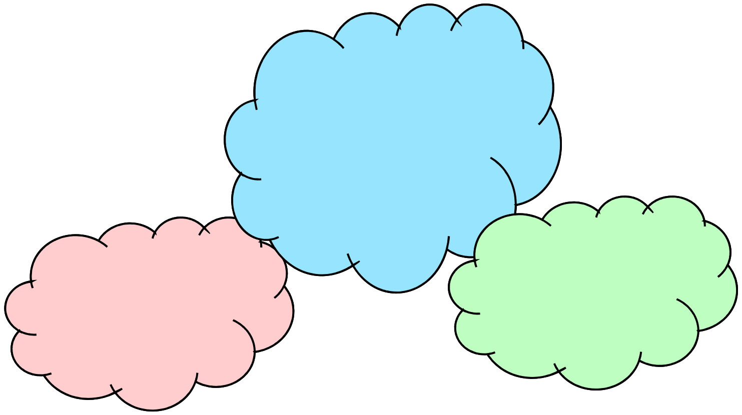


How to Port an Application Between Clouds?

Dana Petcu

Content

- Part 1: Introduction to Cloud Computing
- Part 2: Portability and Interoperability issues
- Part 3: mOSAIC generalities
- Part 4: Demo

How to Port an Application Between Clouds?

Part I: Introduction to Cloud Computing

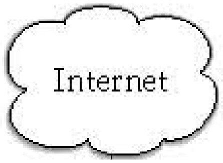
Preliminaries

Content

- Definitions
- Examples
- European efforts in R&D

Symbols and promise

- *Gets its name as a metaphor for the Internet*.
  - Typically, the Internet is represented in network diagrams as a cloud
  - Cloud icon represents “all that other stuff” that makes the network work



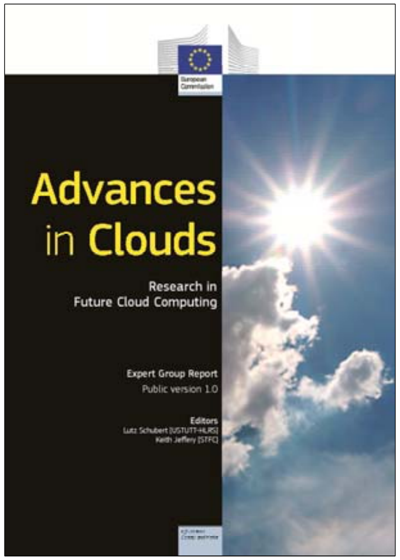
- *Promise*:
  - To cut operational and capital costs
  - Let IT departments focus on strategic projects instead of keeping the datacenter running

[US] NIST definition

Cloud computing
is a *pay-per-use model*
for enabling available, convenient, on-demand
network access
to a shared pool of
configurable computing resources
(e.g.networks, servers, storage, appls, services)
that can be *rapidly provisioned and released*
with minimal management effort
or service-provider interaction.

[EC] Expert Group on Cloud Computing

An environment can be called “CLOUDified”,
if it enables a large dynamic number of users
to access and share the same resource types,
respectively service,
whereby maintaining resource utilisation and costs
by dynamically reacting to changes
in environmental conditions,
such as load, number of users,
size of data



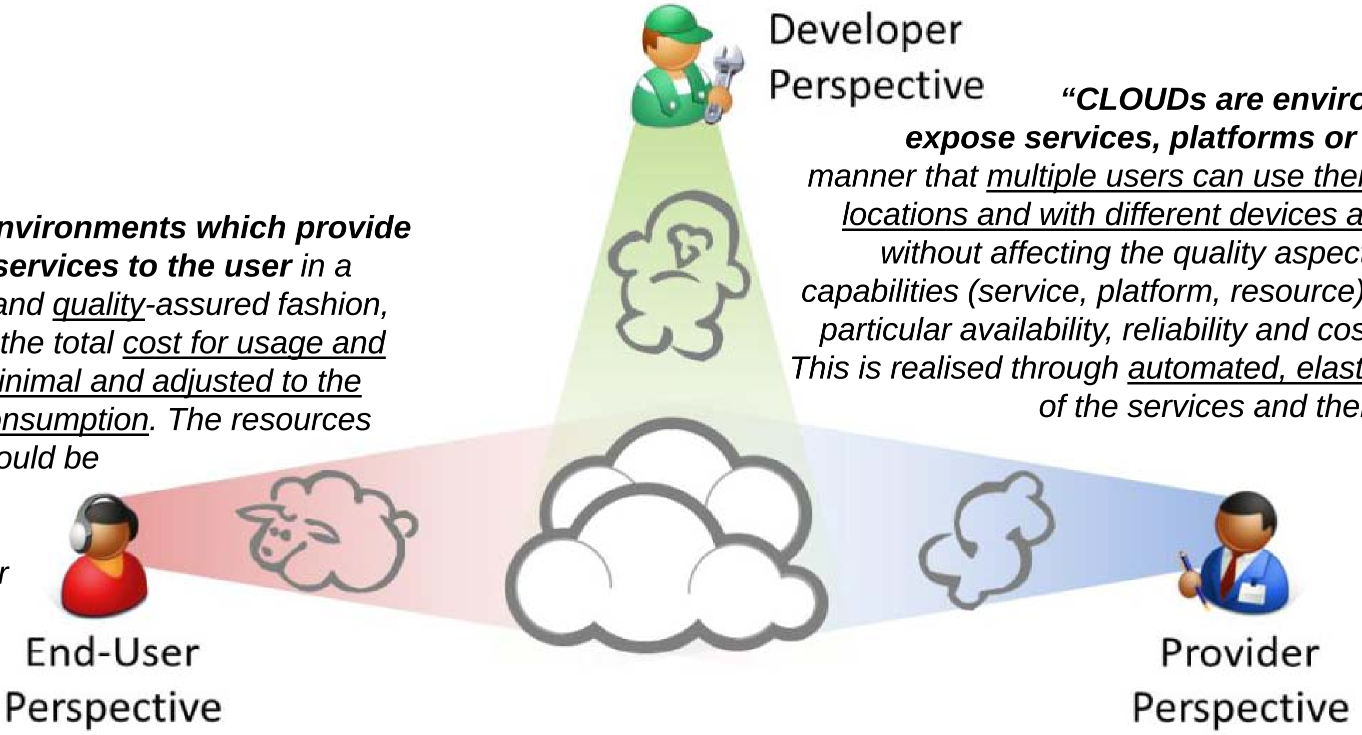
Advances in Clouds

Research in Future Cloud Computing

Expert Group Report
Public version 1.0

Editors
Lutz Schubert [USTUTT-HLRS]
Keith Jeffery [STFC]

Different views [from Expert Group report]



“CLOUDs are environments which provide resources and services to the user *in a highly available and quality-assured fashion, thereby keeping the total cost for usage and administration minimal and adjusted to the actual level of consumption. The resources and services should be accessible for a principally unlimited number of customers from different locations and with different devices with minimal effort and minimal impact on quality. The environment should thereby adhere to security and privacy regulations of the end-user, in so far as they can be met by the internet of services.”*

“CLOUDs are environments which expose services, platforms or resources in *a manner that multiple users can use them from different locations and with different devices at the same time without affecting the quality aspects of the offered capabilities (service, platform, resource) - this means in particular availability, reliability and cost-effectiveness. This is realised through automated, elastic management of the services and their environment.”*

“CLOUDs are dynamic (resource) environment that guarantee availability, reliability and related quality *aspects through automated, elastic management of the hosted services – the services can thereby consist in a platform, a service, or the infrastructure itself (P/S/IaaS). The automated management thereby aims at optimising the overall resource utilisation whilst maintaining the quality constraints.”*

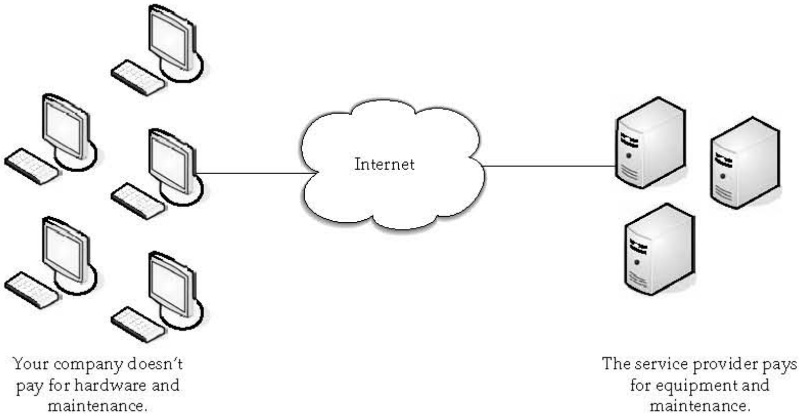
Key characteristics [NIST vs. EC Experts]

1. *On-demand self-service*
2. *Ubiquitous network access*
3. *Location-independent resource pooling*
4. *Rapid elasticity*
5. *Pay per use*

Technical	Business /Economic	Social / Legal	Other
Elasticity / Scalability	Outsourcing	Security	Multi-Tenancy
Virtualisation	Pay per use	Provenance	Ease of Use
Agility & Adaptability	Resource utilisation	Privacy	
Availability	Energy efficiency		
Data Management	Metering		
Reliability			
Programmability			

Benefits vs. drawbacks

- Delegation: another company hosts your appl (or suite of appls)
  - they handle the costs of servers,
  - they manage the software updates,
  - you pay for the service

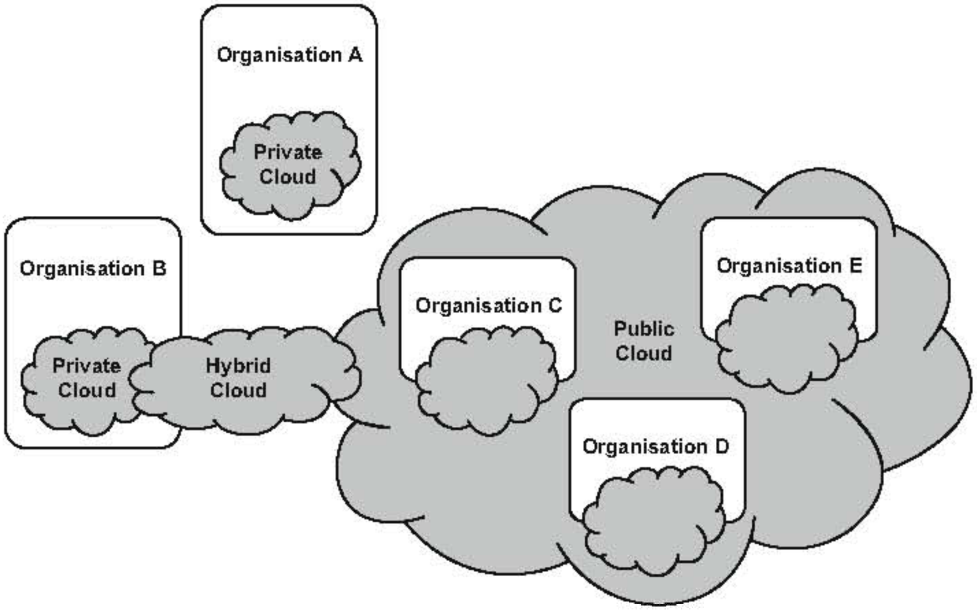

Internet
Your company doesn't pay for hardware and maintenance.
The service provider pays for equipment and maintenance.

