

Cloud



Définition ?

“... the market seems to have come to the conclusion that cloud computing has a lot in common with obscenity--you may not be able to define it, but you'll know it when you see it”

James Urquhart – The Wisdom of Clouds

Cloud origin

- * Cloud computing, introduced by
- * Amazon (2002), suite of cloud-based services including storage, computation and even human intelligence through the [Amazon Mechanical Turk](#).
- * 2006, Amazon launched its Elastic Compute cloud (EC2)
- * was announced as "Azure" in October 2008 and was released on 1 February 2010 as Windows Azure, before being renamed to Microsoft Azure on 25 March 2014.
- Google App Engine** (often referred to as **GAE** or simply **App Engine**)

3

Cloud origin

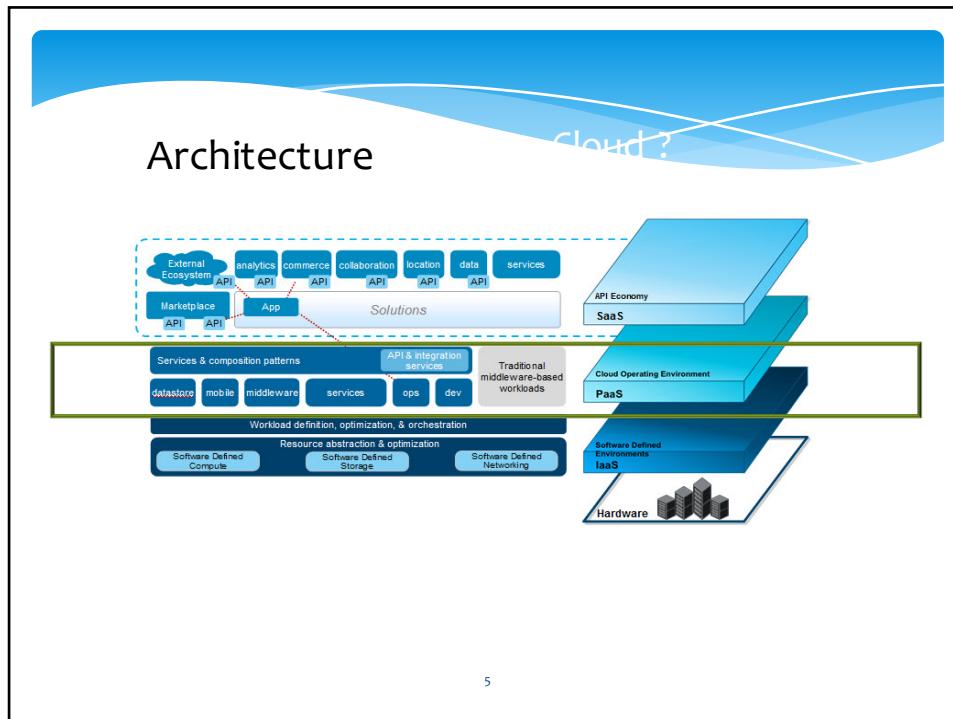
- * Now: GAE, Azure EC2, [IBM SmartCloud](#), Oracle Cloud, Heroku, etc.

- * Dockers, micro-services

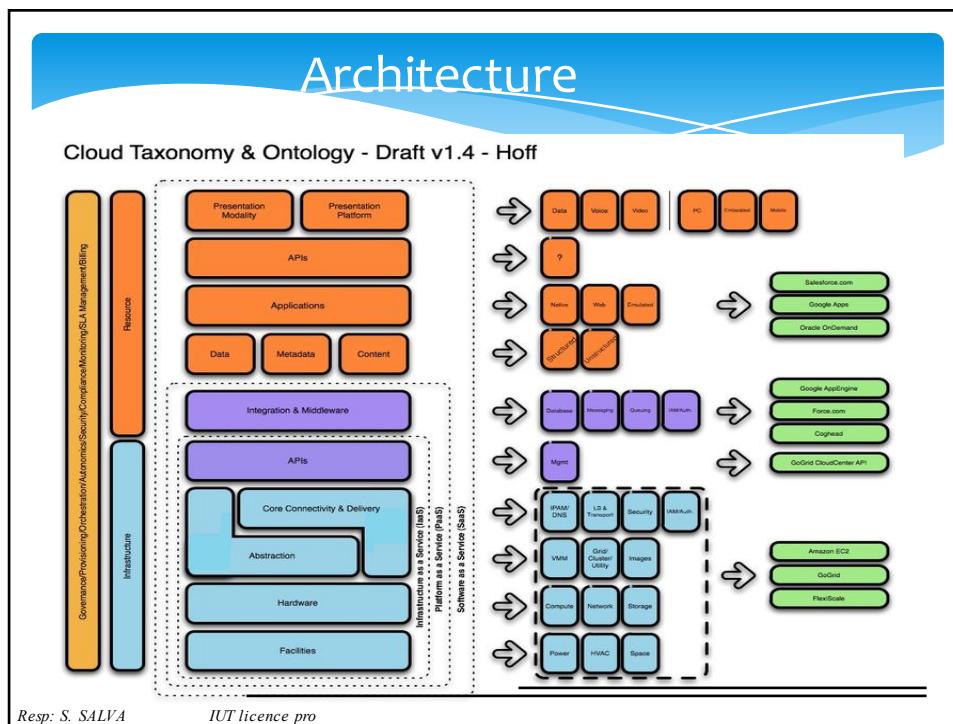
Cloud features :

- * new API,
- * storage,
- * compute,
- * Scalability (long term),
- * Elasticity (short term),
- * etc.

4



5



Architecture

The diagram illustrates the three levels of cloud computing:

- Software as a Service (SaaS):** Applications (e.g., Google Apps, salesforce.com, Pivotal Cloud Foundry)
- Platform as a Service (PaaS):** Frameworks (e.g., Google App Engine, force.com, Windows Azure)
- Infrastructure as a Service (IaaS):** Hardware (e.g., Amazon Web Services, Rackspace Cloud, Joyent)

- IaaS: Infrastructure as a service
 - Virtualisation d'OS
 - Le hardware est extensible et non géré
 - Ex: amazon
- PaaS : platform as a service
- SaaS : software as a service

Architecture

The diagram illustrates the three levels of cloud computing:

- Software as a Service (SaaS):** Applications (e.g., Google Apps, salesforce.com, Pivotal Cloud Foundry)
- Platform as a Service (PaaS):** Frameworks (e.g., Google App Engine, force.com, Windows Azure)
- Infrastructure as a Service (IaaS):** Hardware (e.g., Amazon Web Services, Rackspace Cloud, Joyent)

- PaaS : platform as a service
 - Déploiement d'application dans env. extensible
 - OS+serveur d'application (glassfish, jboss, etc.) + couche persistance + API
 - Ex: GAE, Windows Azure, openshift,etc.
- SaaS : software as a service
 - Service proposés aux clients

Resp: S. SALVA IUT licence pro

Types d'architectures

- Cloud public: solutions de stockage et applications offertes au public par accès via Internet (Amazon, Microsoft, Google)
- Cloud communautaire: infrastructure partagée entre organisation. Gestion du Cloud en interne ou par tierce partie. Travail collaboratif
- Cloud hybride: composé de >1 clouds privés, communautaires ou privés. Offre l'avantage de promouvoir plusieurs modèles de déploiements. Infrastructure interne+ externe => utilisation immédiate et locale et non dépendance à Internet. Evolutif en terme de taille via l'architecture externe,
- Cloud privé: infrastructure privée uniquement à une seule organisation. Nécessité de gérer la partie infrastructure: virtualiser environnement Business, réévaluer les ressources existantes, les problèmes de sécurité à chaque modification. Perte des avantages liés au Clouds; flexibilité, évolutivité

