

Cloud computing

S. Salva 1

---

---

---

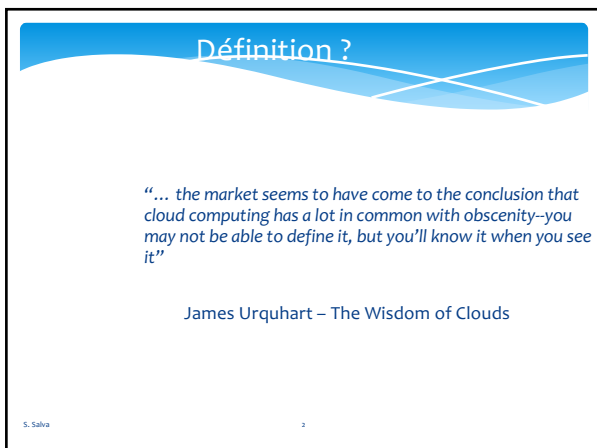
---

---

---

---

---



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

S. Salva 2

---

---

---

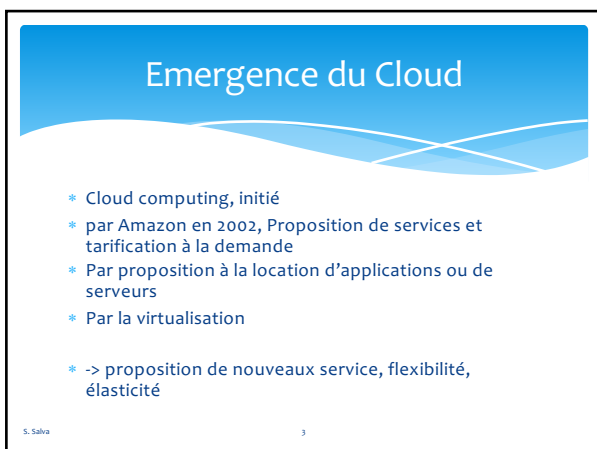
---

---

---

---

---



Emergence du Cloud

- \* Cloud computing, initié
- \* par Amazon en 2002, Proposition de services et tarification à la demande
- \* Par proposition à la location d'applications ou de serveurs
- \* Par la virtualisation
- \* -> proposition de nouveaux service, flexibilité, élasticité

S. Salva 3

---

---

---

---

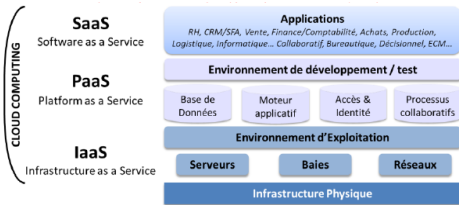
---

---

---

---

## Structuration d'un Cloud ?

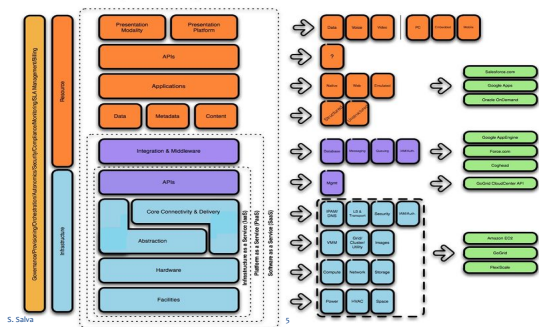


S. Salva

4

## Architecture

Cloud Taxonomy &amp; Ontology - Draft v1.4 - Hoff



S. Salva

5

## Architecture



- 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

S. Salva

6

## Architecture

Applications	Software as a Service (SaaS)
	Google Apps, salesforce.com, Microsoft 365
Frameworks	Platform as a Service (PaaS)
	Google App Engine, force.com, Windows Azure
Hardware	Infrastructure as a Service (IaaS)
	Amazon EC2, Microsoft Azure, 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

S. Salva 7

---

---

---

---

---

---

---

---

## Types d'architectures

- Cloud public: solutions de stockage et applications offertes au public par accès via Internet (Amazon, Microsoft, Google)
- 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 aux Clouds; flexibilité, évolutivité
  - Accès dans l'organisation

S. Salva 8

---

---

---

---

---

---

---

---

## Types d'architectures

- Cloud communautaire:
  - infrastructure partagée entre organisations.
  - 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,
  - données sensibles dans la partie privée

S. Salva 9

---

---

---

---

---




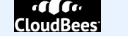
---

---

---

# Offres PaaS

Open source:

	Année	sponsors	plateforme
	2011	VMware	Spring,Rails, sinatra, node.js
	2011	Red hat	Express-ruby, PHP, python, flex, jboss, java EE6
	2009	WSO2	Tomcat, jboss, java EE6
	2012		Java EE6, tomcat, grails, scala, jruby

Autres: Google GAE, Windows Azure, Amazon EC2, IBM Cloud, Heroku, etc.  
(en IAAS, la liste est longue, Atos, SFR, Colt, Joyent, GoGrid, etc.)