- Drawbacks:
  - On-line
  - Privacy and security?
  - Difficult to integrate geographically dispersed components

- Reduced implementation and maintenance costs
- Increased mobility for a global workforce
- Flexible and scalable infrastructures
- Quick time to market
- IT department transformation (focus on innovation vs. maintenance and implementation)
- “Greening” of the data center
- Increased availability of high-performance applications to small/medium-sized businesses

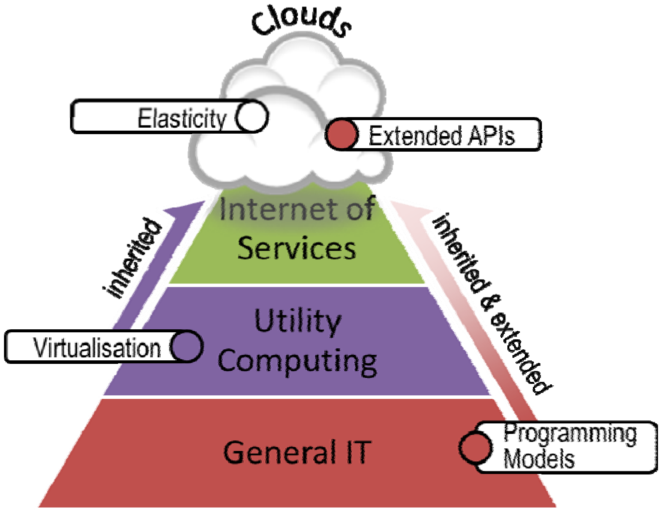
Types of Clouds

- *Private Cloud*
- *Community Cloud*
- *Public Cloud*
- *Hybrid Cloud*

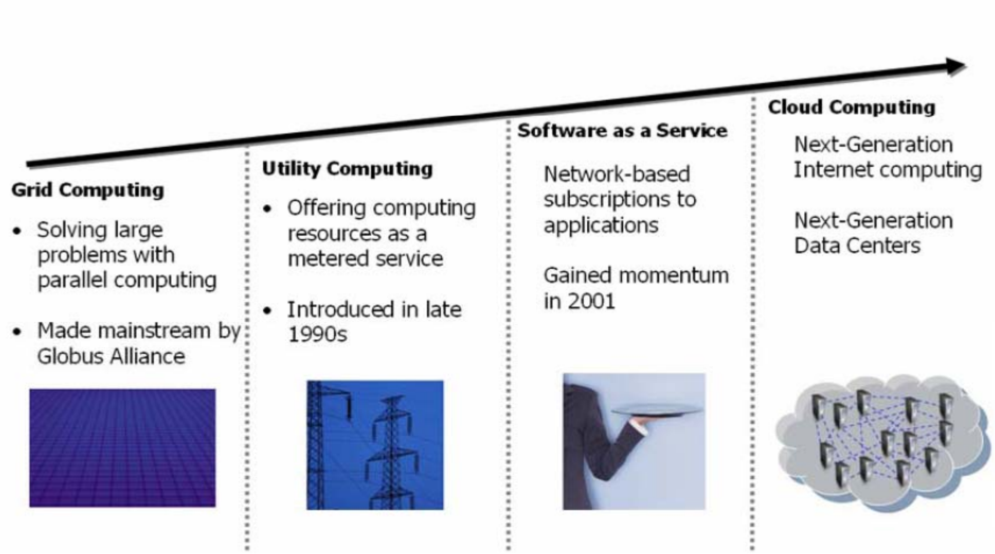


Evolution

- EC Expert Group vision:



- Entreprise vision:



Grid Computing

- Solving large problems with parallel computing
- Made mainstream by Globus Alliance

Utility Computing

- Offering computing resources as a metered service
- Introduced in late 1990s

Software as a Service

Network-based subscriptions to applications

Gained momentum in 2001

Cloud Computing

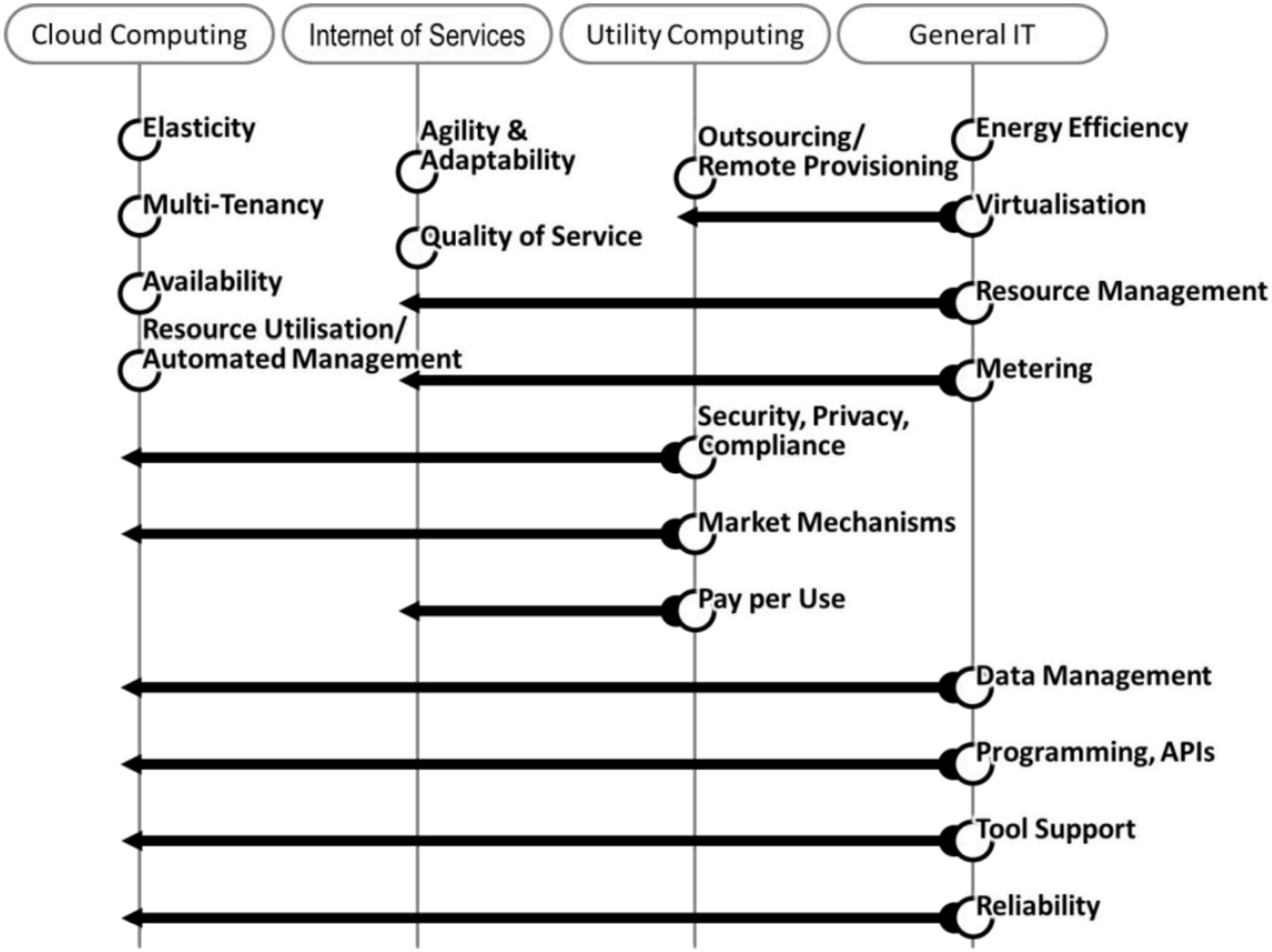
Next-Generation Internet computing

Next-Generation Data Centers

Main differences:

1. Lower entry barriers
2. Multitenancy
3. Reliability
4. Elasticity

Relationships with other concepts [from Expert Group Report]



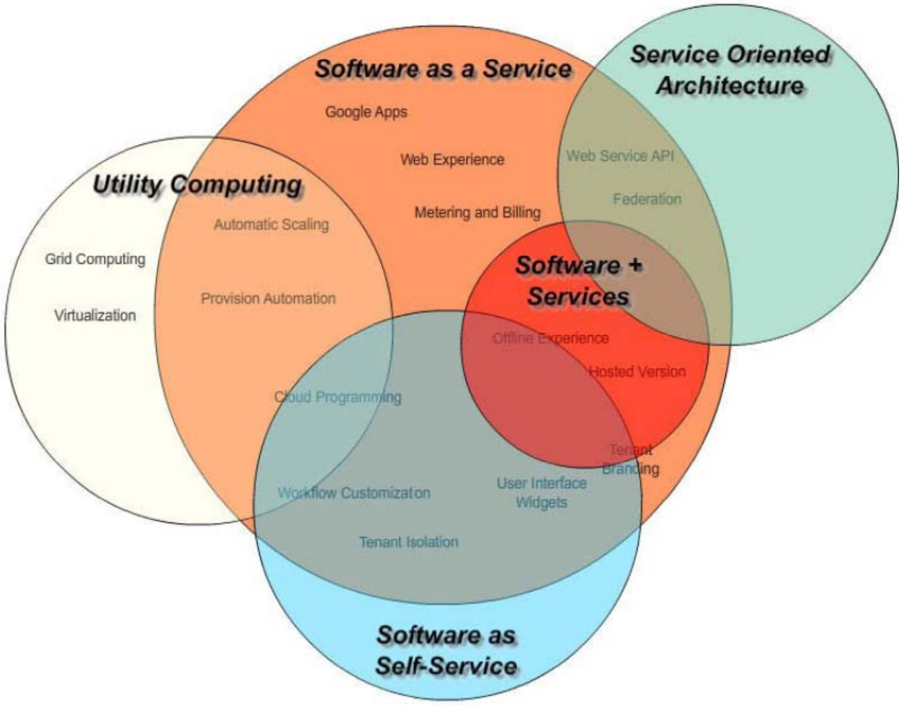
Services in Cloud computing

- Service in CC:
  - the concept of being able to use reusable, fine-grained components across a vendor’s network.
  - “**as a service**.”

