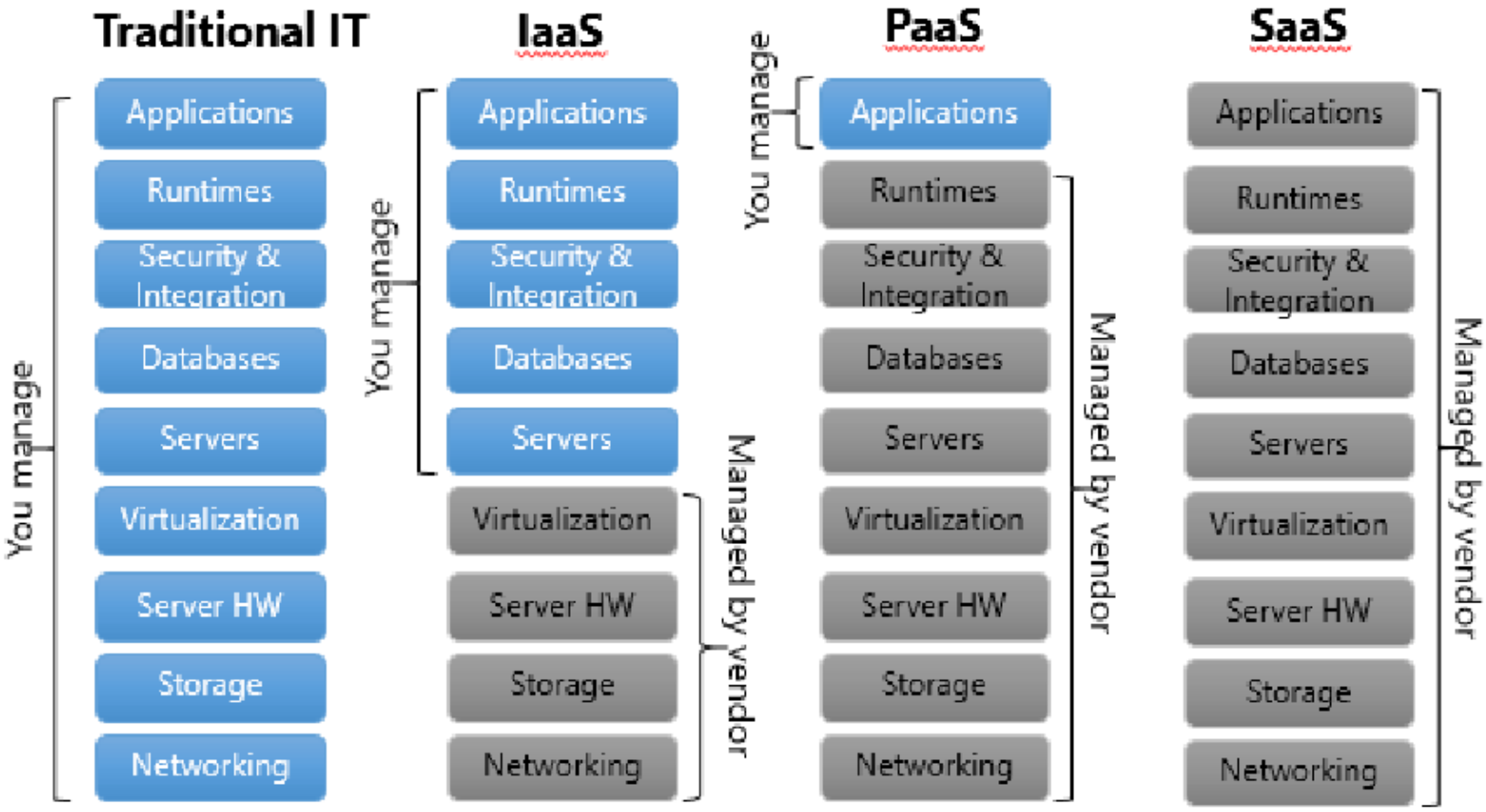
## Access to resources

Consumers have access to application development environment. Tools and libraries can be used to build applications. No control over hardware, control over platform e.g. choose/tune operating system

## Key challenges

- Privacy control;
- Traceability application and data;
- Audit trails;
- API Key Protection.