S. Salva 10

---

---

---

---

---

---

---

---

# Aperçu de Windows Azure

S. Salva 11

---

---

---

---

---

---

---

---

# Introduction à GAE (google App Engine)

S. Salva 12

---

---

---

---

---

---

---

---

### Structuration

\* **App Engine** : service PaaS de Google

\* **Compute Engine** : service IaaS

\* **Cloud Storage**: stockage de fichiers

\* **Big Query** : des fonctionnalités permettant d'analyser de grosses quantités de données (bigdata)

\* **Cloud SQL** : une base de données MySQL distribuée dans le cloud

S. Salva

13

---

---

---

---

---

---

---

---

### PaaS Google App engine

#### Google Cloud Platform

Hosting + Compute

Storage

Big Data

Services

S. Salva

14

---

---

---

---

---

---

---

---

### PaaS Google App engine

#### Google App Engine

SDK

Logs

Google File System

Google Infrastructure

S. Salva

© 2010 Technology 2007-2009

15

16

---

---

---

---

---

---

---

---

## PaaS Google App engine

- \* Service d'exécution en Python, Java, Go
- \* Actuellement gratuit pour une appli avec accès < 5millions/mois
- \* Types d'applications : servlet/JSP, services Web en Rest, app GWT
  - \* Plusieurs librairies Java supportées mais pas toutes
  - \* Implantation JAX-RS Jersey supportée
- \* Authentification : classe userservice pour phase de login
  - \* Émulé en local
  - \* Utilise Google account en déployé

S. Salva

16

## PaaS Google App engine

- \* Des limitations:
- \* Pas de connexion TCP
  - \* Connexion URLConnect pour effectuer des appels entre pl. servlets
- \* Pas de création de threads, de processus
- \* Timeouts limités
- \* Quota
- \* Lent (2012)

S. Salva

17

## PaaS Google App engine

- \* Persistance sous forme d'objets dans le **Datastore**
  - \* noSQL, MAP multidimensionnelle
  - \* Couples (clé, valeur) avec valeur un objet etc.
- \* 3 api :
  - \* JDO (voir plus loin, JDOQL)
  - \* JPA (ORM, J2EE)
  - \* une api de bas niveau
- \* Egalement Projet Objectify : ORM à base de JPA
- \* En complément, MEMcache, idem Datastore mais en mémoire vive

S. Salva

18

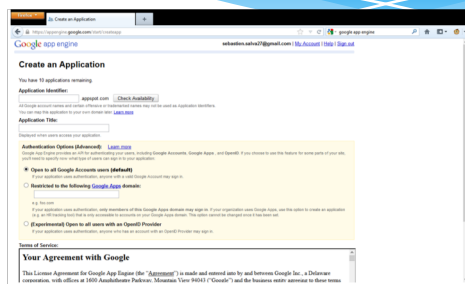
## PaaS Google App engine

- \* **Blobstore** : système de stockage de fichiers
  - \* stockage de fichiers d'une taille de 2 Go chacun maximum.
  - \* Vous disposez de 5 Go de stockage gratuit avec votre application App Engine.

S. Salva

19

## GAE, création d'une application



=> Id unique nécessaire pour le déploiement

S. Salva

20

GAE, portail



S. Salva

21

## GAE, portail



Appel d'une application web:  
[http://version.id\\_app.appspot.com/nom\\_servlet](http://version.id_app.appspot.com/nom_servlet) ou ws  
 ex: [http://s.salva-test.appspot.com/gae\\_restws](http://s.salva-test.appspot.com/gae_restws)

S. Salva 22

---

---

---

---

---

---

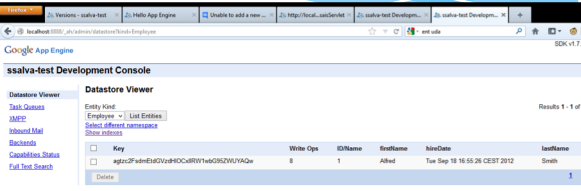
---

---

---

---

## GAE, test de l'application en local



[http://localhost:8888/\\_ah/admin](http://localhost:8888/_ah/admin) => console d'administration locale (affichage du blob de données, des services, etc.)  
[http://localhost:8888/mon\\_servlet](http://localhost:8888/mon_servlet) ou service

S. Salva 23

Resp: S. SALVA IUT licence pro

---

---

---

---

---

---

---

---

---

---

## PaaS Google App engine

\* Stockage

Données  
Pas forcément  
structurées

JPA   JDOQL   Entity

Datastore (noSQL)

Données

CloudSQL

fichiers

Blobstore

S. Salva 24

---

---

---

---

---

---

---

---

---

---



### PaaS Google App engine

- \* Stockage dans CloudSQL
  - \* = base Mysql
- \* Besoin d'activer l'api dans



Projects: info63app0

Enabled APIs

Some APIs are enabled automatically. You can disable them if you're not using their services.