- Service delivery models:
  - **Software as a Service** (SaaS)
  - **Platform as a Service** (PaaS)
  - **Infrastructure as a Service** (IaaS)

Software as a Service (SaaS)

- Appl hosted as a service to customers who access it via the Internet
- Opposite to Software-as-a-Product
- Thousands of customers using a multiuser architecture
- For:
  - Software performing a simple task without interact. with other systems
  - For customers with need of high-powered appls
- Ex:
  - Google Docs, Maps, Gmail, Calendar; Miccrosoft Office Live; Salesforce SFA
- Appls include
  - Customer resource management (CRM)
  - Video conferencing
  - IT service management
  - Accounting
  - Web analytics
  - Web content management



http://www.theartofservice.net/UserFiles/Flash/cloud_computing.swf

Platform as a Service (PaaS)

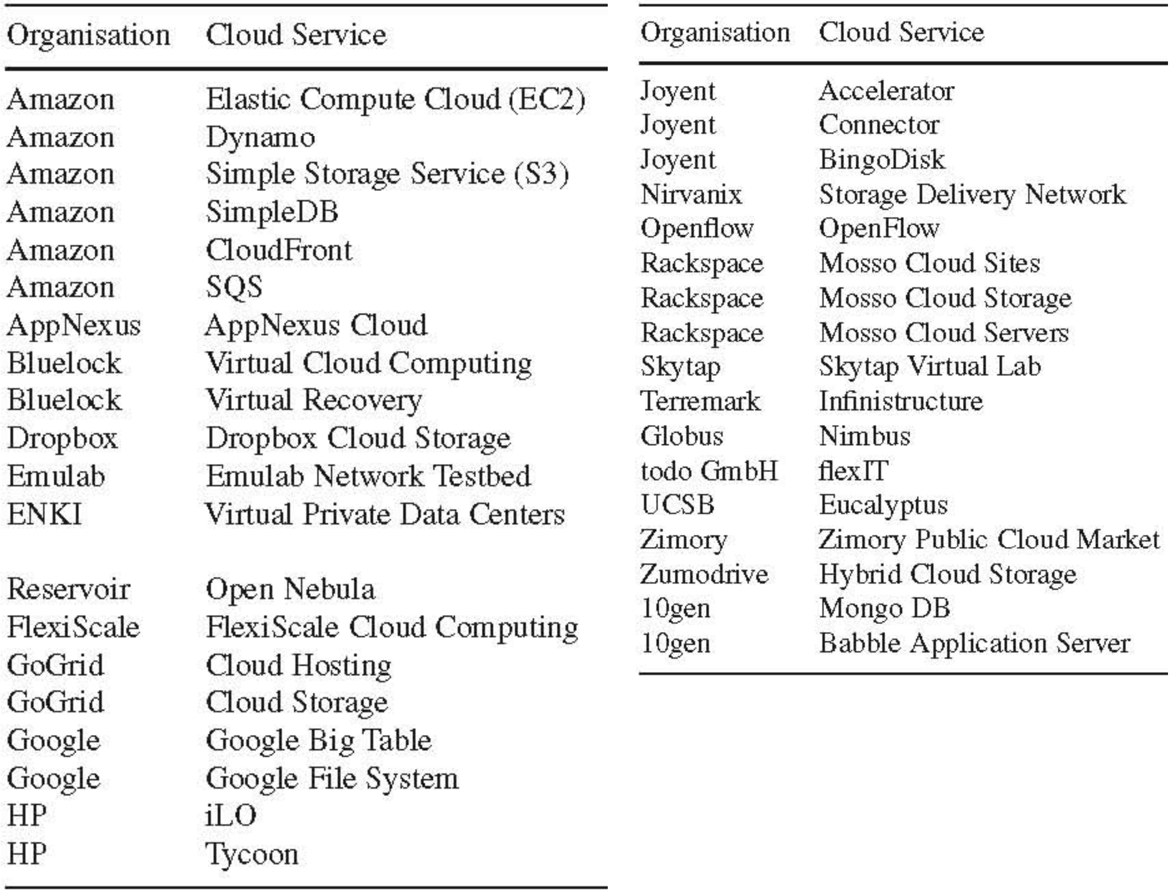
- Known also as Cloudware
- Supplies resources required to build appls and services completely from the Internet, without having to download or install software
- Services include:
  - appl design, development, testing, deployment, and hosting.
  - team collaboration, web service integration, database integration, security, scalability, storage, state management, and versioning
- Delivers a platform from which to work rather than an appl to work with
  - Offer APIs that enable developers to exploit functionality over the Internet, rather than delivering full-blown appls
  - Delivers development environments to programmers, analysts, & software engineers as a service

PaaS

- APIs
  - normally based on HTML or JavaScript.
  - provides automatic facilities for concurrency management, scalability, failover, and security.
  - supports web development interfaces such as SOAP and REST
  - able to access databases & reuse services within a private network
- A general model is implemented under which developers build appls
  - designed to run on the provider's infrastructure
  - delivered to users in via an Internet browser
- Downfall: a lack of interoperability and portability among providers
  - if you create an appl with one cloud provider & decide to move to another, you may not be able to do so or you'll have to pay a high price
- Ex: Google App Engine, Microsoft Azure, Zoho Creator, NetSuite NetFlex, Akamai EdgePlatform, Salesforce Force.com, Facebook Platform

Infrastructure as a Service (IaaS)

- Know also as Hardware as a Service (HaaS)
- Rents resources:
  - Server space
  - Network equipment
  - Memory
  - CPU cycles
  - Storage space
- Needs:
  - Service level agreements
  - Computer hardware
  - Network
  - Internet connectivity
  - Platform virtualization environment
  - Utility computing billing



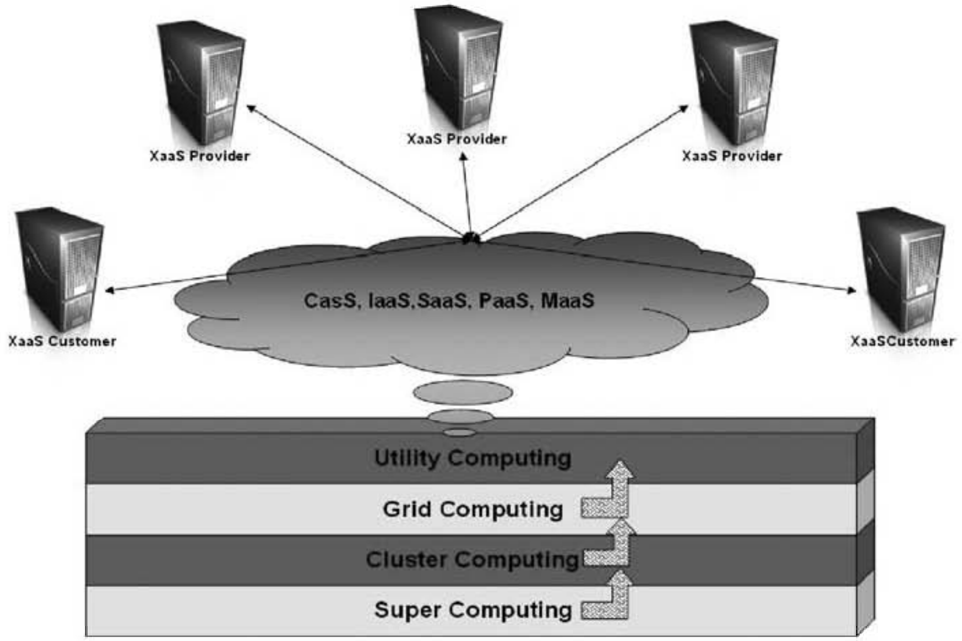
Organisation	Cloud Service
Amazon	Elastic Compute Cloud (EC2)
Amazon	Dynamo
Amazon	Simple Storage Service (S3)
Amazon	SimpleDB
Amazon	CloudFront
Amazon	SQS
AppNexus	AppNexus Cloud
Bluelock	Virtual Cloud Computing
Bluelock	Virtual Recovery
Dropbox	Dropbox Cloud Storage
Emulab	Emulab Network Testbed
ENKI	Virtual Private Data Centers
Reservoir	Open Nebula
FlexiScale	FlexiScale Cloud Computing
GoGrid	Cloud Hosting
GoGrid	Cloud Storage
Google	Google Big Table
Google	Google File System
HP	iLO
HP	Tycoon

Organisation	Cloud Service
Joyent	Accelerator
Joyent	Connector
Joyent	BingoDisk
Nirvanix	Storage Delivery Network
Openflow	OpenFlow
Rackspace	Mosso Cloud Sites
Rackspace	Mosso Cloud Storage
Rackspace	Mosso Cloud Servers
Skytap	Skytap Virtual Lab
Terremark	Infinistructure
Globus	Nimbus
todo GmbH	flexIT
UCSB	Eucalyptus
Zimory	Zimory Public Cloud Market
Zumodrive	Hybrid Cloud Storage
10gen	Mongo DB
10gen	Babble Application Server

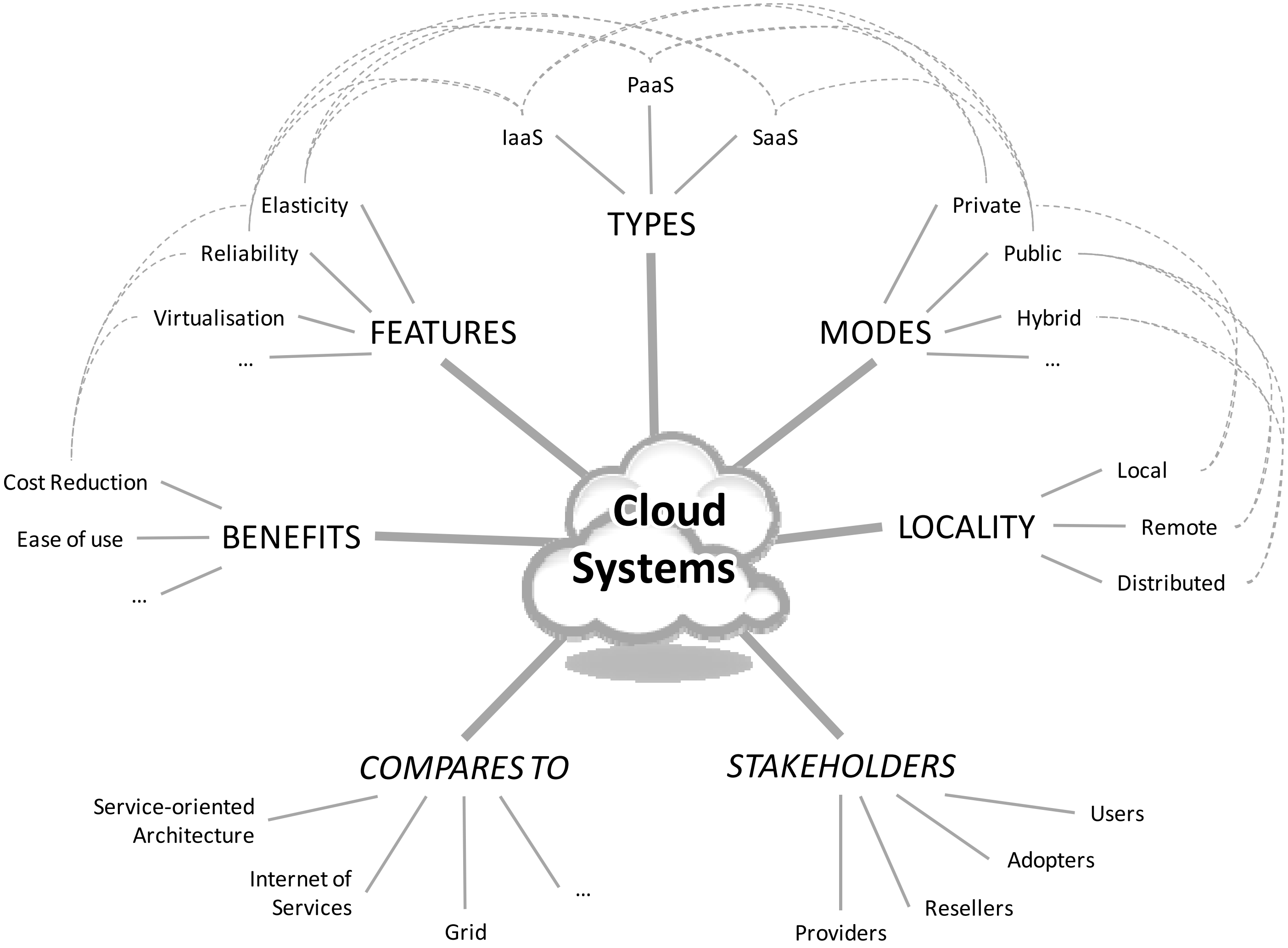
Anything as a Service (XaaS)