Resp: S. SALVA

IUT licence pro

Aperçu de Windows Azure

Mais vraiment un petit aperçu



Fonctionnalités

| | | | | | |
|--------------------------------|--------------------------------|-------------|-----------------|--------|---------|
| Modèles d'exécution | Virtual Machines | Web Sites | Cloud Services | | |
| Gestion des données | SQL Database | Tables | Blobs | | |
| Réseau | Virtual Network | Connect | Traffic Manager | | |
| Analyses business / big data | SQL Reporting | Hadoop | | | |
| Messaging | Queues | Service Bus | | | |
| Cache | Caching | CDN | | | |
| Identité | Windows Azure Active Directory | | | | |
| Haute performance | HPC Scheduler | | | | |
| Media | Media Services | | | | |
| Commerce | Marketplace | | | | |
| SDK / modèles de développement | .NET | Java | PHP | Python | Node.js |

PaaS Windows Azure

- * Partie IaaS et SaaS non traitée (mais vous pouvez louer des VMs)
- * PaaS avec types de services proposés très large:
 - * Langages:
 - * C# VB bien sur
 - * Python
 - * Java avec un serveur d'application tel que tomcat ou autre
 - * PHP (voir en fin)
 - * Ruby,etc.
 - * Types d'applications:
 - * Services Web SOAP, REST, plain/text,
 - * Sites web, applications en worker role
 - * Plusieurs types de services à louer (service bus et autres)
 - * Consoles d'administration, analyse de performance, etc.

Resp: S. SALVA IUT licence pro

Windows Azure

The diagram illustrates the Windows Azure architecture, centered around the Management Portal. The Management Portal is connected to several enterprise-level infrastructure services:

- Execution Model:** Cloud Services, Web Sites, Virtual Machines, Mobile Services.
- Data Management:** Table Storage, Blob Storage, SQL Database, DocumentDB.
- Connectivity:** Connect, Virtual Network, Traffic Manager, Queue Storage Service.
- Business Analytics:** SQL Reporting, Hadoop.
- Performance (caching):** Content Delivery Network (CDN), Caching.
- Identity:** Windows Azure Active Directory, Access Control, Graph API, Audit Library.
- Media:** Media Services.
- Commerce:** Marketplace, Data, Applications.

Local Development Environment: Local Development Environment, Development Tools (Windows Azure Tools for Visual Studio, DotNet), Windows Azure SDK(s) (.NET, Java, PHP, Python, others).

Service Bus: permet d'exposer des services simplement, le bus se chargeant du routage des requêtes vers le service concerné.

Accès control: permet de gérer l'accès au Service Bus suivant des mécanismes standards tels que OAuth et les Simple Web Tokens (SWT) pour les services REST, ou encore des mécanismes à base de revendications de type SAML, WS-Federation et WS-Trust pour l'accès à des services SOAP

Composite App Service et Composition Model: fournissent un environnement de développement pour faciliter la création, la gestion et le déploiement d'applications composées.

Cloud services: SOAP Restweb services, web role, worker roles

14

Windows Azure

The diagram illustrates the Windows Azure architecture, divided into three main sections:

- Management Portal:** Provides access to various management tools and services.
- Enterprise Level Infrastructure:** Includes:
 - Execution Model:** Offers Cloud Services, Web Sites, Virtual Machines, and Mobile Services.
 - Data Management:** Provides Table Storage, Blob Storage Service, SQL Database, and Queue Storage Service.
 - Connectivity:** Features Connect (Virtual Network, Traffic Manager), Messaging (Service Bus, Relays, Brokered), and Queue Storage Service.
 - Business Analytics:** Includes SQL Reporting and Hadoop.
 - Performance (Caching):** Offers Content Delivery Network (CDN) and Caching.
 - Identity:** Manages Windows Azure Active Directory, Access Control, Graph API, and Audit Library.
 - Media:** Provides Media Services.
 - Commerce:** Offers Marketplace, Data, and Applications.
- Local Development Environment:** Contains Development Tools (Windows Azure Tools for Visual Studio, Other) and Windows Azure SDK(s) for .NET, Java, PHP, and Python.

Blobs: blob files allowing to store files or meta-data
Table: non relational tables, fulfilled with entities,
Queue asynchronous FIFO between apps
Drive manage and configure virtual disks

15

PaaS Windows Azure

Les possibilités offertes par ServiceBus sont nombreuses:
 Ex: Utiliser une FIFO pour la réception de messages
<https://www.windowsazure.com/en-us/develop/net/how-to-guides/service-bus-queues/>

The diagram shows the Service Bus Queue architecture:

- Message Sender:** Contains Web App, Mobile App, and Service components.
- Service Bus Namespace:** Contains a central **Queue**.
- Message Receiver:** Contains Service Or Application.

Arrows indicate communication between the sender and receiver via the queue.

Permet un envoi de messages de façon asynchrone: lecture asynchrone, le sender n'a pas à attendre une réponse.

Resp: S. SALVA IUT licence pro

PaaS Windows Azure

ServiceBus, d'autres possibilités:

Relay messaging: Utilisation d'un relay entre entités
 Possibilité de construire des applications hybrides déployées dans Windows Azure ou autre
 Sécurisation de l'ensemble via le relay
<https://www.windowsazure.com/en-us/develop/net/how-to-guides/service-bus-relay/>

Resp: S. SALVA *IUT licence pro*

PaaS Windows Azure

ServiceBus, d'autres possibilités:

Brokered messaging: stockage intermédiaire de haut capacité et durable des messages peuvent être stockés et traités
 les deux extrémités peuvent être complètement hétérogène en terme de puissance
 Elles peuvent être en ligne ou non

Resp: S. SALVA *IUT licence pro*

PaaS Windows Azure

- * D'autres paradigmes (Windows Azure)

- * Rôles d'applications avec commutation de rôles:
 - * web role : service ou appli Web
 - * worker role : démons persistant qui peut recevoir des données d'une autre appli
- * Chaque application est un composant (couche appFabric assure la connectivité)

Resp: S. SALVA

IUT licence pro

PaaS Windows Azure

- * D'autres paradigmes (Web et worker roles)

- * Web Role:
 - * Application pour Requêtes / réponses sur HTTP (pages, WCF Web service configurés par basichttpbinding, etc.)
- * Worker role:
 - * Application de type service fonctionnant en tache de fond. N'accepte pas de requête de l'extérieur
- * Web roles et worker roles peuvent dialoguer ensemble via des objets Queues: classiquement worker produit des données, le web role les lit à la demande de l'utilisateur et produit un affichage

Resp: S. SALVA

IUT licence pro

PaaS Windows Azure

- * D'autres paradigmes (Web et worker roles)
- * Une application peut changer d'état!
- * Un Web service a généralement un Web role mais peut également être implanté en worker role
- * Web roles et worker roles peuvent être placés dans des VM roles (distribution manuelle sur des VM différentes)

A massively scalable web app with background processing

Users

Resp: S. SALVA IUT licence pro

PaaS Windows Azure

- * D'autres paradigmes (Windows Azure)

Un worker role contient en plus un méthode run()

Resp: S. SALVA IUT licence pro

PaaS Windows Azure

- * Sécurité :
 - * Par AppFabric Access control service => propose WS-Trust, HTTPS, token,...
- * Envoi multicast
- * Buffer partagé de type fifo : pour effectuer des partages de données rapidement

Resp: S. SALVA

IUT licence pro

PaaS Windows Azure

- Implantation de services
- Utilisation de WCF services:
- Depuis .Net 3.0, Microsoft propose un nouveau framework de programmation orienté SOA: WCF
 - WCF: Windows Communication Foundation: framework pour création d'applications orientées service sur HTTP
 - Utilisation très simple
 - Envois asynchrones possible
 - Exposition en plain/text, Rest, SOAP, etc.