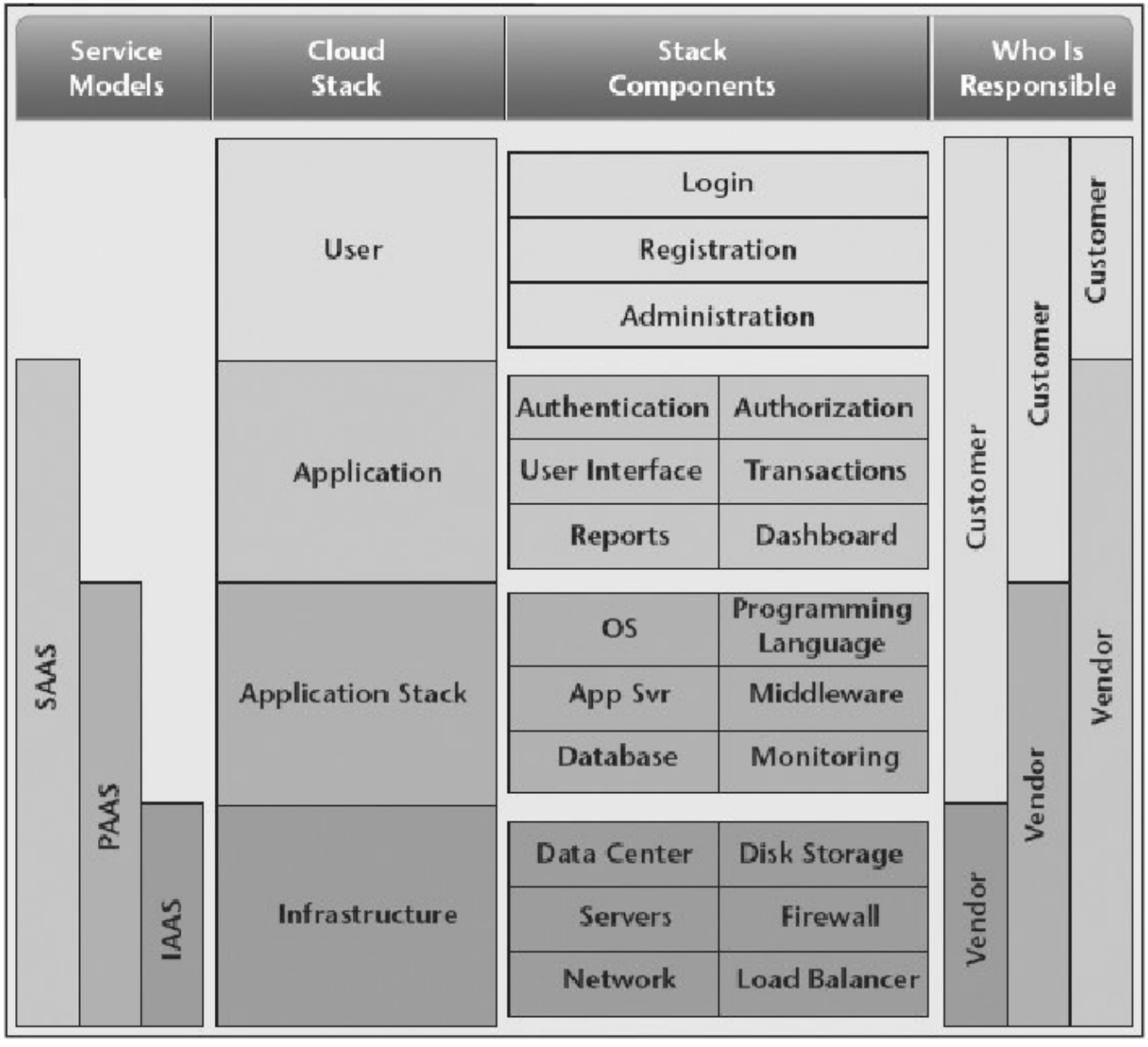
# SaaS


Traditional IT
IaaS
PaaS
SaaS
You manage
You manage
You manage
Managed by vendor
Managed by vendor
Managed by vendor
Applications
Runtimes
Security & Integration
Databases
Servers
Virtualization
Server HW
Storage
Networking

## SaaS — Software as a Service

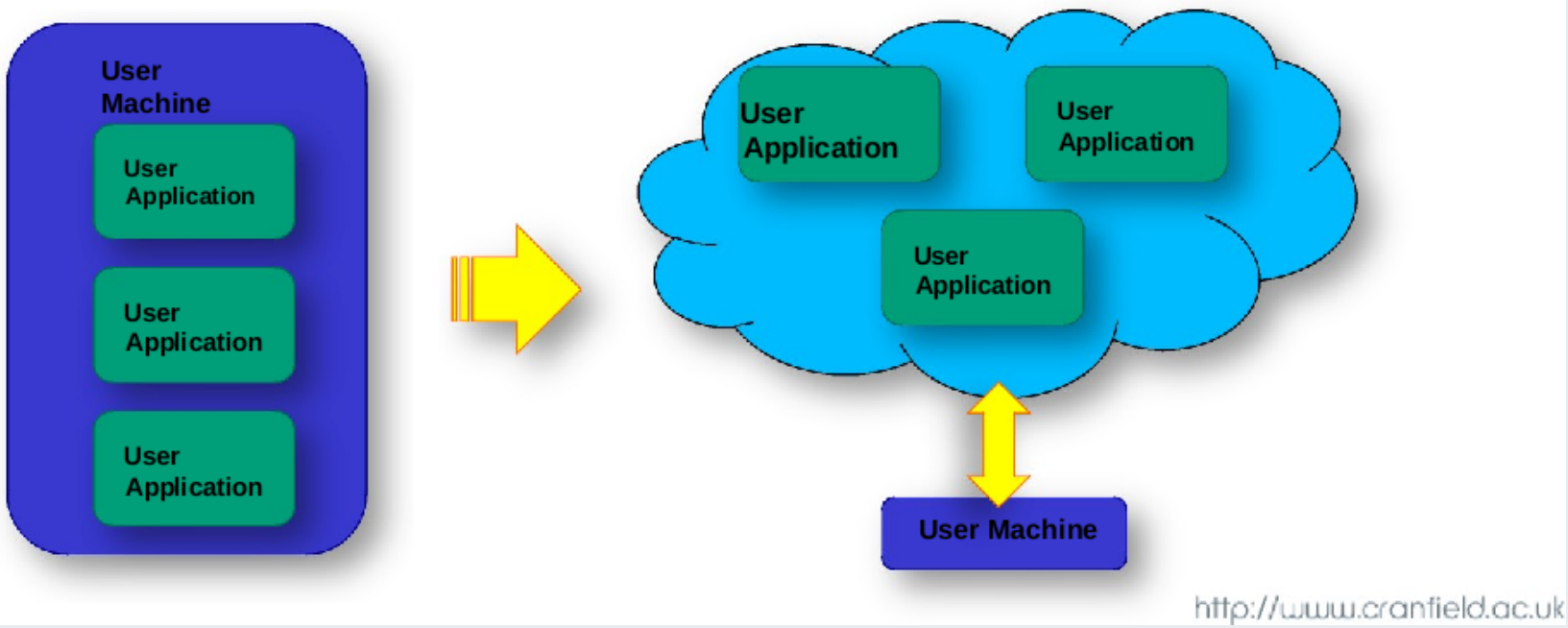
- Pre-packaged services (hardware and software) presented to the user as a complete application solution
- SaaS built on top of a Cloud based IaaS or PaaS
- Includes front end application, often web based allowing significant flexibility in access
- In the background, provides storage and communication services, usually unnoticed by the end-user
- Simplified software installation and maintenance
- Control over versioning
- SaaS can be implemented without Cloud, but requires larger capital expenditure

# Cloud Services SaaS



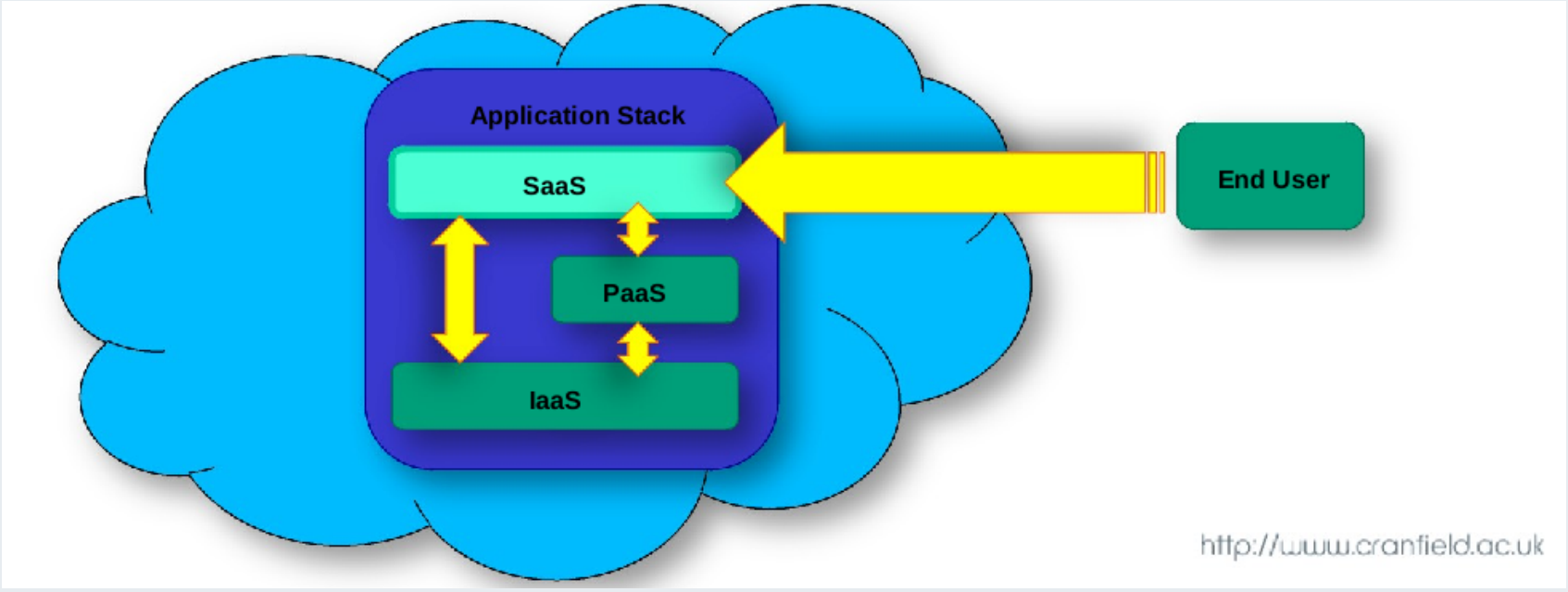
## Pre-packaged applications offered as services