- Storage as a Service
- Database as a Service
- Communication as a Service
- Network as a Service
- Monitoring as a Service
- Testing as a Service
- HPC as a Service
- Human as a Service
- Process as a Service
- Information as a Service
- Identity as a Service
- Application as a Service
- Integration as a Service
- Governance as a Service
- Security as a Service
- Backup as a Service
- Business Processes as a Service

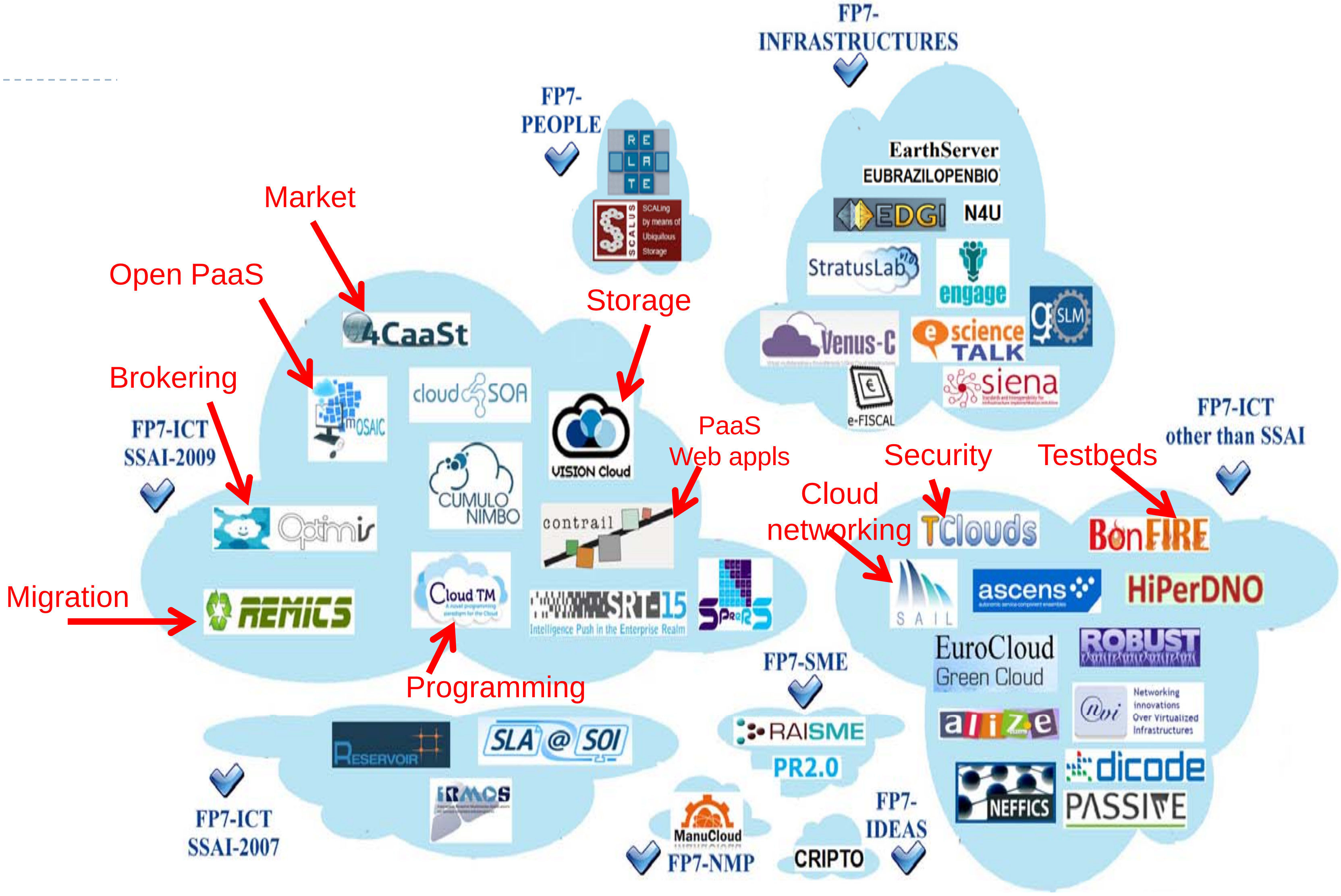
…



Taxonomies [From EC Expert Group Report]



European Cloud initiatives [EC projects]



D.Petcu, J.L. Vazquez-Poletti (eds) European Research Activities in Cloud Computing, CSP, UK, Jan 2012

How to Port an Application Between Clouds?

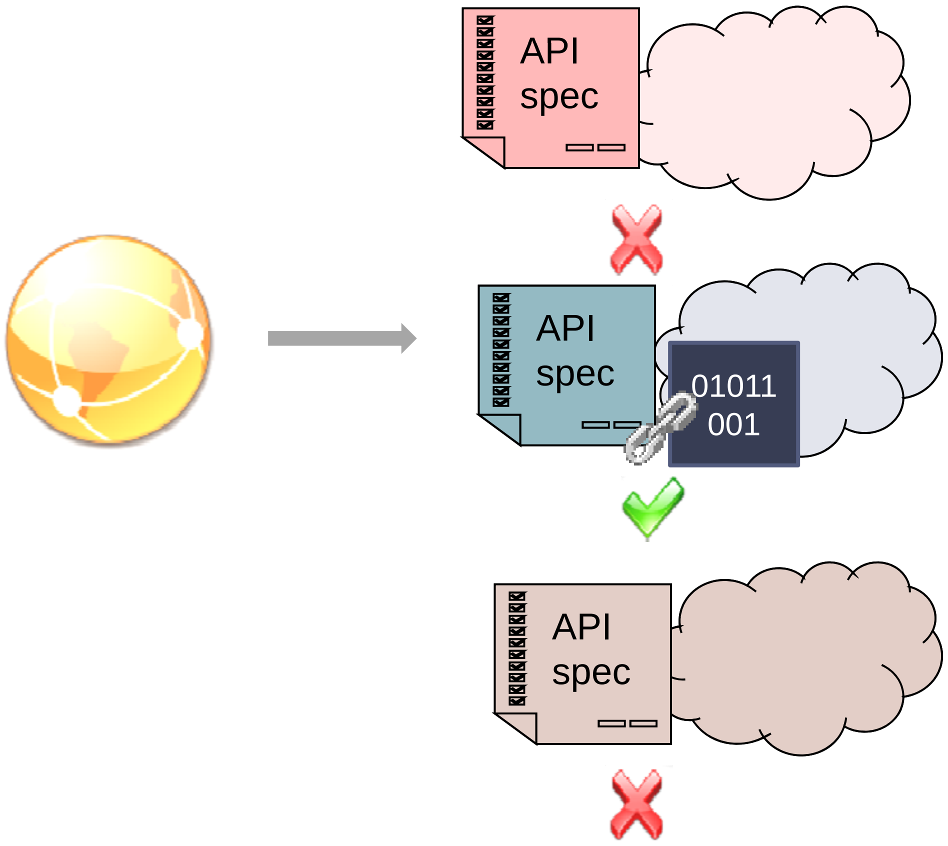
Part II: Portability and Interoperability Issues

Vendor lock-in

Content

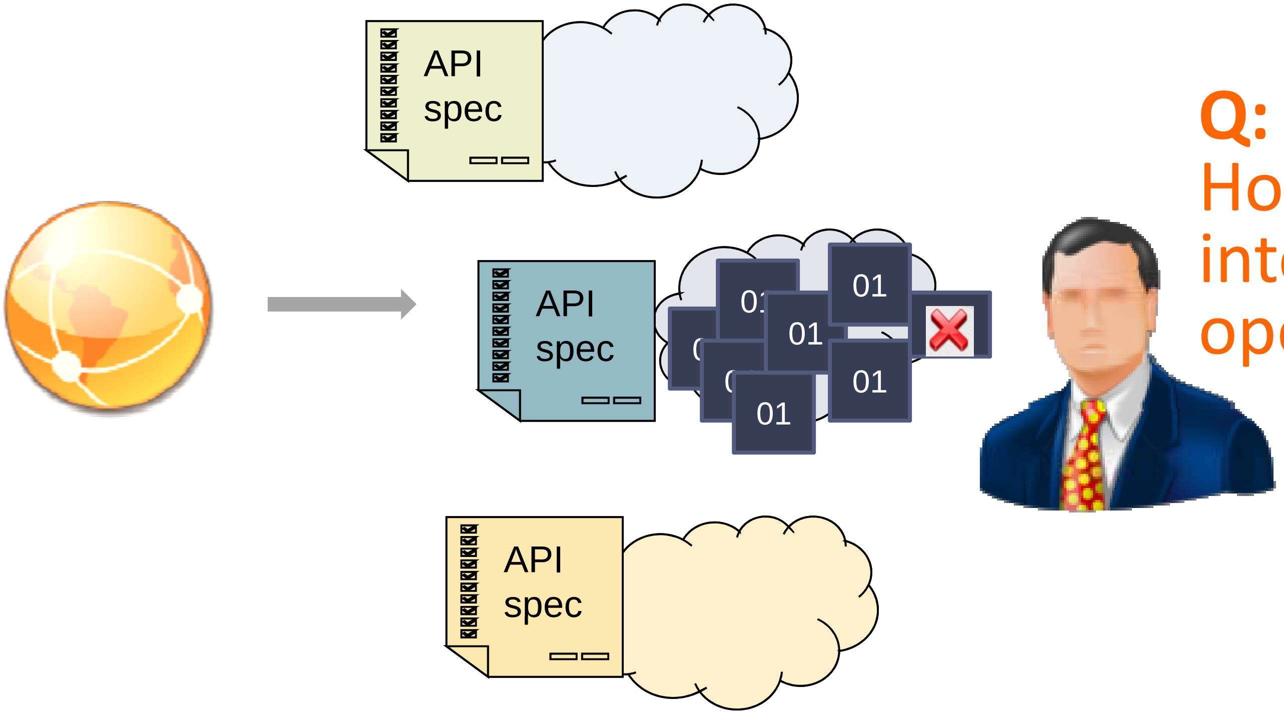
- Problem definition and taxonomy
- Approaches

Portability in Clouds?