Resp: S. SALVA

IUT licence pro

PaaS Windows Azure

Ex en c# (new projet /WCF / WCF library)
 Définition de l'interface du service (Contrat et OperationContrat)
 => Fichier nom_de_la_classe

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ServiceModel;

namespace MathsLibrary
{
    [ServiceContract]
    public interface IMathsOperations
    {
        [OperationContract]
        int Add(int num1, int num2);
        [OperationContract]
        int Multiply(int num1, int num2);
    }
}
```

Resp: S. SALVA

IUT licence pro

PaaS Windows Azure

Ex en c# (new projet /WCF / WCF library)
 Implantation du service
 => Fichier nom_de_la_classe

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MathsLibrary
{
    public class MathsOperations : IMathsOperations
    {
        #region IMathsOperations Members

        public int Add(int num1, int num2)
        {
            return num1 + num2;
        }

        public int Multiply(int num1, int num2)
        {
            return num1 * num2;
        }

        #endregion
    }
}
```

Resp: S. SALVA

IUT licence pro

PaaS Windows Azure

Ex en c# (new projet /WCF / WCF library)

Configuration du contrat:

Resp: S. SALVA IUT licence pro

PaaS Windows Azure Client à un service

- * Exemple: <http://msdn.microsoft.com/en-us/library/windowsazure/1gg651130.aspx>
- * Création du service :
- * Définition de l'interface et définition du code

```
C#
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WCFServiceWebRole1
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]
        string GetHello();
    }
}
```

```
C#
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WCFServiceWebRole1
{
    public class Service1 : IService1
    {
        public string GetHello()
        {
            return "Hello from my WCF service in Windows
Azure!";
        }
    }
}
```

Resp: S. SALVA IUT licence pro

- PaaS Windows Azure
Client à un service
- * Exemple: <http://msdn.microsoft.com/en-us/library/windowsazure/lgg65130.aspx>
 - * Création Client : Ajout d'une référence (comme sur Netbeans), génération de squelettes et complétion du code généré dans Program.cs

```
C#
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

// Modify the following line to use your project name
// If your project is not named ConsoleApplication1.
using ConsoleApplication1.ServiceReference1;

// Modify the following line to use your project name
// If your project is not named ConsoleApplication1.
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Service1Client client = null;
```

Resp: S. SALVA

IUT licence pro

```
try
{
    client = new Service1Client();
    GetHelloRequest request = new
GetHelloRequest();
    GetHelloResponse response;
    response = client.GetHello(request);
    Console.WriteLine("The WCF service called
returned: '{0}'",
                      response.GetHelloResult);
}
catch (Exception e)
{
    Console.WriteLine("Exception encounter: {0}",
                      e.Message);
}
finally
{
    if (null != client)
    {
        client.Close();
    }
}
}
```



Resp: S. SALVA

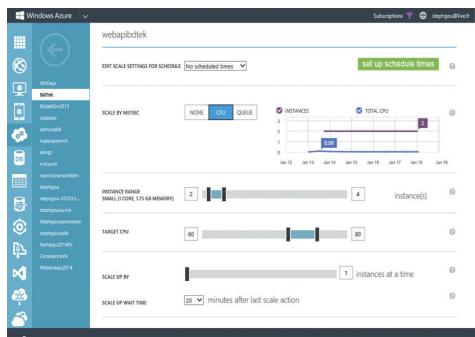
IUT licence pro

PaaS Windows Azure

- * Utilisation de PHP possible
- * <http://azurephp.interoperabilitybridges.com/tutorials>
- * Utilisation de IIS (installation manuelle ou automatique)
- * API de développement pour manipuler SQL Azure (blobs, queues, etc.)
- * Outils d'analyse de performance

Resp: S. SALVA IUT licence pro

Interface de configuration

Introduction à GAE (google App Engine)

PaaS Google App engine

- * Service d'exécution en Python, Java, Go
- * Actuellement gratuit pour une appli avec accès < 5millions/mois
- * Services

| | | | | | | | |
|-----------|--------------|---------|---------------|-----------|------------|--------|-----|
| CloudSQL | Blobstore | Compute | Auth security | SMS, Mail | Task queue | Search | ... |
| Datastore | cloudstorage | | | | | | |

PaaS Google App engine

Google Cloud Platform

| Category | Service | Icon |
|-------------------|-----------------|-------------------------------------|
| Hosting + Compute | App Engine | Icon: Target with arrow |
| | Compute Engine | Icon: Processor chip |
| | | |
| Storage | Cloud Storage | Icon: Two stacked boxes |
| | Cloud Datastore | Icon: Grid with three rows |
| | Cloud SQL | Icon: SQL logo |
| Big Data | BigQuery | Icon: Magnifying glass over a chart |
| | | |
| | | |
| Services | Cloud Endpoints | Icon: Double-headed arrow |
| | Translate API | Icon: Document with arrows |
| | Prediction API | Icon: Brain with arrows |

Resp: S. SALVA *IUT licence pro*

PaaS Google App engine

- * Types d'applications Java :
- * Gestion des VMs (à base de Java) via Docker
- * servlet/JSP, services Web en Rest, app GWT
- * Plusieurs librairies Java supportées mais pas toutes
- * Implantation JAX-RS Jersey supportée (1 & 2)
 - * Par maven
 - * Par Eclipse + plugin Google

Resp: S. SALVA *IUT licence pro*

PaaS Google App engine

- * Des limitations:
- * Pas de connexion TCP
 - * Connexion URLConnect pour effectuer des appels entre pl. servlets
- * Pas de processus
- * Timeouts limités
- * Quota
- * Version gratuite lente (2014)

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

- * Installation avec Eclipse (Google Cloud Platform)
 - * <https://cloud.google.com/eclipse/docs/quickstart>
- * Créer une application, et la déployer
 - * <https://cloud.google.com/eclipse/docs/creating-new-webapp>
 - * <https://cloud.google.com/appengine/docs/standard/java?cs=1>
- * Pour déployer une application web, il est nécessaire de créer une application dans la console google
<https://console.cloud.google.com/appengine> => fournir un id
- * Déploiement aussi en local et appel avec `http://localhost:8888/`

Resp: S. SALVA

IUT licence pro

GAE, création d' une application

Google Cloud Platform - info3app0

App Engine Tableau de bord

Résumé des tâches

Résumé des instances

Etat de facturation

Resource Utilisation

Données stockées dans le Blobstore 0,000088 sur 5 Go

Stockage du code et des fichiers statiques 0,02 sur 1 Go

<https://console.cloud.google.com/appengine/taskqueues?project=info3app0&serviceId=default>