- Traditional applications are installed on a PC
- SaaS delivered over the network rather than on individual machines



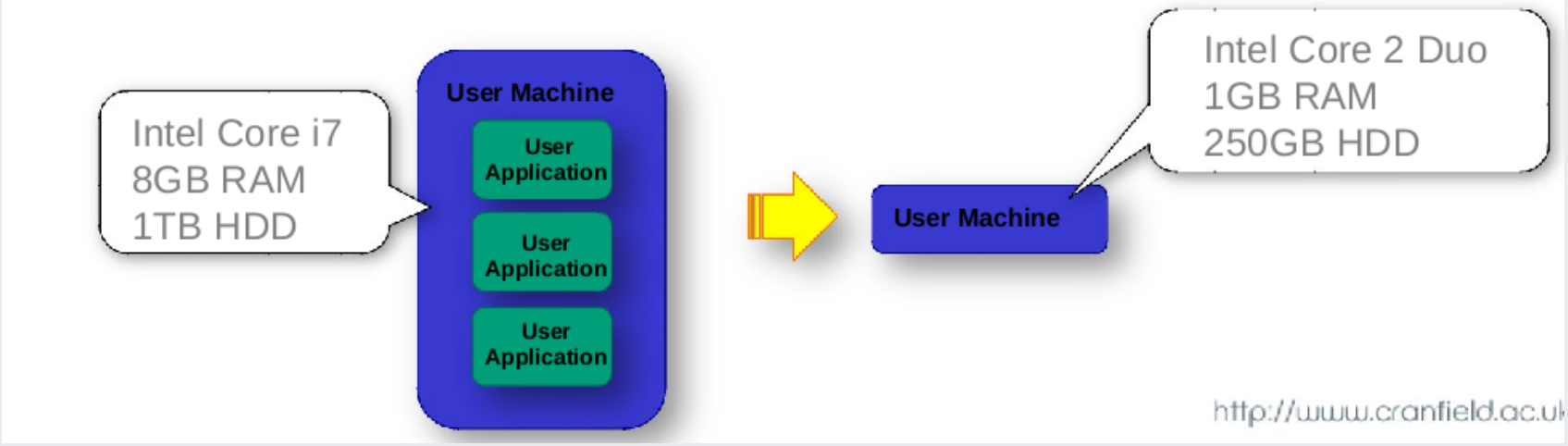
## SaaS built on top of IaaS or PaaS

- Application developer chooses most appropriate abstraction level
- End users of SaaS applications have no access to lower levels and no knowledge of implementation details



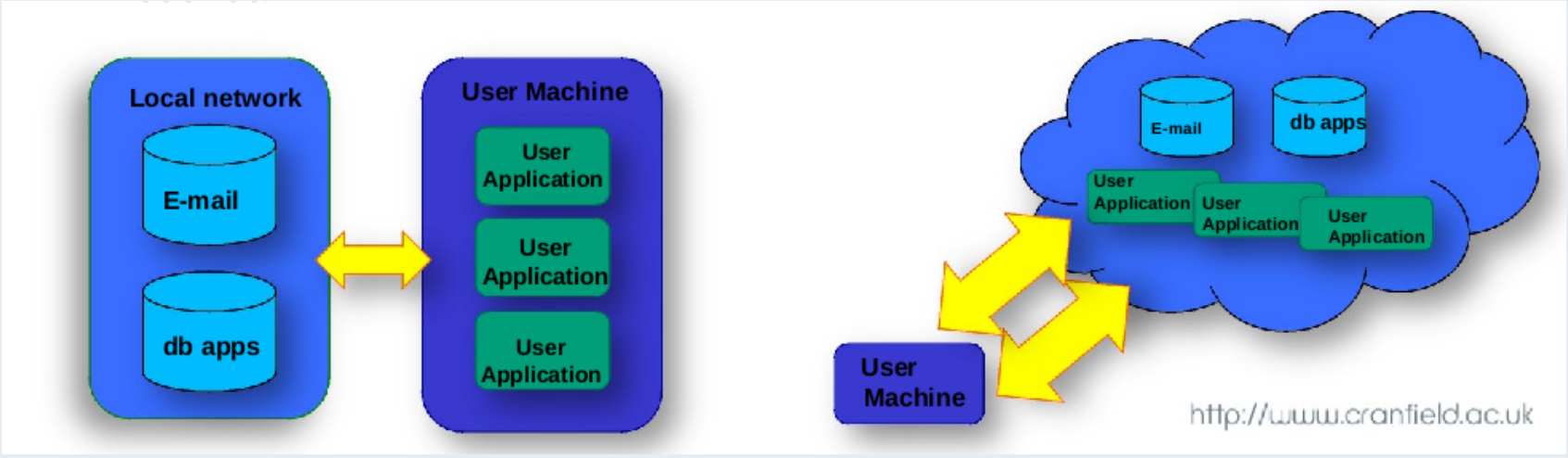
## Potential reduction of computational overheads

- Application executes on Cloud, transferring computational load away from local machine
- Only the results are presented on local machine
- Lean client model, similar to mainframe model
- Reduced capital outlay for initial hardware installation and on-going maintenance/upgrade costs



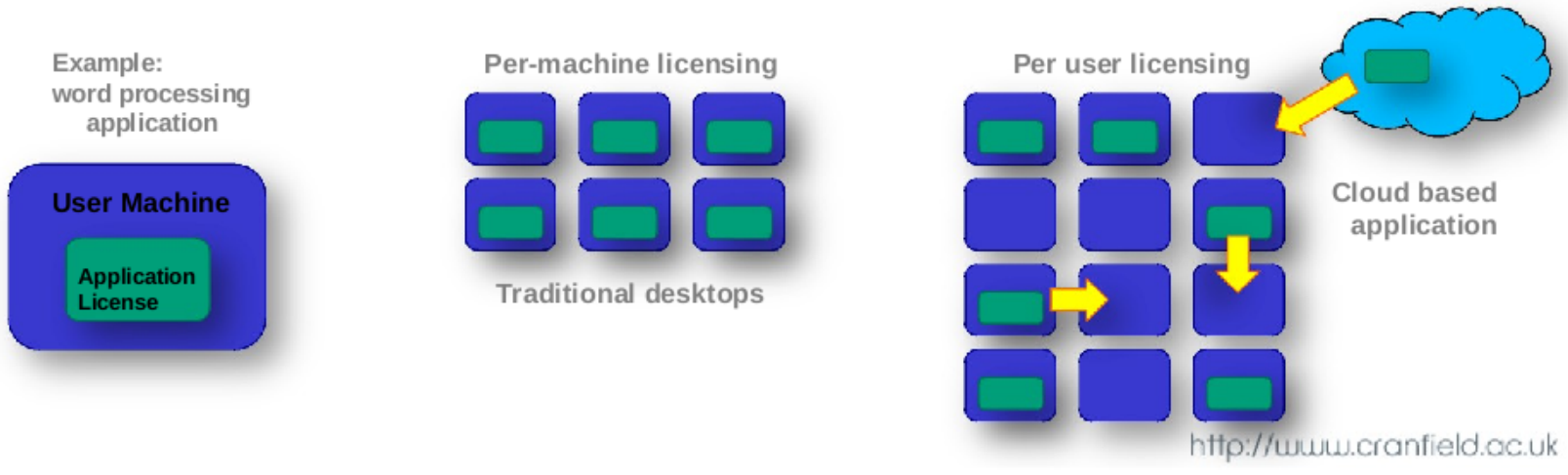
## Increased reliance on network connections