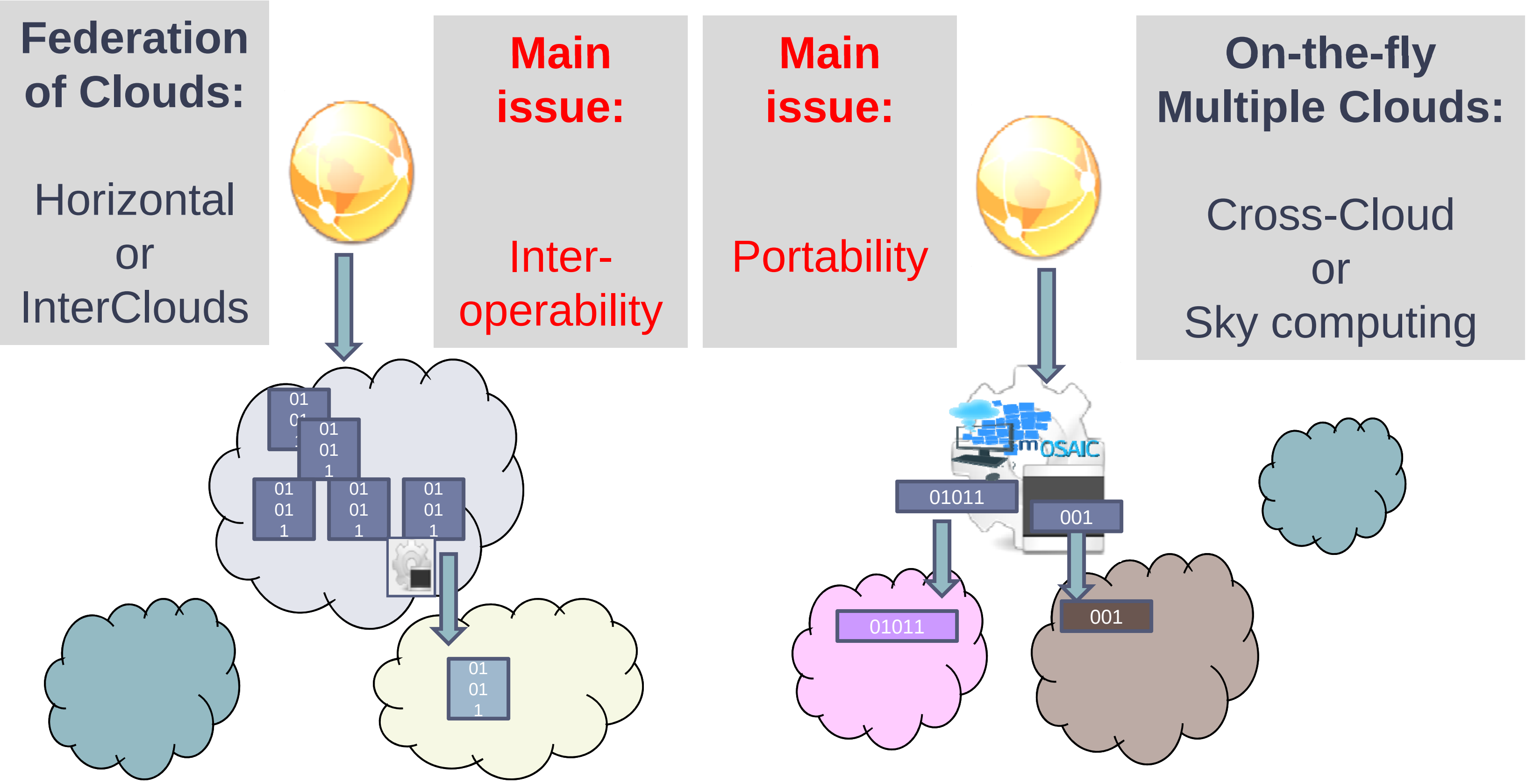
Q:
How to port the appl?

Interoperability in Clouds?



Q:
How to interoperate?

Scenarios for multiple Clouds



Federation of Clouds:

Horizontal
or
InterClouds

Main issue:

Inter-operability

Main issue:

Portability

On-the-fly Multiple Clouds:

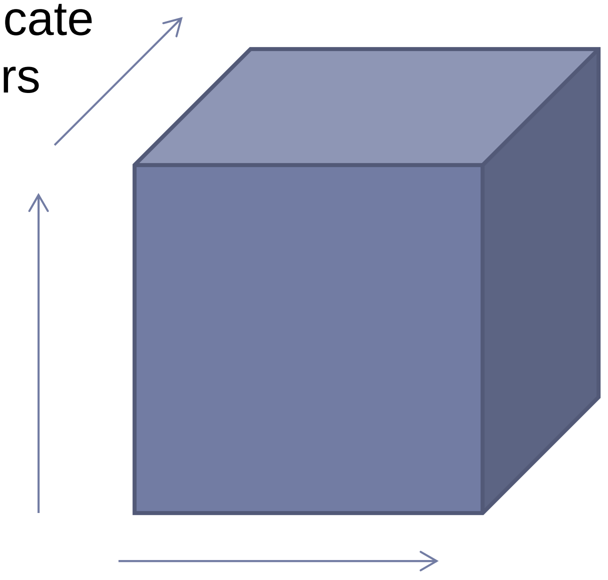
Cross-Cloud
or
Sky computing

Use cases of multiple Clouds

- **NIST CCSRWG (CC standard, 2011) classification**
  - Serially (one Cloud after another)
    - Migration between Clouds
    - Interface across multiple Clouds
    - Work with a selected Cloud
    - Change Cloud vendors
  - Simultaneously (several Clouds at a time)
- **CC Use Case Discussion Group**
  - Changing Cloud vendor
  - Hybrid Cloud (Distributed deployment?)

Interoperability definition & dimensions

- **Dictionary:**
  - Property referring to the ability of diverse systems to work together
- **By mottos:**
  - avoid vendor lock-in
  - develop your application once, deploy anywhere
  - enable hybrid clouds
  - one API to rule them all



POLICY:
Federate, communicate between providers

RUNTIME:
Migration support

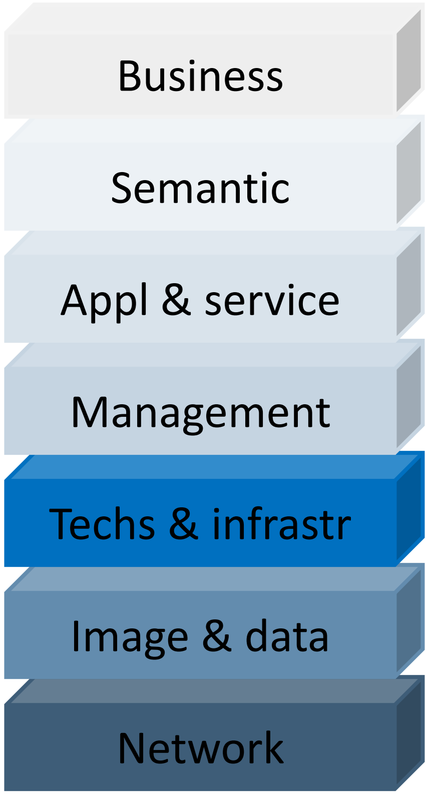
DESIGN:
Abstract the programmatic differences

Interoperability/Clouds- history

1. Migration – targets VMs
   - Create, import, share VMs (e.g. use OVF)
2. Federation – targets networking
   - Portable VMs moved between clouds and hypervisors without reconfiguring anything
3. On-demand (burst) – targets APIs
   - Migration and federation on demand
   - Interoperability focused on storage and compute (e.g. CDMI, OCCI)

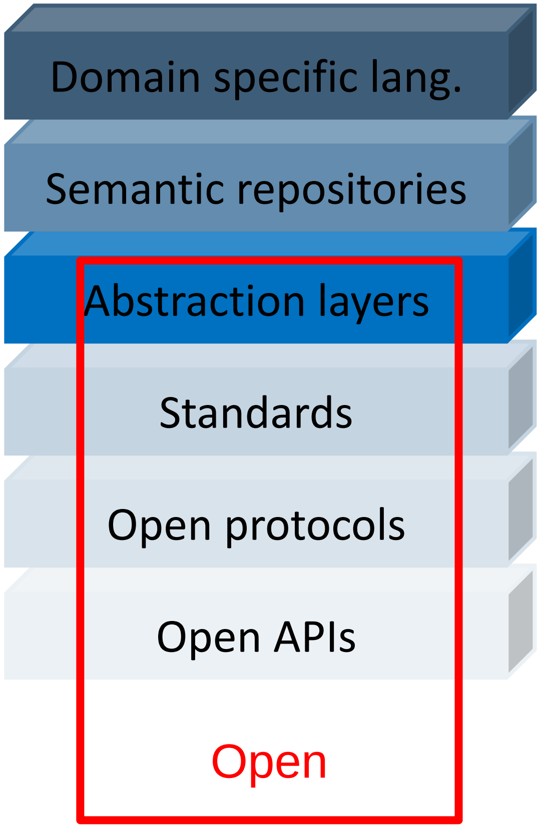
Current solutions

Levels



	E.g.
Business	Strategies, regulations, mode of use
Semantic	Function calls and responses
Appl & service	Automation, configuration
Management	Standards in deployment & migration
Techs & infrastr	Protocols for requests/responses
Image & data	Pre-deployment, work-loads
Network	Allocation, admission

Techs

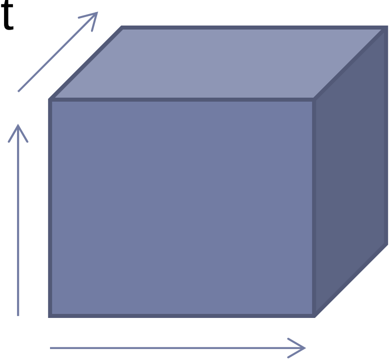


	E.g.
Domain specific lang.	Automated translation in code
Semantic repositories	UCI
Abstraction layers	Mediators, frame-works (SLA@SOI)
Standards	OVF/DMTF, CDMI/SNIA
Open protocols	OCCI, Deltacloud
Open APIs	jClouds, libcloud, OpenStack

Open

Portability between Clouds

- Ability to use components or systems lying on multiple hardware or software environments
- Dimensions:

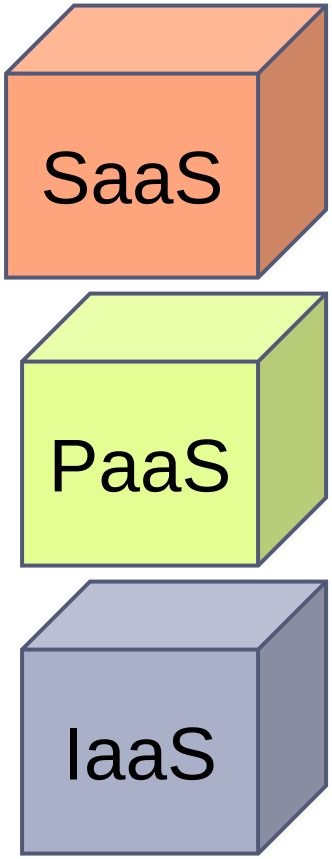


DATA:
Import & export
functionality

SERVICE:
On the fly add, reconfig
and remove resources

FUNCTION:
Define appl. functionality
in platform-agnostic manner

Portability at XaaS level



Preserve/enhance functionality when substitute softw

Measures:

- open source; proprietary/open formats;
- integration techs; appl server/OS

Minim.appl.rewriting while preserve/ enhance control

Measures:

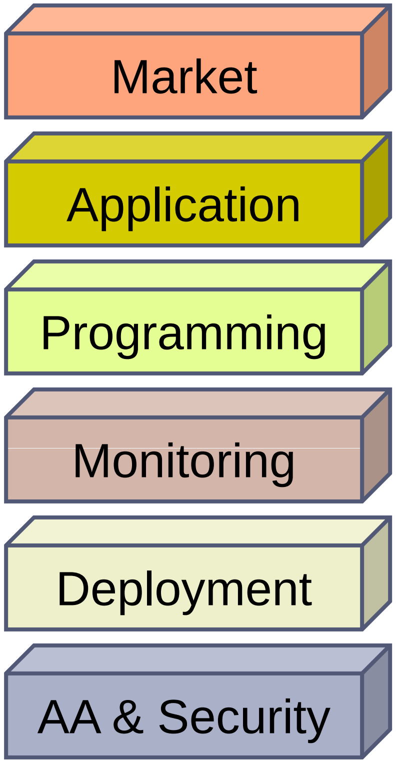
- proprietary vs.open APIs, progr.languages,data formats
- tight vs. loose coupled services
- abstract layers for queuing & messaging

Appls and data migrate and run at a new provider

Measures:

- ability to port VMs and data
- underlying configurations across providers

Requirements for portability



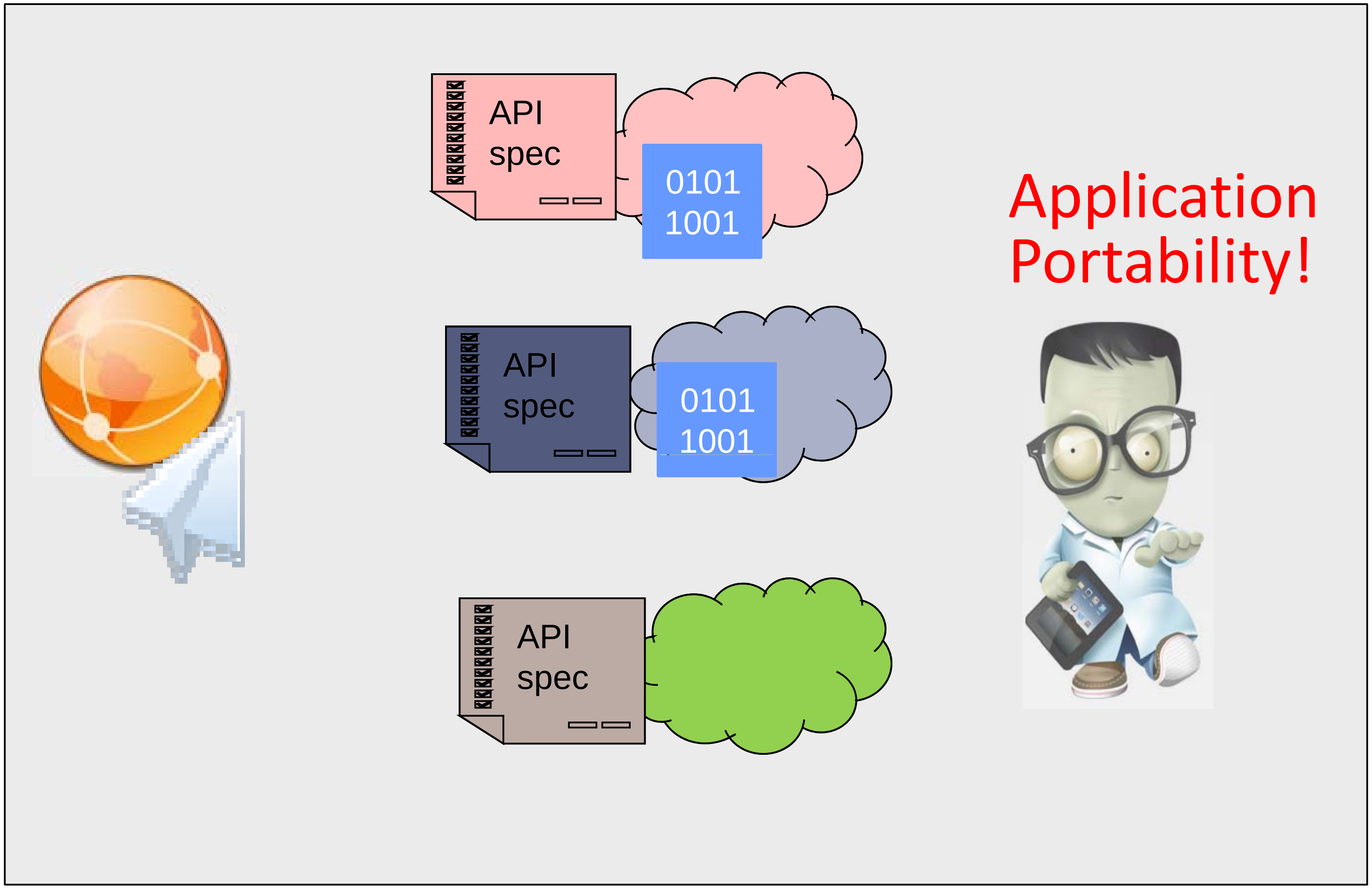
Market	Economic models, cost-effectiveness, license flexibility, negotiated SLAs, leasing mechanisms
Application	Data portability and exchange, scale-out, location-free, workflow management
Programming	Minimal reimplementation when move, standard APIs, same tools for cloud-based and entreprise-based appls
Monitoring	SLA and performance monitoring, QoS aware services, service audit, sets of benchmarks
Deployment	Deploy in multiple clouds with single management tool, navigation between services, automated provisioning, resource discovery and reservation, behavior prediction
AA & Security	Single sign-on, digital identities, security Standards, trust mechanisms, authentication

How to Port an Application Between Clouds?

Part III: mOSAIC Generalities

Open-source API and PaaS for multiple Clouds

mOSAIC’s marketing motto:
“Flying through the Clouds”



mOSAIC as R&D collaboration effort

Norwegian Sea
ICELAND
NORWAY
SWEDEN
FINLAND
RUSSIA
North Atlantic Ocean
UNITED KINGDOM
IRELAND
BELARUS
POLAND
GERMANY
UKRAINE
FRANCE
ROMANIA
BULGARIA
ITALY
SPAIN
PORTUGAL
GREECE
TURKEY
Mediterranean Sea
ALGERIA
TUNISIA
MOROCCO

European Commission

www.mosaic-cloud.eu

Consortium:

1. Second University of Naples, Italy
2. Institute e-Austria Timisoara, Romania
3. European Space Agency, France
4. Terradue SRL, Italy
5. AITIA International Informatics, Hungary
6. Tecnalia, Spain
7. Xlab, Slovenia
8. University of Ljubljana, Slovenia
9. Brno University of Technology, Czech Republic

September 2011: 1st API implementat. (Java)

September 2012: 1st stable PaaS,
2nd API impl. (Python)

March 2013: Full software package

Open-source Platform Software

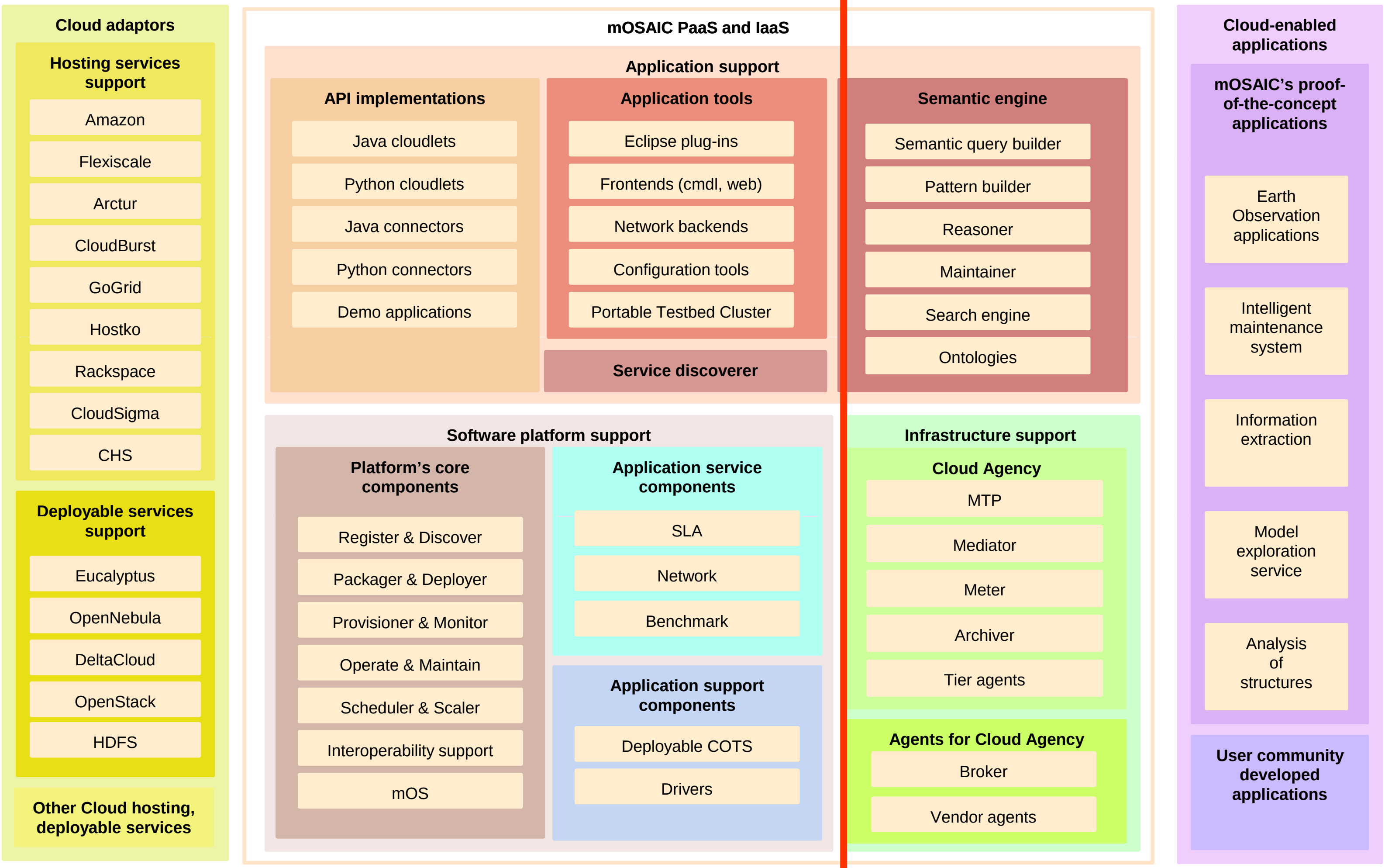
Product	AppScale	Cloud Foundry	ConPaaS	mOSAIC	OpenShift	TyphoonAE	WaveMaker
Owner	Univ. California	VMWare	Contrail Consortia	mOSAIC Consortia	RedHat	Tobias Rodäbel	VMWare
Site	appscale.cs. ucsb.edu	www. cloud foundry.com	www. conpaas. eu	www. mosaic-cloud.eu	open shift.com	code. google. com/p/typhoonae	www.wave maker.com
Repository	appscale. googlecode. com/svn/	github. com/ cloud foundry	www. conpaas. eu/ download/	bitbucket. org/ mosaic	github. com/ openshift	code. google. com/p/ typho-onae/ downloads/	dev.wavemaker. com/wiki/ bin/
State	1.5/Jul 2011	0.x , Beta	0.1/Sep 2011	0.5/Jun’12, Beta	Production	0.2/Dec 2010/beta	6.4.4/Dec 2011
Languages	Python, Java, Go	Java, Ruby,Node.js, Groovy	PHP	Java, Python	Java, Python, Perl, PHP, Ruby	Python	Java
Data Support	HBase, Redis Hypertable, MySQL Cluster, Cassandra, Voldermort, MongoDB, MemcacheDB	MongoDB, SQLFire, PotsgreSQL, Redis	Scalaris, MySQL, XtreemFS	Riak, MemcacheDB, Redis, MySQL, HDFS	MySQL, MongoDB, Amazon RDS	MongoDB, MySQL, Berkeley DB JE	Amazon S3, Rackspace
OS	Ubuntu, CentOS on Xen, KVM	VMWare image	XtreemOS image	CentOS, RedHat, Ubuntu, Suse	Red Hat Virtualization	Debian, Ubuntu	VMWare image
Messaging	Channel	RabbitMQ	Own design	RabbitMQ	Own design	RabbitMQ, ejabberd, Channel	Own design
Clouds tested	Amazon EC2, Eucalyptus	VMWare	Own testbed	Amazon EC2, Eucalyptus, OpenNebula, Flexiscale	RightScale Rackspace, Smart-Cloud, Amazon	Google	EC2, Rackspace, OpSource, Eucalyptus
Interface	CLI, Web	CLI	Web	CLI,Web,REST	CLI,REST	CLI	Studio

