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
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, apps, services) that can be rapidly provisioned and released with minimal management effort or service-provider interaction.
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.
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

<table>
<thead>
<tr>
<th>Technical</th>
<th>Business /Economic</th>
<th>Social / Legal</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elasticity / Scalability</td>
<td>Outsourcing</td>
<td>Security</td>
<td>Multi-Tenancy</td>
</tr>
<tr>
<td>Virtualisation</td>
<td>Pay per use</td>
<td>Provenance</td>
<td>Ease of Use</td>
</tr>
<tr>
<td>Agility &amp; Adaptability</td>
<td>Resource utilisation</td>
<td>Privacy</td>
<td></td>
</tr>
<tr>
<td>Availability</td>
<td>Energy efficiency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Management</td>
<td>Metering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmability</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

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:

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; Microsoft 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 apps and services completely from the Internet, without having to download or install software

- Services include:
  - Appliance 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 app to work with
  - Offer APIs that enable developers to exploit functionality over the Internet, rather than delivering full-blown apps
  - 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 apps
  - 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 app 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

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

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Cloud Service</th>
<th>Organisation</th>
<th>Cloud Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>Elastic Compute Cloud (EC2)</td>
<td>Joyent</td>
<td>Accelerator</td>
</tr>
<tr>
<td>Amazon</td>
<td>Dynamo</td>
<td>Joyent</td>
<td>Connector</td>
</tr>
<tr>
<td>Amazon</td>
<td>Simple Storage Service (S3)</td>
<td>Joyent</td>
<td>BingoDisk</td>
</tr>
<tr>
<td>Amazon</td>
<td>SimpleDB</td>
<td>Nirvanix</td>
<td>Storage Delivery Network</td>
</tr>
<tr>
<td>Amazon</td>
<td>CloudFront</td>
<td>Openflow</td>
<td>OpenFlow</td>
</tr>
<tr>
<td>Amazon</td>
<td>SQS</td>
<td>Rackspace</td>
<td>Mosso Cloud Sites</td>
</tr>
<tr>
<td>AppNexus</td>
<td>AppNexus Cloud</td>
<td>Rackspace</td>
<td>Mosso Cloud Storage</td>
</tr>
<tr>
<td>Bluelock</td>
<td>Virtual Cloud Computing</td>
<td>Rackspace</td>
<td>Mosso Cloud Servers</td>
</tr>
<tr>
<td>Bluelock</td>
<td>Virtual Recovery</td>
<td>Skytap</td>
<td>Skytap Virtual Lab</td>
</tr>
<tr>
<td>Dropbox</td>
<td>Dropbox Cloud Storage</td>
<td>Terremark</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>Emulab</td>
<td>Emulab Network Testbed</td>
<td>Globus</td>
<td>Nimbus</td>
</tr>
<tr>
<td>ENKI</td>
<td>Virtual Private Data Centers</td>
<td>todo GmbH</td>
<td>flexIT</td>
</tr>
<tr>
<td>Reservoir</td>
<td>Open Nebula</td>
<td>UCSB</td>
<td>Eucalyptus</td>
</tr>
<tr>
<td>FlexiScale</td>
<td>FlexiScale Cloud Computing</td>
<td>Zimory</td>
<td>Zimory Public Cloud Market</td>
</tr>
<tr>
<td>GoGrid</td>
<td>Cloud Hosting</td>
<td>Zumodrive</td>
<td>Hybrid Cloud Storage</td>
</tr>
<tr>
<td>GoGrid</td>
<td>Cloud Storage</td>
<td>10gen</td>
<td>Mongo DB</td>
</tr>
<tr>
<td>Google</td>
<td>Google Big Table</td>
<td>10gen</td>
<td>Babble Application Server</td>
</tr>
<tr>
<td>Google</td>
<td>Google File System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP</td>
<td>iLO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP</td>
<td>Tycoon</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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]

Cloud Systems

TYPES
- IaaS
- PaaS
- SaaS

MODES
- Public
- Private
- Hybrid

LOCALITY
- Local
- Remote
- Distributed

FEATURES
- Reliability
- Elasticity
- Virtualisation

BENEFITS
- Cost Reduction
- Ease of use

COMPARES TO
- Service-oriented Architecture
- Internet of Services
- Grid

STAKEHOLDERS
- Users
- Adopters
- Providers
- Resellers

COMPAS TO STAKEHOLDERS
- Internet
- Service-oriented
- Architecture
- Grid

ADOPTERS
- Resellers
- Providers
- Users

INTERNET OF SERVICES
- Grid

TECHNOLOGIES
- Service-oriented
- Architecture
- Internet
- Grid

7/22/2012
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 app?
Interoperability in Clouds?

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

ComputationWorld, Nice 7/22/2012
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

- Business
- Semantic
- Appl & service
- Management
- Techs & infrastr
- Image & data
- Network

E.g.

- Strategies, regulations, mode of use
- Function calls and responses
- Automation, configuration
- Standards in deployment & migration
- Protocols for requests/responses
- Pre-deployment, work-loads
- Allocation, admission

Techs

- Domain specific lang.
- Semantic repositories
- Abstraction layers
- Standards
- Open protocols
- Open APIs
- Open

E.g.