- Many in-house applications access data from central on-site database, so internal network performance and reliability has always been important
- Examples: e-mail, document control, database applications
- Reliable connection to external site is required
- Increased costs associated with network connection performance and redundancy



## Pre-packaged application services

- Sometimes referred to as Software on Demand because of its minimal installation and maintenance requirements
- Lowers the cost of ownership associated with traditional software licensing
- Cloud pay-per-use model allows reduced expenditure at the point of delivery
- Similar to pay-per-user licensing rather than pay-per-machine licensing



# SaaS

## Providers

NetSuite (ERP), Taleo (HR), SalesForce (CRM), Google Docs, Microsoft Office Live

## Properties

- Web interface:
- No installation requred
- Shared software
- Only data onership
- Pay per use

## Access to resources

Consumers have only access to the software as a service. No control over tuning of software, operating system and hardware resources

## Key challenges

- Credential management;
- Usage and accountablity;
- Data traceability and security;
- API Key Protection.

# Application Architecture

## Many IaaS applications are based on web technologies:

- Web browser used as the user front end GUI
- Use web technologies to implement user experience RESTful or RPC
- Some limitations when compared with native applications, but improvements in standards enrich the user experience (ie HTML5)
- Ease of access to key applications
- Reduced application installation and maintenance problems (dependencies, versioning)

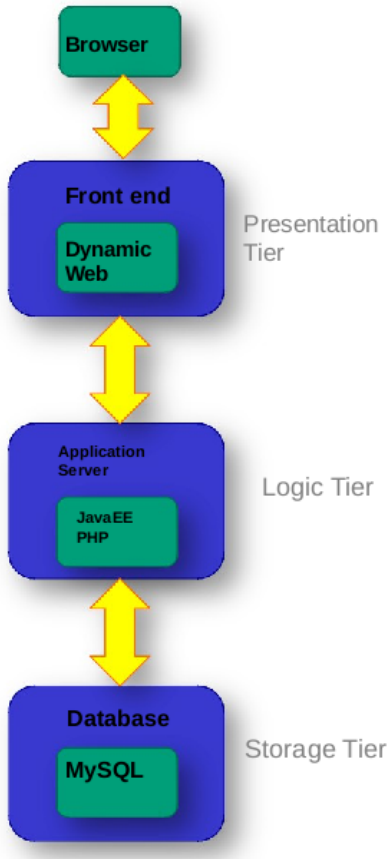
## Enforces clear separation of concerns

- GUI and backend processing separated by location and technologies
- Good software engineering practice

# Application Architecture

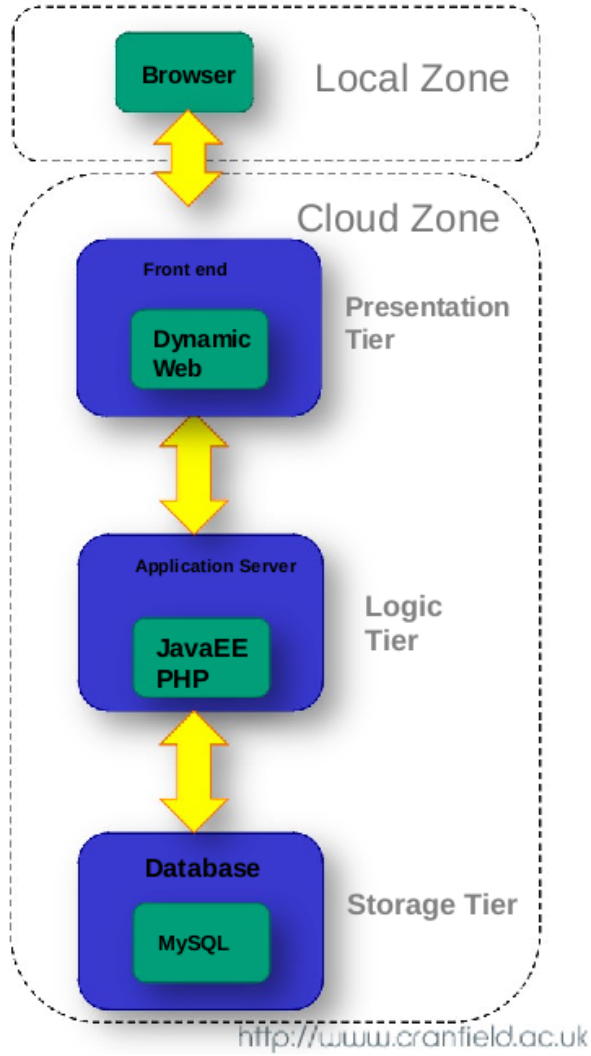
## Separation of Concerns for web applications

- Typical architectural pattern uses a 3 tier design
- The three tiers are usually presentation, logic and Application storage (or data)
- Each layer is responsible for a specific task and communicates only with the layer above or below it
- For example, the presentation layer cannot directly access application data without communicating through the logic layer
- Increases security and application robustness



# Application Architecture

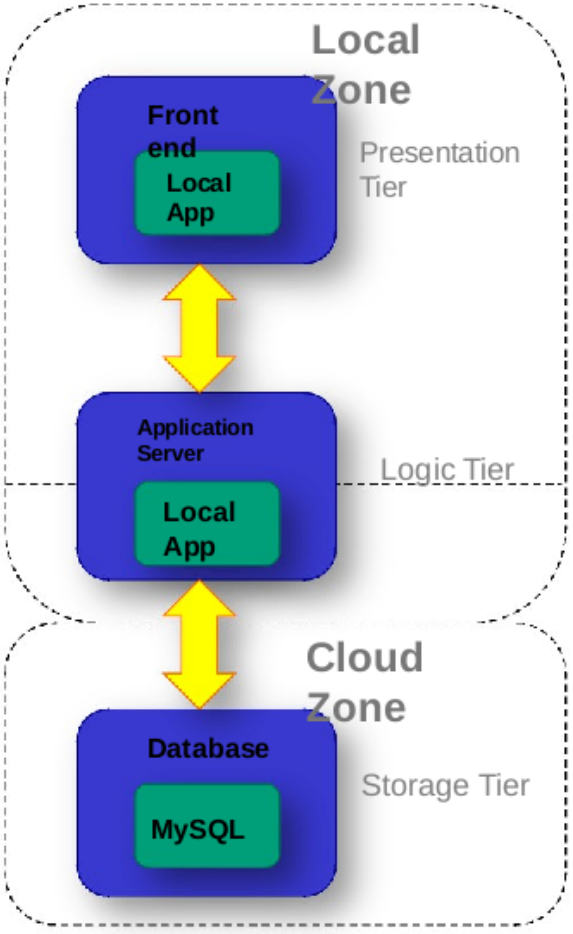
For typical Cloud based applications, the separation is well defined into local and Cloud zones