Open-source Platform Software

Product	CloudFoundry	mOSAIC	OpenShift
Development support	**1**	**2**	**3**
Dedicated to web aps or general	Web apps	General	Web apps
Desktop Cloud Simulator	Yes	Yes	No
API access	No	Yes	No
Support standard programming libs	Yes	Yes	Yes
Impact on web application architecture	No	Yes	No
Complexity of porting web application	Medium	Low	Low
Standard support tools	Spring Tools	No	JBoss, Zend
Thread access	Yes	No	Yes
MySQL	Yes	Yes	Yes
Allows to choose stack components	Yes	Yes	No
Allow to pull data out	Yes	Yes	Yes
Debugging mode	Yes	Yes	Yes
Deployment support	**1**	**2**	**3**
Lock-in when building own Cloud	Yes (VMWare)	No	Yes (RHE)
Web server (e.g. Tomcat)	Yes	Yes	Yes
Build-in-balancer	No	Yes	Yes
Auto-scaling app server	No	Yes	Yes
Auto-scaling database	No	Yes	No
Performance analytics	Yes	No	Yes
Support multiple Cloud providers	Yes	Yes	Yes
Agreements SLA	No	Yes	No
Deploy with a special tool	Yes	No	No
Support Private Cloud	Yes	Yes	No
Allows to add third party components	Yes	Yes	Yes
Execution support	**1**	**2**	**3**
Command line (CLI)	Yes	Yes	Yes
Web console	No	Yes	Yes
Access to logs via web	No	Yes	Yes
Web based monitoring	No	Yes	Yes
Multitenant	Yes	Yes	Yes

Layered architecture

Open-source and deployable PaaS



Cloud adaptors

- **Hosting services support**
  - Amazon
  - Flexiscale
  - Arctur
  - CloudBurst
  - GoGrid
  - Hostko
  - Rackspace
  - CloudSigma
  - CHS
- **Deployable services support**
  - Eucalyptus
  - OpenNebula
  - DeltaCloud
  - OpenStack
  - HDFS
- **Other Cloud hosting, deployable services**

mOSAIC PaaS and IaaS

- **Application support**
  - **API implementations**
    - Java cloudlets
    - Python cloudlets
    - Java connectors
    - Python connectors
    - Demo applications
  - **Application tools**
    - Eclipse plug-ins
    - Frontends (cmdl, web)
    - Network backends
    - Configuration tools
    - Portable Testbed Cluster
  - **Service discoverer**
  - **Semantic engine**
    - Semantic query builder
    - Pattern builder
    - Reasoner
    - Maintainer
    - Search engine
    - Ontologies
- **Software platform support**
  - **Platform's core components**
    - Register & Discover
    - Packager & Deployer
    - Provisioner & Monitor
    - Operate & Maintain
    - Scheduler & Scaler
    - Interoperability support
    - mOS
  - **Application service components**
    - SLA
    - Network
    - Benchmark
  - **Application support components**
    - Deployable COTS
    - Drivers
- **Infrastructure support**
  - **Cloud Agency**
    - MTP
    - Mediator
    - Meter
    - Archiver
    - Tier agents
  - **Agents for Cloud Agency**
    - Broker
    - Vendor agents

Cloud-enabled applications

- **mOSAIC's proof-of-the-concept applications**
  - Earth Observation applications
  - Intelligent maintenance system
  - Information extraction
  - Model exploration service
  - Analysis of structures
- **User community developed applications**

OS repository: https://bitbucket.org/mosaic

Restrictions

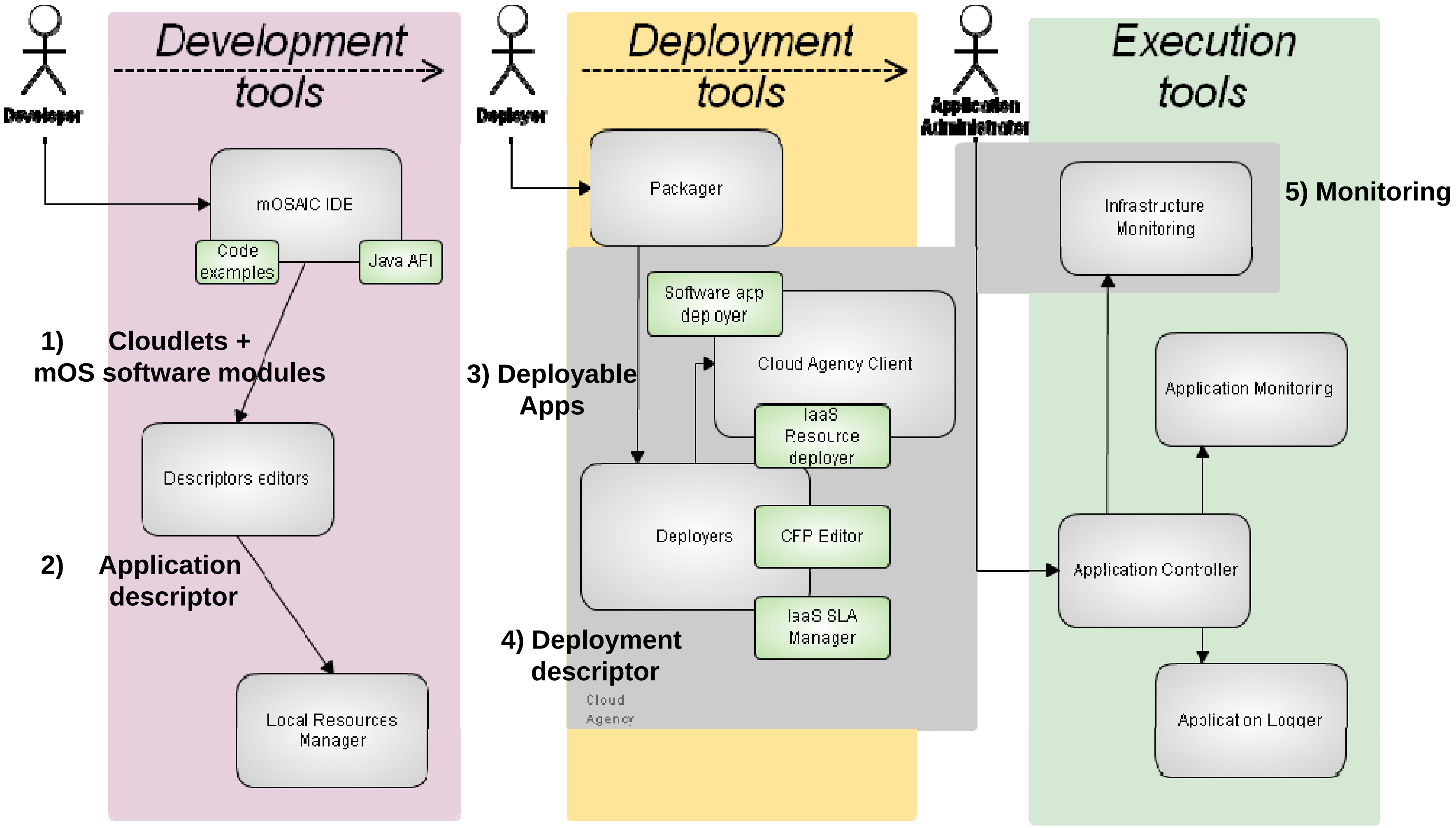
Guide-lines:

1. Split in components
2. Establish dependencies
3. Use specific communication patterns
   - RPC, message queues
   - All exchanges (including exterior) through API
   - Avoid sockets

Steps:

1. Develop components
   - Specify resources reqs
2. Submit reqs to resource broker/provisioner
   - Bootstrap the resources
   - Deploy and start appl
3. Monitor the appl