Resp: S. SALVA IUT licence pro

GAE, portail

Google Cloud Platform

Accueil

Entités

Calcul

App Engine

Compute Engine

Kubernetes Engine

Cloud Functions

Stockage

Bigtable

Datastore

Tableau de bord

Index

Admin

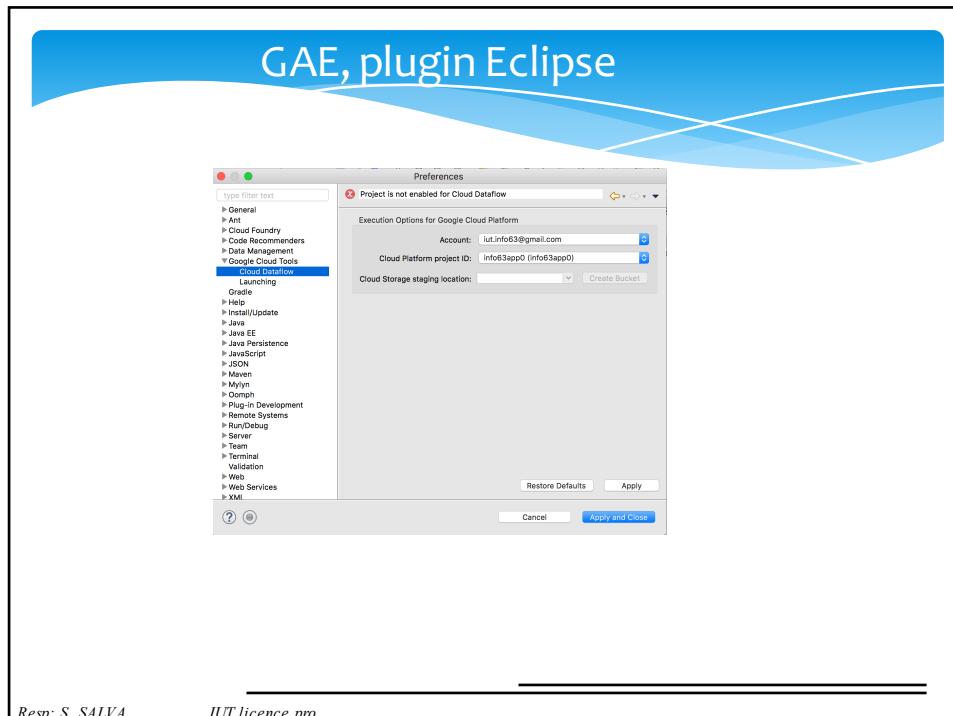
Réseau

Réseau VPC

Services réseau

Appel d'une application web sur votre env:
[http://version.id_app.appspot.com/nom_servlet_ou ws](http://version.id_app.appspot.com/nom_servlet_ou_ws)
ex: http://1-dot-ssalva-test.appspot.com/Service/rest/gae_restws

Resp: S. SALVA IUT licence pro



GAE, test de l'application en local

Firefox - Versions - ssalva-test - Hello App Engine - Unable to add a new ... - http://localhost:8888/_ah/admin - ssalva-test Developm... - ssalva-test Developm... - +

localhost:8888/_ah/admin datastore?kind=Employee

Google App Engine

ssalva-test Development Console

Datastore Viewer

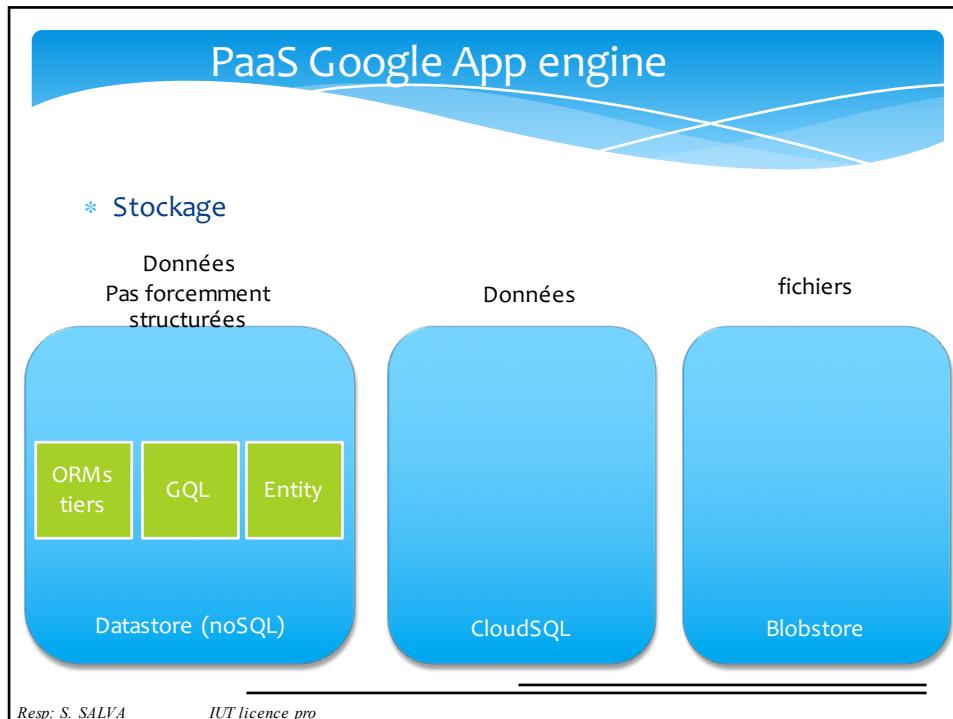
| Key | Write Ops | ID/Name | firstName | hireDate | lastName |
|--|-----------|---------|-----------|-------------------------------|----------|
| agtzc2FsdmEtdGVzdHlOCxlRW1wbG95ZWUYAQw | 8 | 1 | Alfred | Tue Sep 18 16:55:26 CEST 2012 | Smith |

©2008-2011 Google

http://localhost:8888/_ah/admin => console d'administration locale (affichage du blob de données, des services, etc.)

http://localhost:8888/mon_servlet ou service

Resp: S. SALVA IUT licence pro



PaaS Google App engine

- * Stockage dans CloudSQL
- 1. Création de tables
- 2. utilisation de jdbc:odbc classique

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

Servlet:

```
Public class GuestbookServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp) {
        try {
            Connection c = null;
            DriverManager.registerDriver(new AppEngineDriver());
            c = DriverManager.getConnection("jdbc:google:rdbms://simple-it.fr:testcloudsql:test-cloudsql/guestbook");

            ResultSet resultats = c.createStatement().executeQuery("SELECT name, message FROM messages
ORDER BY id DESC LIMIT 20");
            req.setAttribute("messages", resultats);
            this.getServletContext().getRequestDispatcher("/WEB-INF/guestbook.jsp").forward(req, resp);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

- * Datastore
- * MAP (clé valeur)
- * Plusieurs accès possibles (entity, Rest, ...)
- * Persistance bas niveau via des objets Entity
 - * Génial pour stockage
 - * Limité pour effectuer des requêtes (requêtes par clé ou id)
 - * Possibilité d'imbriquer des entités ensemble
- * <https://cloud.google.com/datastore/docs/concepts/entities>

Resp: S. SALVA *IUT licence pro*

PaaS Google App engine

The screenshot shows the Google Cloud Platform Datastore interface. On the left, there's a sidebar with options: Entités (selected), Tableau de bord, Index, and Admin. The main area has tabs for 'Entités' (selected), 'CRÉER UNE ENTITÉ', 'ACTUALISER', and 'SUPPRIMER'. There are filters for 'Requête par genre' (Genre) set to 'Employee' and 'Requête avec GQL'. A search bar and a 'Filterer les entités' button are also present. The results table shows three rows of data:

| Nom/Identifiant | name |
|---------------------|------|
| id=5639445604728832 | toto |
| nom=first | tutu |

At the bottom right, there's a dropdown for 'Nombre de colonnes à afficher' (50).

Resp: S. SALVA *IUT licence pro*

The screenshot shows the Google Cloud Platform Datastore interface. The left sidebar has options: Datastore, Entités (selected), Tableau de bord, Index, and Admin. The main area is titled "Entités" and shows a table with columns "Nom/Identifiant" and "name". A dropdown menu "Genre" is set to "Employee". A search bar "Requête avec GQL" is present. At the bottom right, there's a "Nombre de colonnes à afficher" dropdown set to 50. The table contains three rows:

| Nom/Identifiant | name |
|---------------------|------|
| id=5639445604728832 | toto |
| nom=first | tutu |

Resp: S. SALVA *IUT licence pro*

- * Datastore identités:
- * gcloud auth application-default login
 - * permet de donner votre identité pour accès au datastore (et au reste?)
 - * A utiliser pour des tests
- * sinon génération d'un fichier .json qui doit être dans votre env. de développement (export)
 - * <https://cloud.google.com/docs/authentication/getting-started>

Resp: S. SALVA *IUT licence pro*

PaaS Google App engine

- * Datastore
- * CRUD avec mot clés get, put, delete update

Création d'une instance Entity

<https://cloud.google.com/datastore/docs/reference/libraries>

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

- * Datastore