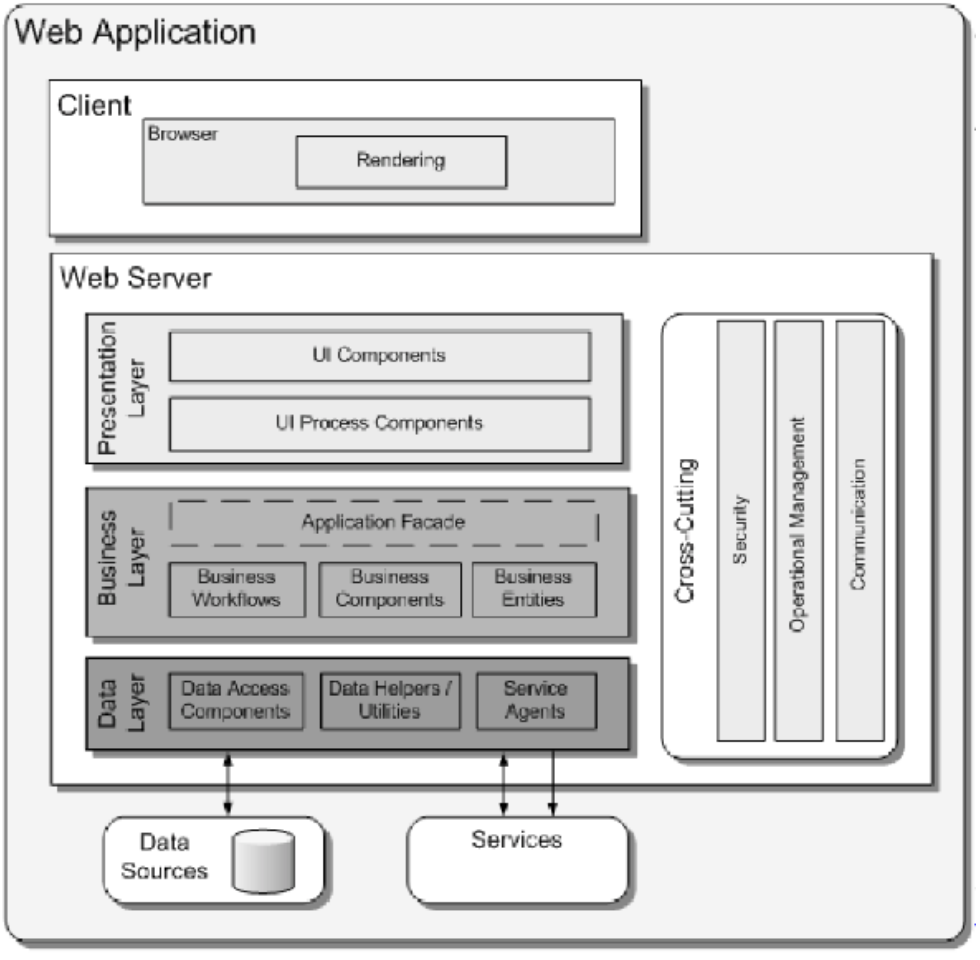
# Application Architecture

## Storage as a service

- Applications are executed locally but access data on the Cloud
- An example might be an on-line backup system, such as DropBox
- Logic tier can be shared between on-line and local portions



# Web Application Architecture

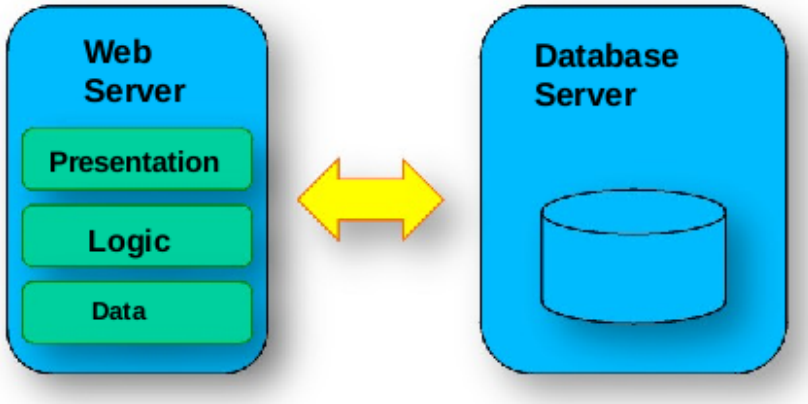



## Typical Web Application Architecture

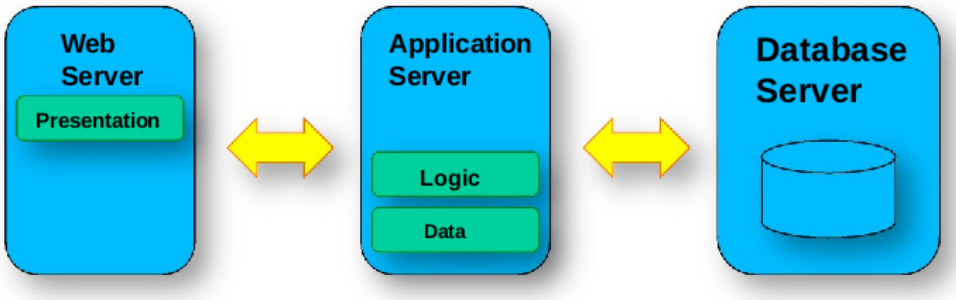
- Microsoft Patterns and Practices Guide
- Three layer architecture
- Layered architecture: security and loose coupling
- Cross-cutting concerns (security)
- Communication between layers is important — use caching

**Source**: `http://msdn.microsoft.com/en-us/library/dd673617.aspx`

# Web Application Architecture




- Non-distributed
- All three tiers reside on the same physical server




- Distributed
- Tiers located on different servers

# Application Architecture

## Front-end does not have to be browser based

- We can implement local applications
- Microsoft Azure platform provides development and run-time environment based on .Net framework
- User can interact with application through rich GUI
- Back-end processing can occur on the Cloud infrastructure
- Strong coupling between desktop and Cloud (Azure) APIs ensure a seamless user experience
- Loses advantage of lower installation and maintenance costs
- Ties the application specifically to Azure — other platforms offer little portability anyway!