Application lifecycle



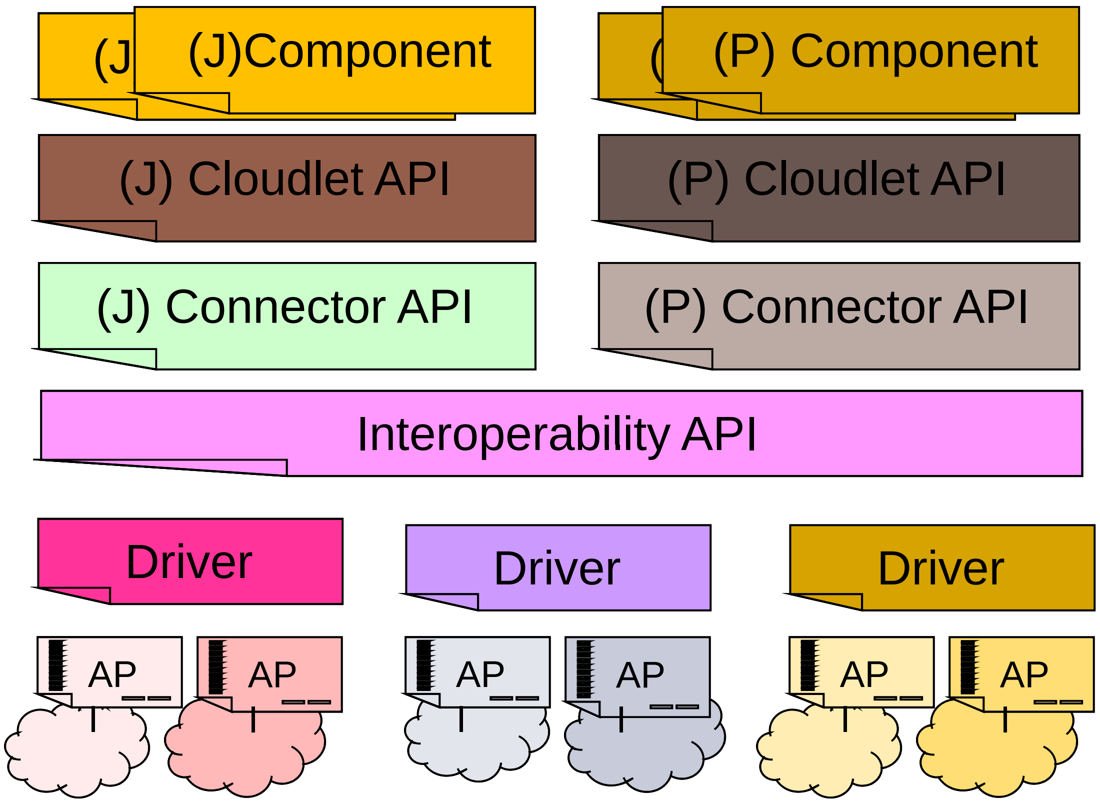
Basic concepts of API's

- **Cloud Building Block (CBB):**
  - basic component of an application
  - can be a resource (CR) or a configurable component (CC)
- **Cloud Resource (CR):**
  - Controlled by Cloud provider (e.g. key-value store, message queue syst)
  - Can be a hosted service (via adaptor) or a software service (deployable)
- **Cloud Component (CC):**
  - Controlled by application developer
  - CC instances consume CR
  - communication between CC via CR like message queues (to control redirection in case of faults or scale-up/scale-down)

CC properties

- **Elastic**
  - scale up and down no.of instances of the same CC
- **Manageable**
  - Possible to configure it and change the parameters
- **Isolated**
  - CC instances independent from other CC
- **Fault tolerant**
  - Automated using the Container (instance manager)
- **Implemented by**
  - a Container + several Cloudlets instances

Layers of mOSAIC’ set of APIs



Classical components of applications

Component reacting to events

Operations with standard type of resources

Proxies generator

API for same type of resource

Cloudlet and Connector

- **Cloudlet:**
  - Behavior: event-driven, stateless
  - Automated elasticity: no. of Cloudlet instances controlled by Container
  - Programmable elasticity: no. of containers
  - Functionality do not depend on no. instances
- **Connector:**
  - Behavior: RPC
  - Interface defining the set of events

    to which the Cloudlet should react
  - Abstract the access to Cloud resources

Interoperability API and Drivers

- **Interoperability API**
  - Ensure language independence
  - protocol syntax and semantic enforcements.
  - RPC solution that abstracts addressing
  - stubs to Driver API and proxies to Connector.
- **Driver API**
  - wraps the native API
  - all resources of the same type are exposed with the same interface
  - eg. HBase vs. Riak key-value store: a matter of configuration.

How to Port an Application Between Clouds?

Part IV: Demos

mOSAIC's examples

How to use it?

- Write component-based application
  - Languages: Java or Python
  - Communications through message passing
  - Respect the event-driven style of programming
  - Find the proper functionalities with the Semantic Engine
- Debug your application on the desktop or on-premise server(s)
  - Within Eclipse
  - Use Personal Testbed Cluster using VirtualBox for the VMs
- Deploy your application in a Cloud
  - Assisted by Cloud Agency and Broker (with SLAs)
- Monitor & modify the applications
  - Control the life-cycle of the components (start/stop/replace)

Need help?

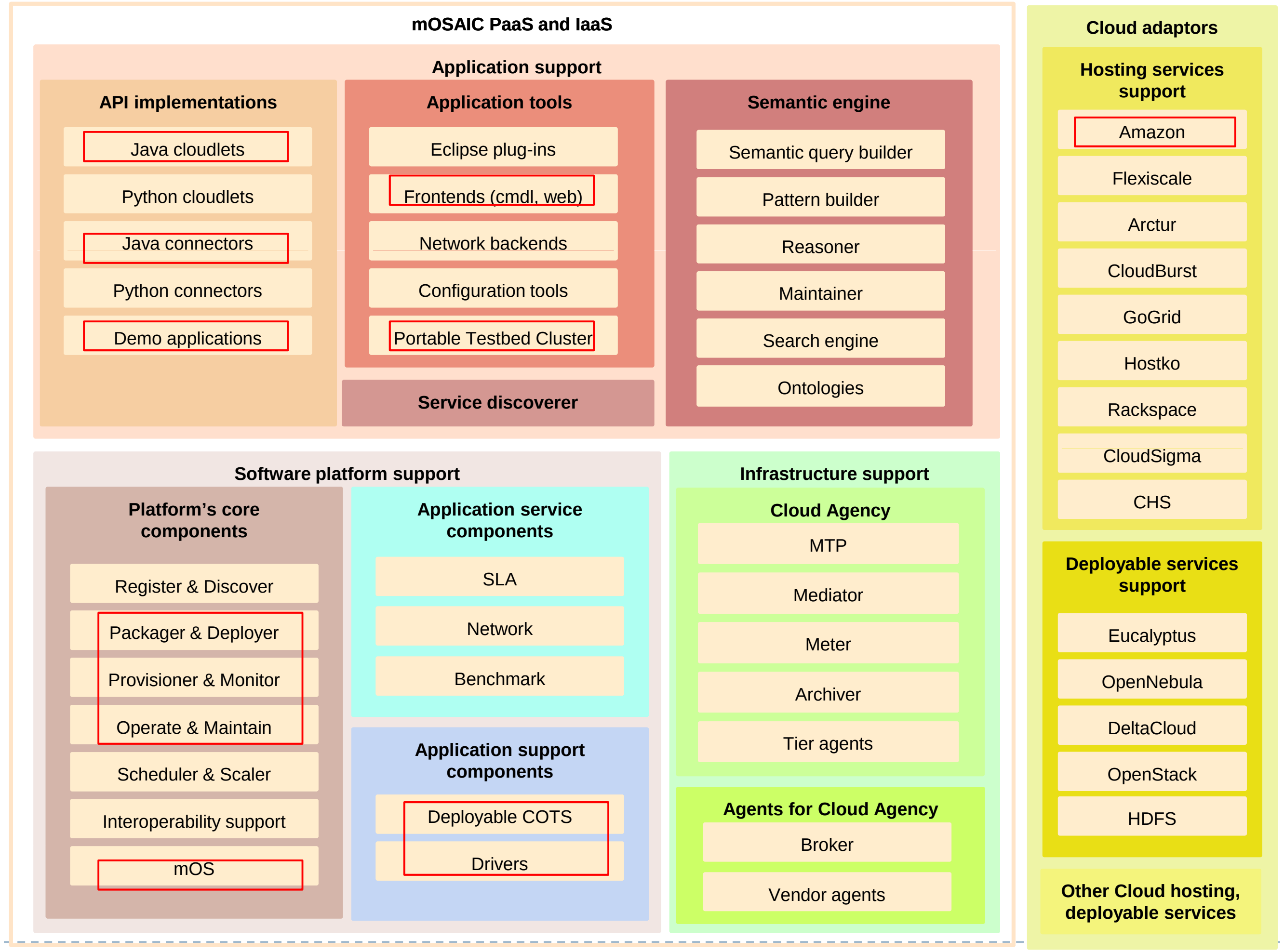
Follow documentation from http://developers.mosaic-cloud.eu

and YouTube demos (search “mOSAIC Cloud computing”)

From application development to the execution in a Cloud

Components used
in the demo

□



mOSAIC PaaS and IaaS

Application support

API implementations
- Java cloudlets
- Python cloudlets
- Java connectors
- Python connectors
- Demo applications

Application tools
- Eclipse plug-ins
- Frontends (cmdl, web)
- Network backends
- Configuration tools
- Portable Testbed Cluster

Service discoverer

Semantic engine
- Semantic query builder
- Pattern builder
- Reasoner
- Maintainer
- Search engine
- Ontologies

Software platform support

Platform's core components
- Register & Discover
- Packager & Deployer
- Provisioner & Monitor
- Operate & Maintain
- Scheduler & Scaler
- Interoperability support
- mOS

Application service components
- SLA
- Network
- Benchmark

Application support components
- Deployable COTS
- Drivers

Infrastructure support

Cloud Agency
- MTP
- Mediator
- Meter
- Archiver
- Tier agents

Agents for Cloud Agency
- Broker
- Vendor agents

Cloud adaptors

Hosting services support
- Amazon
- Flexiscale
- Arctur
- CloudBurst
- GoGrid
- Hostko
- Rackspace
- CloudSigma
- CHS

Deployable services support
- Eucalyptus
- OpenNebula
- DeltaCloud
- OpenStack
- HDFS

Other Cloud hosting, deployable services

Two examples

1. **Hello!**
   - API in Java
   - mOS in Amazon EC2
   - A Cloudlet running on Amazon EC2
   - Components storage in Amazon S3
   - Manually launch of a component
   - Not a web application

2. **Twitter watcher**
   - Personal Testbed Cluster
   - Application descriptor and deployer
   - mOSAIC public repository of components
   - Automated launch from PTC of the application: packager and deployer
   - Same application running locally on PTC (debug) and on Amazon (final)

Videos / YouTube

1. ***Application development***
   - How we start the PTC and how we use locally the platform: http://youtu.be/5GTolXs9gm0
   - Write a "hello-cloudlet" and debugging it on local computer: http://youtu.be/1xrtN7kPAp4

2. ***Application deployment***
   - How we make a package from “hello-cloudlet”, how we upload it in a public repository (in this case on Amazon S3), and how we execute it: http://youtu.be/HX7eL4DhIRo
   - How we start manually an application components (user cloudlets, COTS and drivers) on EC2 : http://youtu.be/VlHuE-D9i_Q
   - How we start the application from PTC using a deployment descriptor: http://youtu.be/BGzw7StHeVU
   - With voice: http://youtu.be/ctO9fqaDMBc