- Automated translation in code
- UCI
- Mediators, frameworks (SLA@SOI)
- OVF/DMTF, CDMI/SNIA
- OCCI, Deltacloud
- jClouds, libcloud, OpenStack

ComputationWorld, Nice 7/22/2012
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

Minimappl.rewriting while preserve/ enhance control

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

Apps 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 enterprise-based apps
- **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

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

<table>
<thead>
<tr>
<th>Product</th>
<th>AppScale</th>
<th>Cloud Foundry</th>
<th>ConPaaS</th>
<th>mOSAIC</th>
<th>OpenShift</th>
<th>TyphoonAE</th>
<th>WaveMaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Univ. Ca.ifornia</td>
<td>VMWare</td>
<td>Contrail Consortia</td>
<td>mOSAIC Consortia</td>
<td>RedHat</td>
<td>Tobias Rodäbel</td>
<td>VMWare</td>
</tr>
<tr>
<td>State</td>
<td>1.5/Jul 2011</td>
<td>0.x, Beta</td>
<td>0.1/Sep 2011</td>
<td>0.5/Jun’12, Beta</td>
<td>Production</td>
<td>0.2/Dec 2010/beta</td>
<td>6.4.4/Dec 2011</td>
</tr>
<tr>
<td>Languages</td>
<td>Python, Java, Go</td>
<td>Java, Ruby, Node.js, Groovy</td>
<td>PHP</td>
<td>Java, Python</td>
<td>Java, Python, Perl, PHP, Ruby</td>
<td>Python</td>
<td>Java</td>
</tr>
<tr>
<td>Data Support</td>
<td>HBase, Redis, Hypertable, MySQL Cluster, Cassandra, Voldermort, MongoDB, MemcacheDB</td>
<td>MongoDB, SQLFire, PotsgreSQL, Redis</td>
<td>Scalaris, MySQL, XtreemFS</td>
<td>Riak, MemcachedDB, Redis, MySQL, HDFS</td>
<td>MySQL, MongoDB, Amazon RDS</td>
<td>MongoDB, MySQL, Berkeley DB JE</td>
<td>Amazon S3, Rackspace</td>
</tr>
<tr>
<td>OS</td>
<td>Ubuntu, CentOS on Xen, KVM</td>
<td>VMWare image</td>
<td>XtreemOS image</td>
<td>CentOS, RedHat, Ubuntu, Suse</td>
<td>Red Hat Virtualization</td>
<td>Debian, Ubuntu</td>
<td>VMWare image</td>
</tr>
<tr>
<td>Messaging</td>
<td>Channel</td>
<td>RabbitMQ</td>
<td>Own design</td>
<td>RabbitMQ</td>
<td>Own design</td>
<td>RabbitMQ, ejabberd, Channel</td>
<td>Own design</td>
</tr>
<tr>
<td>Clouds tested</td>
<td>Amazon EC2, Eucalyptus</td>
<td>VMWare</td>
<td>Own testbed</td>
<td>Amazon EC2, Eucalyptus, OpenNebula, Flexiscale</td>
<td>RightScale Rackspace, Smart-Cloud, Amazon</td>
<td>Google</td>
<td>EC2, Rackspace, OpSource, Eucalyptus</td>
</tr>
<tr>
<td>Interface</td>
<td>CLI, Web</td>
<td>CLI</td>
<td>Web</td>
<td>CLI, Web, REST</td>
<td>CLI, REST</td>
<td>CLI</td>
<td>Studio</td>
</tr>
</tbody>
</table>
# Open-source Platform Software

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

### Open-source and deployable PaaS

<table>
<thead>
<tr>
<th>Cloud adaptors</th>
<th>mOSAIC PaaS and IaaS</th>
<th>Infrastructure support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hosting services support</td>
<td>API implementations</td>
<td>Cloud Agency</td>
</tr>
<tr>
<td>Amazon</td>
<td>Java cloudlets</td>
<td>MTP</td>
</tr>
<tr>
<td>Flexiscale</td>
<td>Python cloudlets</td>
<td>Mediator</td>
</tr>
<tr>
<td>Arctur</td>
<td>Java connectors</td>
<td>Meter</td>
</tr>
<tr>
<td>CloudBurst</td>
<td>Python connectors</td>
<td>Archiver</td>
</tr>
<tr>
<td>GoGrid</td>
<td>Demo applications</td>
<td>Tier agents</td>
</tr>
<tr>
<td>Hostko</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rackspace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CloudSigma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Deployable services support

- Eucalyptus
- OpenNebula
- DeltaCloud
- OpenStack
- HDFS
- Other Cloud hosting, deployable services

### Software platform support

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

### Application support components

- Deployable COTS
- Drivers

### Application support

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

### Application tools

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

### Service discoverer

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

### Operating System (OS) repository

[https://bitbucket.org/mosaic](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

1) Cloudlets + mOS software modules
2) Application descriptor
3) Deployable Apps
4) Deployment descriptor
5) Monitoring
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

(J) Component
(J) Cloudlet API
(J) Connector API
Interoperability API

(P) Component
(P) Cloudlet API
(P) Connector API

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](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

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

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

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

Network backends
- Java connectors

Java cloudlets
- Java cloudlets

Java connectors
- Java connectors

Python connectors
- Python connectors

Demo applications
- Demo applications

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
- OpenStack
- HDFS

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

Application service components
- SLA
- Network
- Benchmark
- Deployable COTS
- Drivers

Application support components
- MTP
- Mediator
- Meter
- Archiver
- Tier agents
- Brokers
- Vendor agents

Infrastructure support
- Cloud Agency
- Agents for Cloud Agency
- Mediator
- Meter
- Archiver
- Tier agents
- Brokers
- Vendor agents

ComputationWorld, Nice 7/22/2012
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)
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/VIHuE-D9i_Q
   - How we start the application from PTC using a deployment descriptor:  
     http://youtu.be/BGzw7StHeVU
   - With voice: http://youtu.be/ctO9fqaDMBc