# Application Architecture

## Software deployment considerably simplified

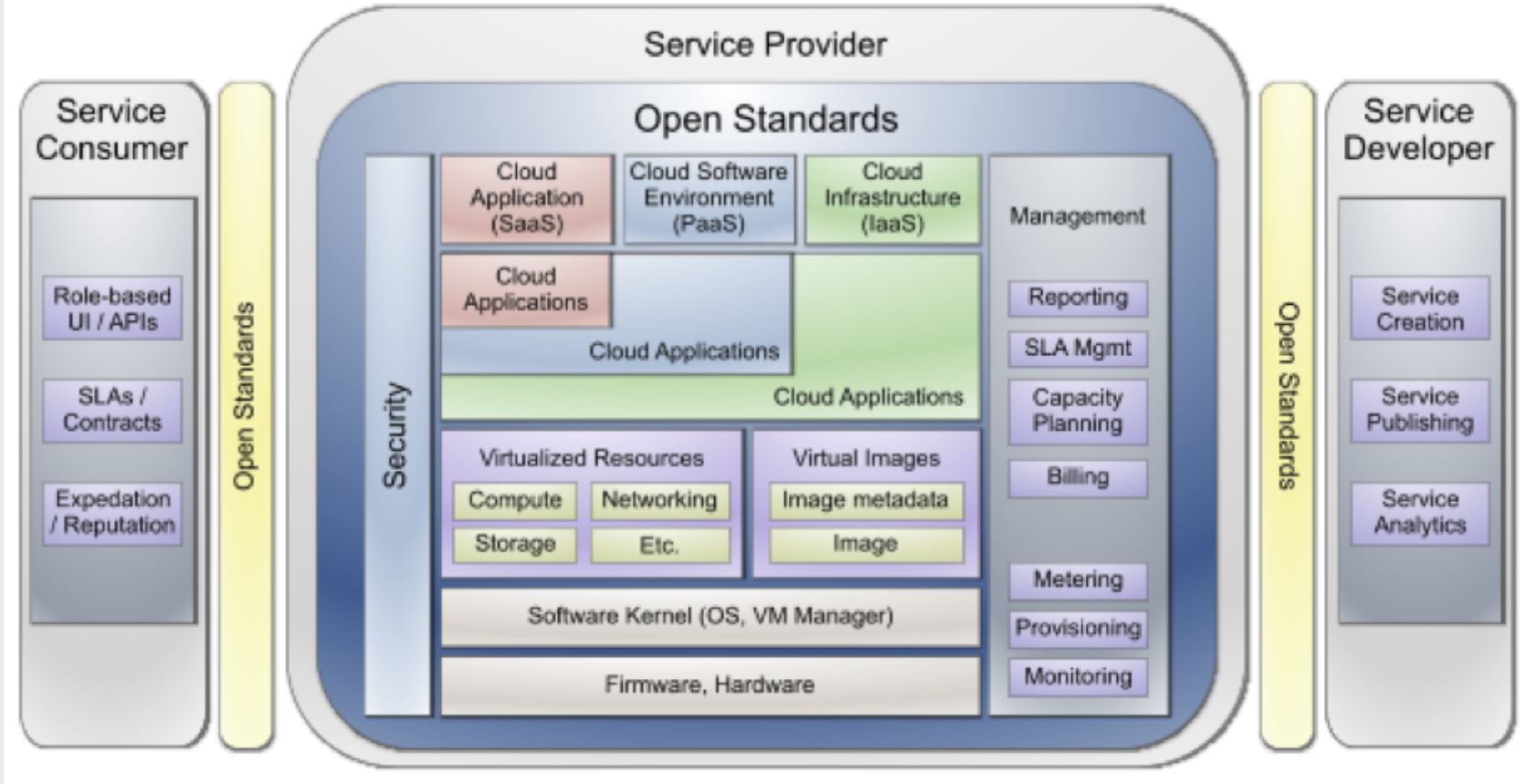
- Software assets located centrally rather than on each machine
- No physical installation no need to physically visit each machine
- Dependency issues are reduced
- Reduces opportunities for users to "explore" the installation (and break it!)

## Maintenance costs are reduced

- Upgraded versions of application are automatically available to the user
- Reduced down time — no rolling upgrade programmes needed
- Reduced roll-out costs
- Reduces issues connected with multiple versions being used within an organisation

# Application Architecture

## Combining IaaS, PaaS and SaaS



# Deployment Models

**Public Cloud:** compute resources are offered by an external vendor

**Private Cloud:** built for exclusive use by one client, providing more control of data, security and quality of service; services offered by internal data centres

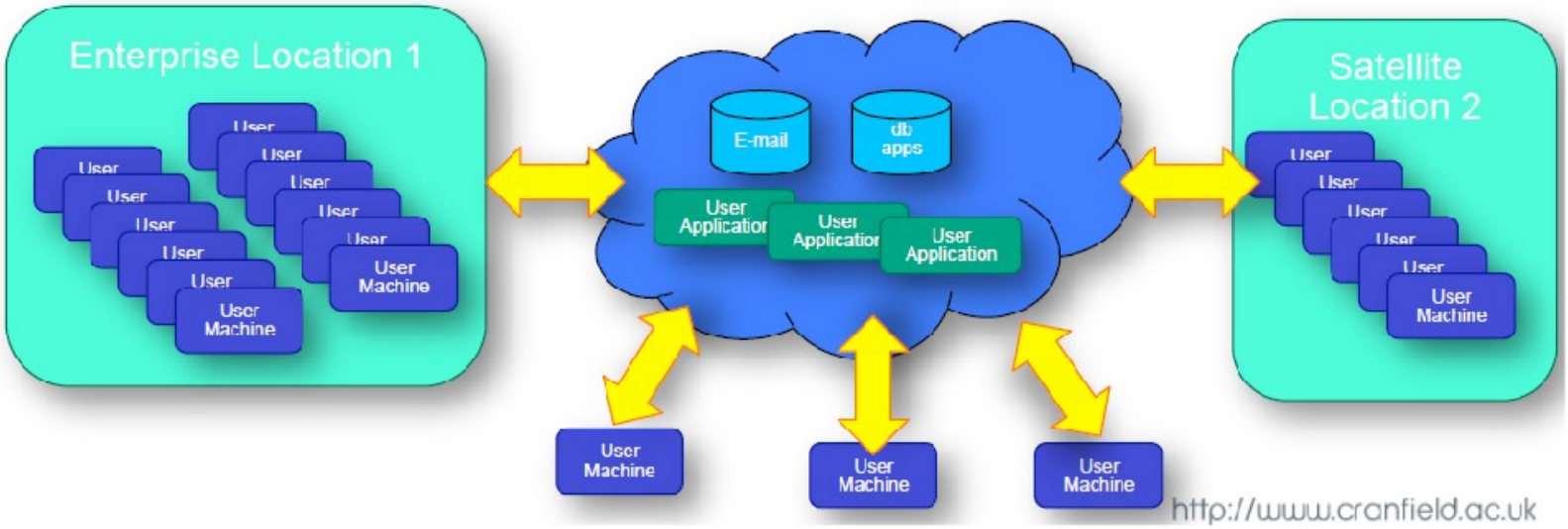
**Hybrid Clouds:** Using private and public Cloud resources

**Community Clouds:** A public cloud available to a select group of organisations

# Deployment Models

## Public Cloud

- Cloud is offered by an external vendor, mostly on a PAYG basis
- Connectivity is achieved via the Internet
- Large, consolidated data centres offer effective elasticity — infinite resource



# Deployment Models

Who are the potential audiences for Public Clouds?