```
Datastore datastore
    =DatastoreOptions.newBuilder().setProjectId("info63appo").build().getService();
String kind = "Employee";
String name = "first";
// The Cloud Datastore key for the new entity
Key taskKey = datastore.newKeyFactory().setKind(kind).newKey(name);

// Prepares the new entity
Entity employee = Entity.newBuilder(taskKey)
    .set("name", t)
    .build();

// Saves the entity
datastore.put(employee);
```

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

- * Datastore

```
//Retrieve entity
Entity retrieved = datastore.get(taskKey);

System.out.printf("Retrieved %s: %s%n", taskKey.getName(),
retrieved.getString("name"));
```

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

- * Datastore

Récupération avec requêtes

```
Query<Entity> query =
    Query.newEntityQueryBuilder().setKind("Employee").setOrderBy(OrderBy.asc(
        c("name"))).build();
Iterator<Entity> it=datastore.run(query);
String ret=<html><body>";

while (it.hasNext()) {
    Entity e = it.next();ret+="iteration";
    ret+=e.getString("name")+"<br>";
}
```

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

- * Datastore

Récupération avec requêtes au exemple avec GQL (voir doc)
https://cloud.google.com/datastore/docs/reference/gql_reference

```
String kind = "my_kind"; String gqlQuery = "select * from " + kind; Query<Entity>
query = Query.newGqlQueryBuilder(Query.ResultType.ENTITY,
gqlQuery).build(); QueryResults<Entity> results = datastore.run(query); // Use
results }
```

Resp: S. SALVA *IUT licence pro*

PaaS Google App engine

- * Datastore

Utilisation de l'API Rest
<https://cloud.google.com/datastore/docs/reference/rest/>

| | |
|-------------------------|--|
| <u>beginTransaction</u> | POST /v1/projects/{projectId}beginTransaction Begins a new transaction. |
| <u>commit</u> | POST /v1/projects/{projectId}:commit Commits a transaction, optionally creating, deleting or modifying some entities. |
| <u>rollback</u> | POST /v1/projects/{projectId}:rollback Rolls back a transaction. |
| <u>runQuery</u> | POST /v1/projects/{projectId}:runQuery Queries for entities. |

Resp: S. SALVA *IUT licence pro*

PaaS Google App engine

- * Datastore
- * D'autres possibilités
- * App engine et Entiry (une autre façon, viable combien de temps ?)

- * Objectify (<https://github.com/objectify/objectify>)
- * Java data access API specifically designed for the Google App Engine datastore. I

Resp: S. SALVA

IUT licence pro

PaaS Google App engine

- * Datastore
- * Objectify (<https://github.com/objectify/objectify>)

```

@Entity class Car {
@Id String vin; // Can be Long, long, or String
String color; }

ofy().save().entity(new Car("123123", "red")).now();

Car c = ofy().load().type(Car.class).id("123123").now();
// ou.filter("color", "red").
ofy().delete().entity(c);

```

Resp: S. SALVA

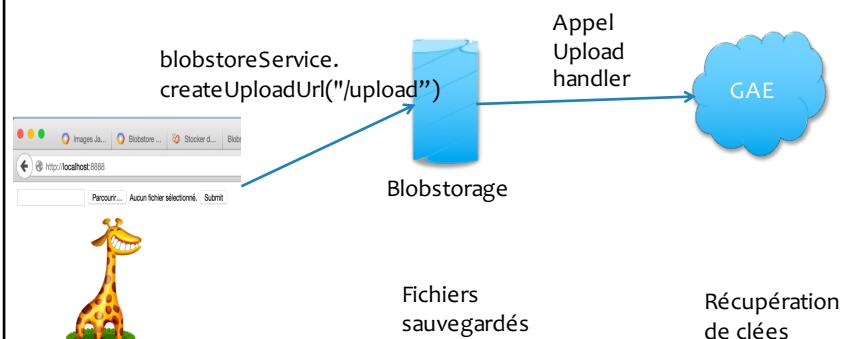
IUT licence pro

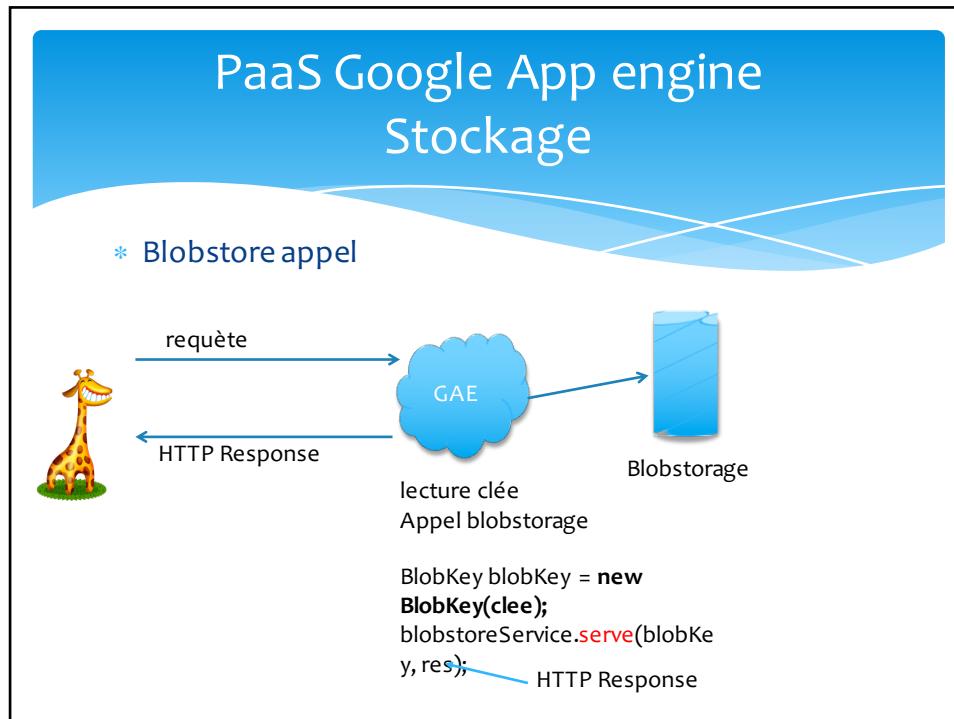
PaaS Google App engine Stockage

- * Blobstore
 - * <https://cloud.google.com/appengine/docs/java/blobstore/>
 - * Espace de stockage de fichiers
 - * Moins complexe que CloudStorage
 - * Accessible directement dans les applications

PaaS Google App engine Stockage

- * Blobstore sauvegarde





PaaS Google App engine Stockage

- * Blobstore
- * Principe:
 - * Implantation du upload handler
 - * Stockage des clés des éléments
 - * À l'appel de l'handler, le fichier est déjà stocké
 - * Ex:


```
Map<String, List<BlobKey>> blobs = blobstoreService.getUploads(req);
List<BlobKey> blobKeys = blobs.get("Fichier");

if (blobKeys == null || blobKeys.isEmpty()){
    res.sendRedirect("/");
} else {
    //renvoi vers autre ressource
    res.sendRedirect("/serve?clee=" + blobKeys.get(0).getKeyString());
}
```

Objets
 req=htt
 pServlet
 Request
 Res=htt
 pServlet
 Respons
 e

PaaS Google App engine Stockage

- * Blobstore
 - * Principe:
 - * Récupération
- ```
public void doGet(HttpServletRequest req, HttpServletResponse
res)
throws IOException{
 BlobKey blobKey = new
 BlobKey(req.getParameter("clee"));
 blobstoreService.serve(blobKey, res);
```

## PaaS Google App engine

Déploiement d'un service Rest Jersey 2 (eclipse):

1. Créer un projet Google App Engine project,
2. Ajouter les lib de Jersey dans le projet eclipse et dans le projet (WEB-INF/lib)
3. Implanter un WS:

```
package wsrest;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
@Path("/helloworld")
public class HelloWorldResources {
 @GET
 @Produces("text/plain")
 public String getClickedMessage() {
 return "Hello World";
 }
}
```

Resp: S. SALVA

IUT licence pro

## PaaS Google App engine

### 4. Modifier Web.xml

```
<servlet>
 <servlet-name>myrest</servlet-name>
 <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class> Nom du service
 <init-param>
 <param-name>jersey.config.server.provider.packages</param-name>
 <param-value>wsrest</param-value>
 </init-param> Nom du package
 <init-param>
 <param-name>unit.WidgetPU</param-name>
 <param-value>persistence/widget</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
 <servlet-name>myrest</servlet-name>
 <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Resp: S. SALVA

IUT licence pro

# GAE authentification

- \* Authentification :
- \* classe Userservice pour phase de login
- \* Émulé en local

**Not logged in**

Email: test@example.com

Sign in as Administrator

**Log In** **Log Out**

- \* Utilise Google account
- \* en déployé



Resp: S. SALVA      IUT licence pro

# GAE authentification

- \* Authentification : Donnez les permissions à vos applications

Google Developers Console

Sign up for a free trial. iut.info63@gmail.com

| Projects               |  | Add Member                                                      | Remove |
|------------------------|--|-----------------------------------------------------------------|--------|
| <b>info63app0</b>      |  |                                                                 |        |
| Overview               |  |                                                                 |        |
| Permissions            |  | <input type="checkbox"/> iut.info63@gmail.com (you)    Is owner |        |
| Billing & settings     |  |                                                                 |        |
| <br>                   |  |                                                                 |        |
| <b>APIs &amp; auth</b> |  |                                                                 |        |
| APIs                   |  |                                                                 |        |
| Credentials            |  |                                                                 |        |
| Consent screen         |  |                                                                 |        |
| Push                   |  |                                                                 |        |
| Monitoring             |  |                                                                 |        |
| Source Code            |  |                                                                 |        |

| Service accounts                                                            |  | PERMISSION                        |
|-----------------------------------------------------------------------------|--|-----------------------------------|
| <input type="checkbox"/> 278438158444-compute@developer.gserviceaccount.com |  | Google APIs service account       |
|                                                                             |  | <input type="checkbox"/> Can edit |

Resp: S. SALVA      IUT licence pro

## GAE authentification

- \* Authentification
- \* `UserService userService = UserServiceFactory.getUserService();`  
instancie le moteur d'authentification
- \* `userService.getCurrentUser()` retourne null si le client n'est pas connecté un objet User sinon
- \* `userService.createLoginURL("***URL***") et`  
`userService.createLogoutURL("***URL***")` génère la connexion ou deconnexion et renvoie vers une URL
- \* `userService.getCurrentUser().getNickname()` et l'e-mail avec  
`userService.getCurrentUser().getEmail()` retournent des infos

Resp: S. SALVA

IUT licence pro

## Introduction à Heroku

## Présentation héroku

- \* Offre PaaS (repose sur AWS) depuis 2007
- \* Langages:
  - \* Java, PHP, Ruby, Go, Scala, python, node.js
  - \* Des addons (redis, mongodb, etc.)
  - \* <https://addons.heroku.com/>
- \* Compte gratuit avec 1 dyno et accès BD (postgresql)  
<=10000 lignes



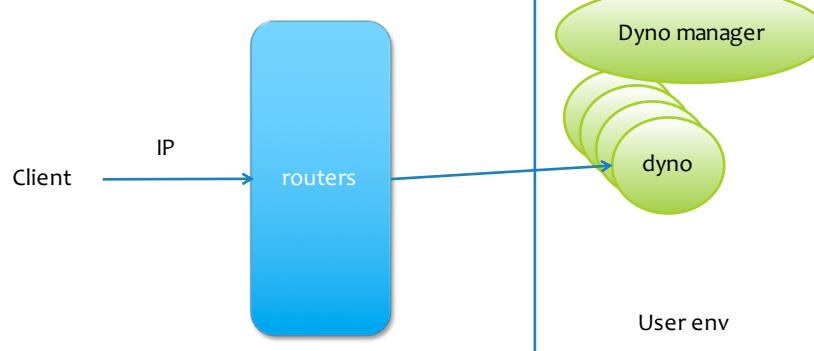
## Présentation héroku

- \* A base de Git
  - \* Push à partir de dépôts locaux ou Github
  - \* Utilisation de Maven pour gestion des dépendances
- \* A base de Debian,
  - \* Se contrôle via ligne de commande
  - \* Prix se calcule sur le nb de dyno (container debian) + addons
  - \* (mais aussi dropbox, travis, etc.)

## Présentation héroku

- \* dyno = container à base de cedar (ubuntu)
- \* Exécute une seule commande à la fois (1 instance de serveur)
- \* types de dyno:
  - \* web,
  - \* worker (background)
  - \* one-off dyno: dyno temporaire pour tâches d'admin (migration etc.) (ex: heroku run bash)

## Présentation héroku



## Héroku

- \* Prérequis (pour ce cours au moins):
  - \* Maven
    - \* Gest. De projet et de dépendances
    - \* Pom.xml-> décrit les deps (mvn clean install les télécharge et les installe)
  - \* Git
    - \* Add, commit et push
  - \* Heroku toolset

## Héroku

- \* Gestion des dynos
- \* <https://devcenter.heroku.com/articles/dynos>
- \* heroku ps:scale web=X , X nb d'instances
  - \* pour 1, 2 -> processus mis en veille après 1 heure
- \* heroku ps

## Héroku, applications Java

- \* <https://devcenter.heroku.com/articles/getting-started-with-java#introduction>
- \* Principe:
  1. Maven -> crée une application Web (et les tests), télécharge les deps
  2. Heroku create -> crée une appli sur Heroku
  3. Git add, git commit -> crée un dépôt local
  4. Git push -> lance les tests, upload l'application, la compile, la déploie

## Héroku, applications Java

- \* Gestion des dépendances dans pom.xml

```
<dependencies>
 <dependency>
 <groupId>org.glassfish.jersey.containers</groupId>
 <artifactId>jersey-container-servlet</artifactId>
 </dependency>

 <dependency>
 <groupId>org.eclipse.jetty</groupId>
 <artifactId>jetty-servlet</artifactId>
 <version>${jetty.version}</version>
 <scope>provided</scope>
 </dependency>

 <dependency>
 <groupId>org.eclipse.jetty</groupId>
 <artifactId>jetty-webapp</artifactId>
 <version>${jetty.version}</version>
 <scope>provided</scope>
 </dependency>
```

## Héroku, applications Java

- \* Gestion des dépendances dans pom.xml

```
<dependency>
<groupId>postgresql</groupId>
<artifactId>postgresql</artifactId>
<version>9.0-801.jdbc4</version>
</dependency>

</dependencies>
```

## Héroku, Web service Rest Jersey

- \* <https://jersey.java.net/documentation/latest/getting-started.html#heroku-webapp>