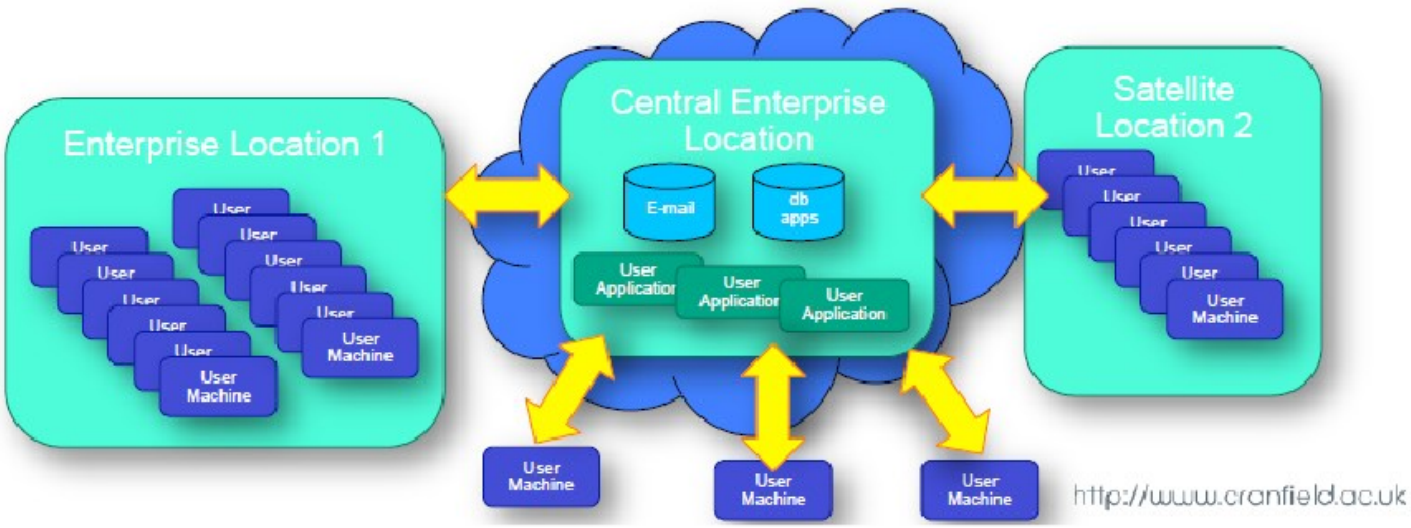
- Clouds are for those who do not want to manage their own hardware
- Very high-end HPC — Cloud will not replace this
- Enterprise level cluster users — public Clouds can be used to improve manageability and access
- Departmental occasional usage — no requirement for sustained local HPC resources

Issues of data security and sovereignty (see later)

# Deployment Models

## Private Cloud

- Services are offered from within an organisation
- Hardware is managed internally
- Elasticity is still available but may be limited in scope
- Connectivity is managed by the organisation



# Deployment Models

Who are the potential audiences for Private Clouds?

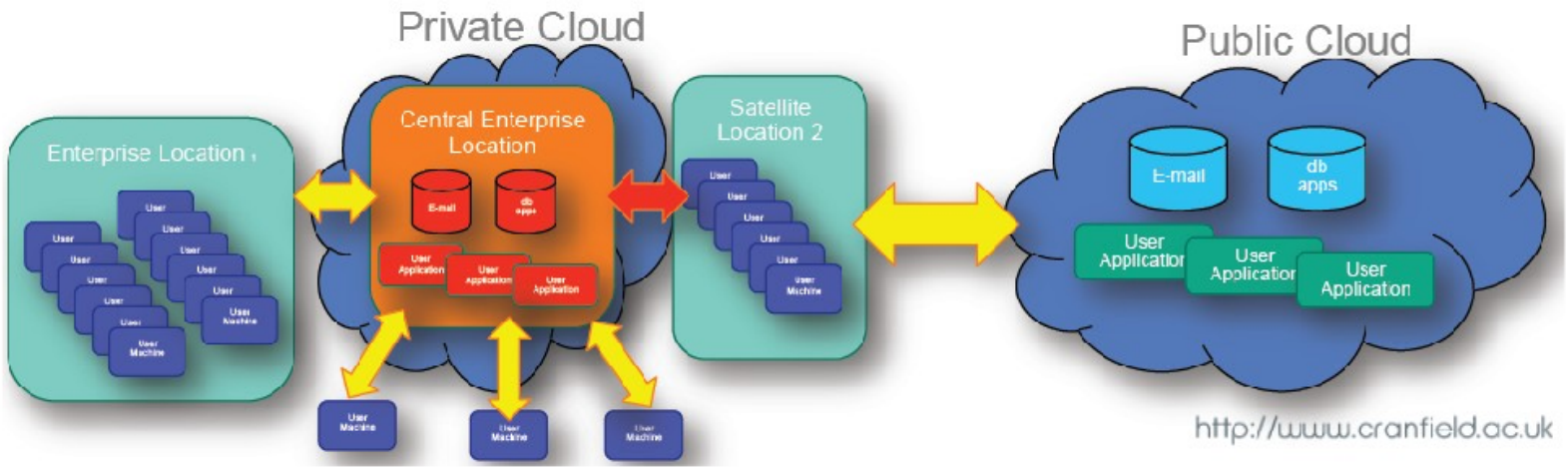
- Organisations who want to streamline their existing IT infrastructure
- Introduces flexibility, redundancy and resilience to enterprise
- Offers a relaxation of network bandwidth constraints
- May be more cost effective than public cloud

Not as prone to legal and security issues as public clouds

- Data remains within the jurisdiction of the enterprise
- Information is as secure as it was on existing IT systems

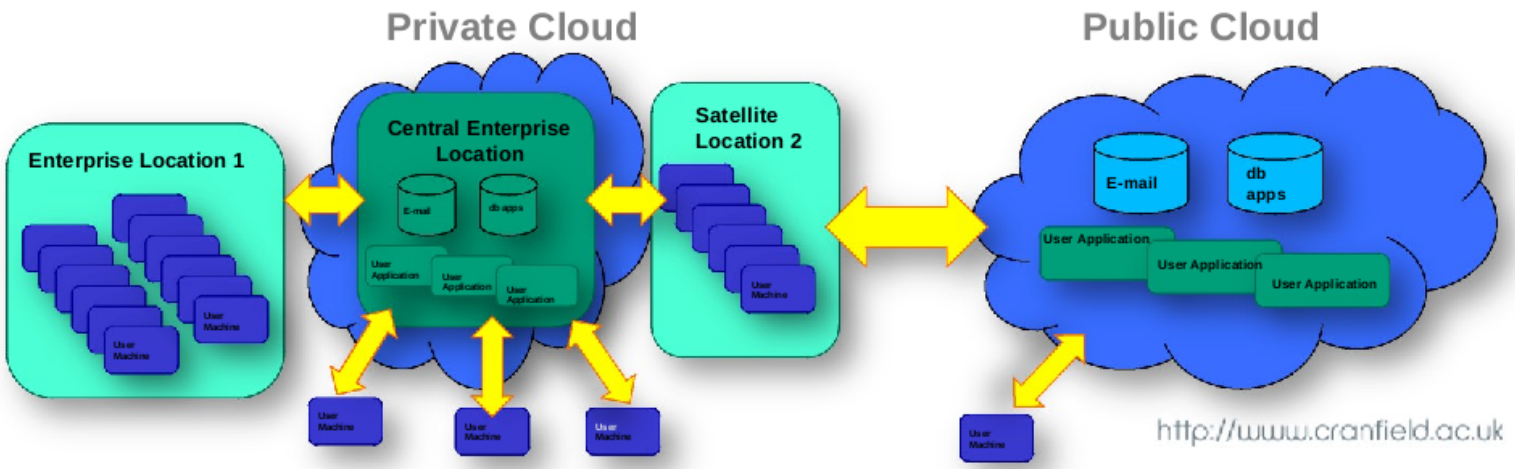
## Hybrid Cloud — Scenario 1

- Mix of public and private cloud resources
- Primary resource offering is private
- If demand exceeds ability to supply, users can be transferred to a public cloud resource — Cloud Bursting
- Problems of data access and integrity across public and private domains



## Hybrid Cloud — Scenario 2

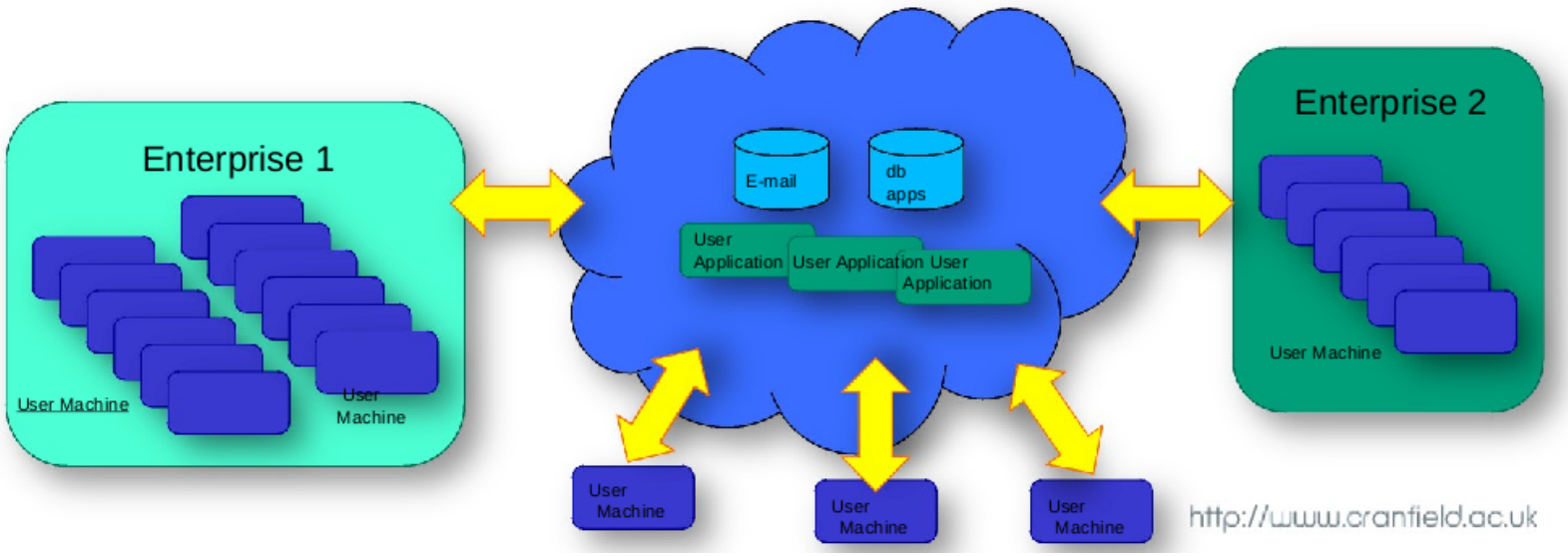
- Mix of public and private cloud resources
- Critical or sensitive operations performed in-house
- Less sensitive functionality can be moved to public Cloud
- Remote data access can be moved to public Cloud



# Deployment Models

## Community Cloud

- Services and data are centralised
- Multiple co-operating organisations can access these shared resources
- Shared projects or requirements
- Virtual Organisations



# Cloud Resources

Resources can be repackaged and presented as services

- Compute
- Storage — Very large datasets
- Communications
- Request Driven Provisioning

Services do not have to offer a web based front-end (ex: Dropbox offers both a native application and a web based interface)

# Opportunities and Motivations

Some general examples of application areas

- Mobile Interactive Applications
- CC provides the backend infrastructure
- Relieves the processing burden from the device
- Parallel Batch Processing
- HPC or enterprise applications (ie. Payroll)
- Business Analytics
- Marketing analysis
- Historical data analysis
- Business planning
- Extension of desktop applications
- Seamless integration of desktop applications into Cloud
- Rich APIs allow implementation of diverse cross-over between local and remote processing

# Opportunities and Obstacles

Availability of Service

- Use Multiple Cloud Providers
- Use Elasticity to Prevent DDOS

Data Lock-In

- Standardize APIs
- Compatible SW to enable Surge Computing (Cloud Bursting)

Data Confidentiality and Auditability

- Deploy Encryption, VLANs, Firewalls
- Geographical Data Storage

# Opportunities and Obstacles

Data Transfer Bottlenecks

- FedExing Disks
- Data Backup/Archival
- Higher BW Switches

Performance Unpredictability

- Improved VM Support
- Gang Schedule VMs

# Opportunities and Obstacles

**Scalable Storage**

**Bugs in Large Distributed Systems** $\Rightarrow$ Invent Debugger that relies on Distributed VMs

**Scaling Quickly:** Invent Auto-Scaler that relies on ML; Snapshots for Conservation

**Reputation Fate Sharing**

$\Rightarrow$ Offer reputation-guarding services like those for email

**Software Licensing**

- Pay-for-use licenses;
- Bulk use sales

# Several distributed computing technologies

- Virtualization
- Web services
- Client-Server
- Distributed Computing
- Meta Computing
- Cluster Computing
- Grid Computing
- Peer-to-Peer
- Cloud Computing
- Ubiquitous Computing/Pervasive Computing

# Recommended Reading


- Wang et al, "Cloud Computing: a Perspective Study", New Generation Computing, 28(2), 137-146, 2010
- I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," 2008 Grid Computing Environments Workshop, 2008
- K.A. Delic, M.A. Walker, "Emergence of the Academic Computing Clouds", Ubiquity, 2008(8)
- B. Adler, "Grid Computing Applications in the Cloud", RightScale Inc, Technical White Paper, 2010
- J. Napper, P. Bientinesi, "Can cloud computing reach the top500?", UCHPC-MAW '09: Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop, Proceedings, 17-20, 2009
- L. Wang, J. Tao, M. Kunze, A.C. Castellanos, D. Kramer, W. Karl, "Scientific Cloud Computing: Early Definition and Experience", High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on, 825–830, 2008

# Vendors and Platforms

## Commercial Vendors:

- Amazon AWS `https://aws.amazon.com/`
- Google App Engine
  `http://code.google.com/appengine/`
- Microsoft Azure
  `http://www.microsoft.com/windowsazure/`

## Open Source Environments

- Eucalyptus `http://www.eucalyptus.com/`
- Nimbus `http://www.nimbusproject.org/`
- Openstack `http://openstack.org/`