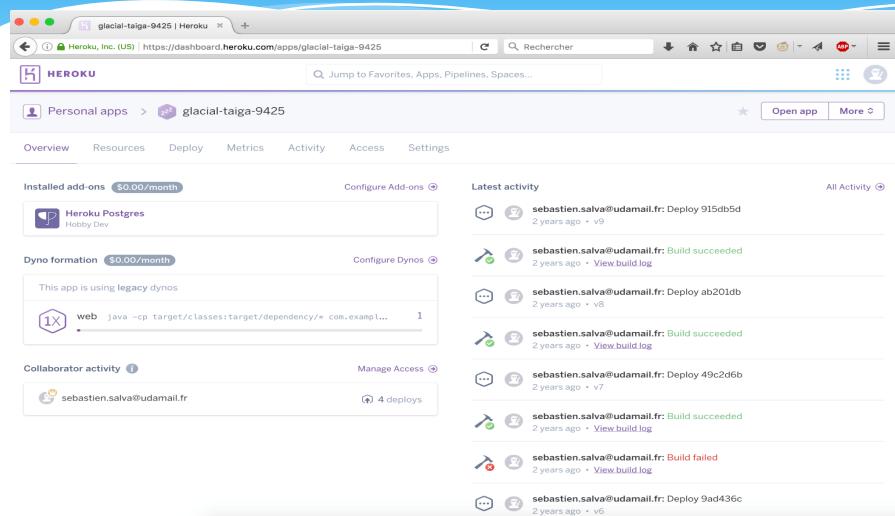
1. Création d'une application

```
mvn archetype:generate -DarchetypeArtifactId=jersey-heroku-webapp \
-DarchetypeGroupId=org.glassfish.jersey.archetypes \
-DinteractiveMode=false \
-DgroupId=com.example \
-DartifactId=simple-heroku-webapp \
-Dpackage=com.example \
-DarchetypeVersion=2.14
```

# Héroku, Web service Rest Jersey

2. Création d'une appli Web packagée
  - \* mvn clean package
  
3. Déploiement
  - \* Git init, heroku create, git add, git commit, git push,
  
4. Accès:
  - \* URL fourni par ligne de commande
  - \* Ex: <https://glacial-tiaga-9425.herokuapp.com/wsrest/appel>

# Héroku, interface d'administration



The screenshot shows the Heroku dashboard for the application 'glacial-tiaga-9425'. The top navigation bar includes links for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. Under the Overview section, it shows 'Installed add-ons' (Heroku Postgres) and 'Dyno formation' (1X web). The Dyno formation section notes that the app is using legacy dynos. The Activity section lists several deployment logs from 'sebastien.salva@udamail.fr' over the past two years, with some failing builds. The Collaborator activity section shows 4 deployments by 'sebastien.salva@udamail.fr'.

The screenshot shows the Heroku administration interface. At the top, there's a blue header with the text "Héroku, interface d'administration". Below the header, the Heroku logo is on the left, and a search bar with the placeholder "Jump to Favorites, Apps, Pipelines, Spaces..." is on the right. The main content area has a light gray background. At the top of this area, there's a navigation bar with tabs: Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. The "Resources" tab is currently selected. Below the navigation bar, there's a section titled "Legacy Dynos" with a sub-section "Change...". It shows one dyno entry: "1X web java -cp target/classes:target/dependency/\* com.example.heroku.Main". To the right of this entry, it says "Dyno Count: 1 \$0.00". Below this, there's a section titled "Add-ons" with a search bar "Quickly add add-ons from Elements". A single add-on is listed: "Heroku Postgres :: Bronze" (Hobby Dev, Free). At the bottom, there's a note about "Estimated Monthly Cost" being \$0.00.

The screenshot shows the Heroku administration interface with a blue header "Héroku, interface d'administration". On the left, there's a sidebar titled "Add-on Categories" listing various service categories. The main content area displays a grid of nine add-on cards. The cards are arranged in three rows of three. From top-left to bottom-right, the cards are: "ClearDB MySQL" (The high speed database for your MySQL powered applications.), "mLab MongoDB" (Cloud-hosted MongoDB), "Redis Cloud" (Enterprise-Class Redis for Developers); "JawsDB MySQL" (The database you trust, with the power and reliability you need.), "Citus" (Horizontally scalable Postgres), "Heroku Postgres" (Reliable and powerful database as a service based on PostgreSQL.); "Compose MongoDB" (Hosted, optimized and super-fast MongoDB databases.), "Redis To Go" (#1 Redis Provider with over 50,000 Redis instances.), "ObjectRocket for MongoDB" (High-Performance MongoDB with Fanatical Support™ and DBAs); "Instaclustr" (openredis) (Instaclustr), "openredis" (openredis), and "Treasure Data" (Treasure Data).

## Choregraphie, orchestration ?

## Composition de services Web

- \* Composition
  - \* Faire interagir des services Web ensemble
  - \* Déployés sur le même serveur, ou sur des Clouds/serveurs différents
- \* La composition appelé service composite, services invoqués appelé des composants de service
- \* D'un point de vue Client, service composite = service
- \* 2 types de composition
  - \* Orchestration
  - \* chorégraphie

## Composition de services Web

- \* Difficultés:
  - \* Gestion des erreurs
  - \* Si 1 composant remonte une erreur, elle doit être gérée par le service appelant pour un retour vers le client
  
- \* 2 types de composition
  - \* Orchestration
  - \* chorégraphie

Resp: S. SALVA

IUT licence pro

## Gestion des services web

### Orchestration des services

- Lorsqu'un service web coordonne d'autres services
- 1 processus global avec appel vers d'autres services, gestion des erreurs
- Compositions simples en Java etc.
- Compositions complexe, besoin de meta langages -> BPEL,

Resp: S. SALVA

IUT licence pro

## Gestion des services web

### Orchestration des services

- Langage BPEL

- processus BPEL (processus écrit en XML qui décrit comment interagissent les WS suivant des stimuli extérieurs)

- Besoin d'un serveur qui exécute les processus BPEL  
la gestion des erreurs doit être gérée par le processus  
(mécanisme de replis, re-exécution du processus)

- Langage de programmation de processus mais aussi interface graphique (boîtes)

Resp: S. SALVA

IUT licence pro

## Aperçu de WSBPEL

- Définition des partenaires

- Utilisation de variables, assignation de valeurs (assign)

- Activités basiques (invoke, receive, reply, wait, throw)

- Activités structurés (while, switch, sequence, pick(temporisation))

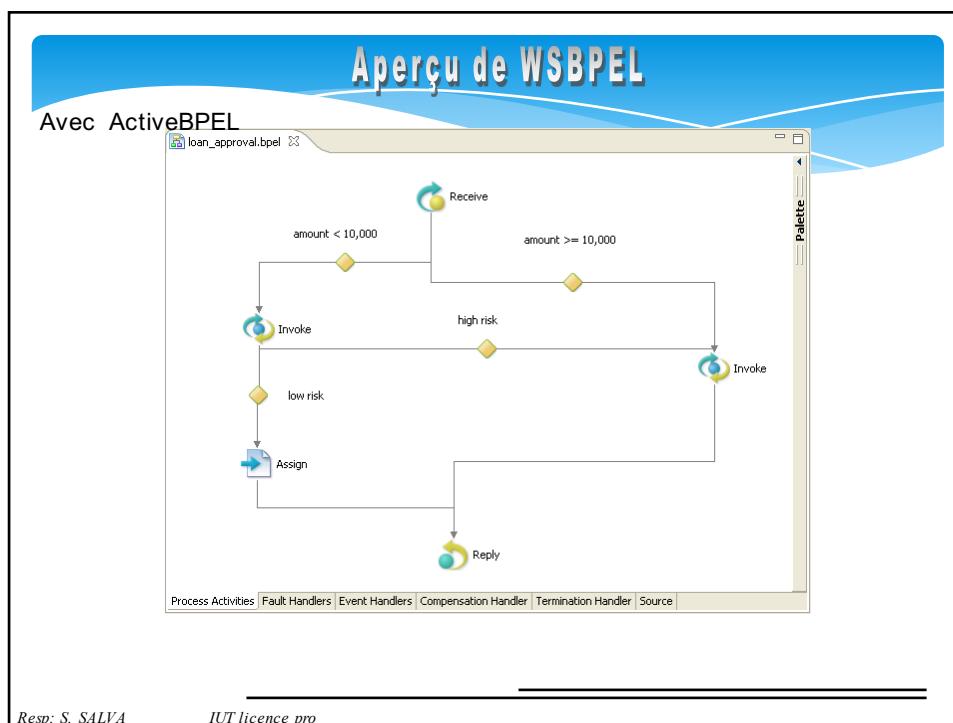
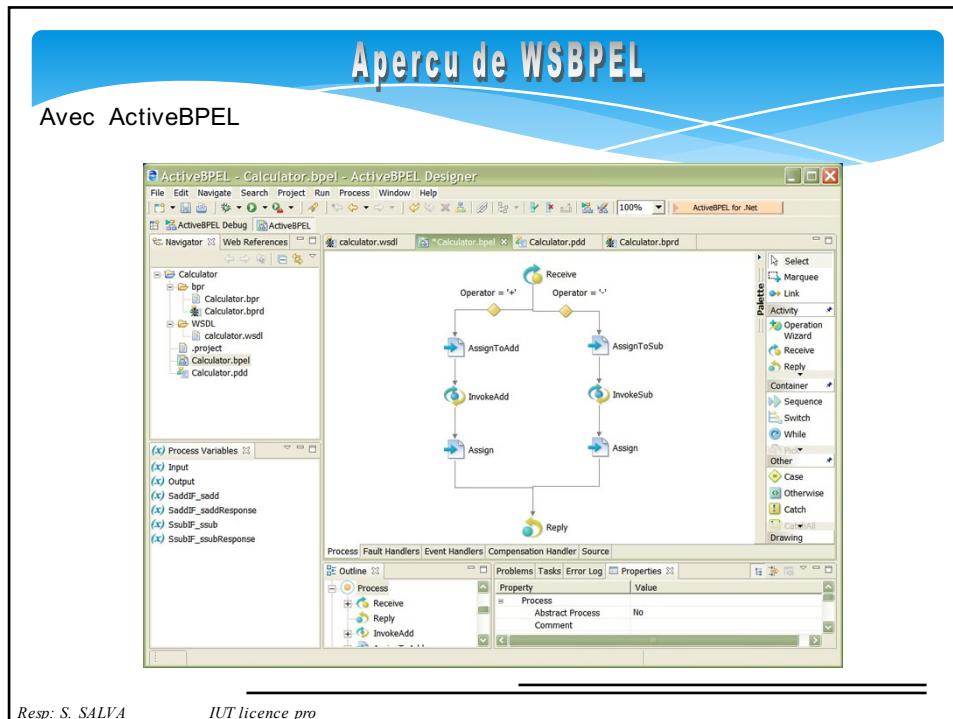
- Corrélation = session

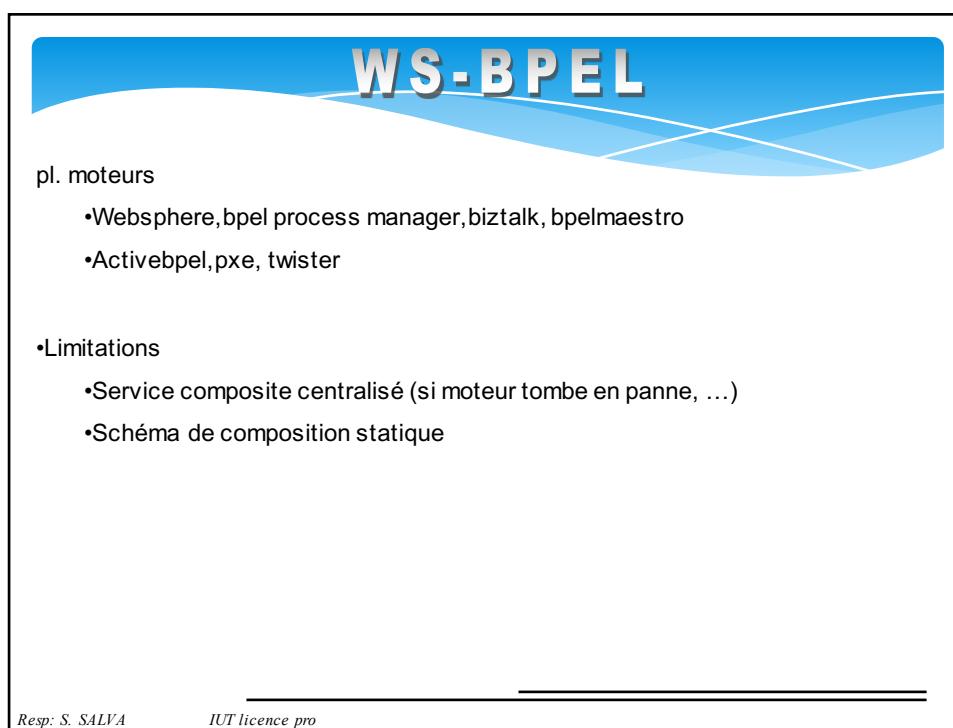
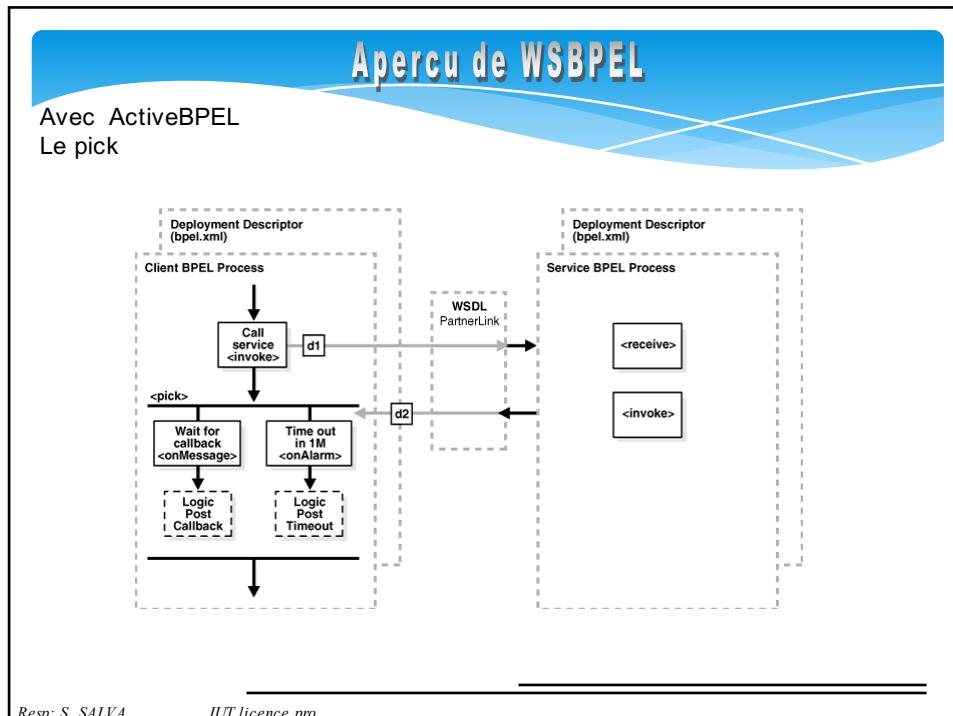
- Scope découpage d'un processus en plusieurs parties

- Pl. handler possibles par scope (compensation, fault, event)

Resp: S. SALVA

IUT licence pro





## Gestion des services web

### Chorégraphie de services

- Comportement global basé sur les interactions des services entre eux.
- Chaque service web mêlée dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu.
- Services font office de services et de clients vers d'autres services

Resp: S. SALVA

IUT licence pro

## Gestion des services web

### Chorégraphie de services

- Description des interactions de service uniquement de pair à pair
- Pas de processus, chaque service connaît les actions à effectuer par rapport aux messages reçus
- Langages standards de description de choreographies
  - en XML WS-CL ou WSCI
  - Description des messages
  - Ordre des messages
  - ne définit pas un processus global
- Travaux de recherche sur composition dynamique

Resp: S. SALVA

IUT licence pro