NAME	QUOTA	STATUS
Cloud SQL API	10%	ON
Google Cloud SQL		ON

- \* Demande infos de paiement -> non vu en TP

S. Salva 25

---

---

---

---

---

---

---

---

### PaaS Google App engine

- \* Stockage dans CloudSQL

1. Création de tables
2. utilisation de jdbc:odbc classique

S. Salva 26

---

---

---

---

---

---

---

---

### PaaS Google App engine

- \* Datastore
  - \* MAP (clé valeur)
  - \* Plusieurs accès possibles (entity,JDO,JPA)
- \* Persistance 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/appengine/docs/java/datastore/entities>

S. Salva 27

---

---

---

---

---

---

---

---

PaaS Google App engine

Google Cloud Platform

Entités

Requête par genre Requête avec GQL

Genre: Account

Filtrer les entités

Non/Identifiant	account	lastName	job
name/Phone	0678000	Alban	low
name/Thomas	2	Anthony	high
name/Toto	200	Toto	high

Nombre de colonnes à afficher: 50

S. Salva 28

---

---

---

---

---

---

---

---

PaaS Google App engine

\* Datastore en Java

`DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();`

`Entity employee = new Entity("Employee", "salva");` ← Identifiant

`employee.setProperty("firstName", "sebastien");`  
`employee.setProperty("lastName", "Salva");`

`Date hireDate = new Date();`  
`employee.setProperty("hireDate", hireDate);`  
`datastore.put(employee);`

//imbrication en Entity  
`Entity contact = new Entity("contact1", "toto", employee.getKey());`

S. Salva 29

---

---

---

---

---

---

---

---

PaaS Google App engine

\* Datastore en Java (Mode Entity)

\* Maj d'une Entity => par put

\* Suppression => par delete

\* Lecture par clé

- \* `Key cle = KeyFactory.createKey("Employee", "salva");`
- \* `Entity Employeeetrouve = datastore.get(cle);`
- \* Puis utilisation de `getProperty(propriété)`

S. Salva 30

---

---

---

---

---

---

---

---

## PaaS Google App engine

### \* Datastore en Java (Mode Entity)

Possibilité de faire des requêtes:

```
...Query q =
    new Query("Person")
        .setFilter(new FilterPredicate("lastName", FilterOperator.EQUAL,
            targetLastName));

PreparedQuery pq = datastore.prepare(q);

for (Entity result : pq.asIterable()) {
    String firstName = (String) result.getProperty("firstName");
    String lastName = (String) result.getProperty("lastName");
    Long height = (Long) result.getProperty("height");

    System.out.println(firstName + " " + lastName + " " + height + " inches tall");
}
```

S. Salva

31

## PaaS Google App engine

### \* Datastore en Java (classes métier annotées)

- \* Persistance sous forme d'objet (pas d'SQL, MAP multidimensionnelle)
  - \* 3 api : JDO, JPA
- \* Def d'une classe avec des annotations @persistantcapable, @primarykey, @persistent
- \* Rendre persistant : méthode makepersistent (object)

S. Salva

32

## PaaS Google App engine

### Datastore en Java (classes métier annotées)

#### Exemple Accès aux données

```
import com.google.appengine.api.datastore.Key;

import java.util.Date;
import javax.jdo.annotations.IdGeneratorStrategy;
import javax.jdo.annotations.PersistenceCapable;
import javax.jdo.annotations.Persistent;
import javax.jdo.annotations.PrimaryKey;

@PersistenceCapable
public class Employee {
    @PrimaryKey
    @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
    private Key key;

    @Persistent
    private String firstName;
```

S. Salva

33

## PaaS Google App engine

```
@Persistent
private String lastName;
@Persistent
private Date hireDate;

public Employee(String firstName, String lastName, Date hireDate) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.hireDate = hireDate;
}

// Accessors for the fields. JDO doesn't use these, but your application does.

public Key getKey() {
    return key;
}

public String getFirstName() {
    return firstName;
}

// ... other accessors...
}
```

S. Salva

34

## PaaS Google App engine

### Datastore en Java (classes metier annotées)

#### Interaction avec le Datastore via une factory PersistenceManagerFactory

```
import javax.jdo.JDOHelper;
import javax.jdo.PersistenceManagerFactory;

public final class PMF {
    private static final PersistenceManagerFactory pmfInstance =
        JDOHelper.getPersistenceManagerFactory("transactions-optional");

    private PMF() {}

    public static PersistenceManagerFactory get() {
        return pmfInstance;
    }
}
```

S. Salva

35

## PaaS Google App engine

### Datastore en Java (classes metier annotées)

#### Stockage:

```
PersistenceManager pm = PMF.get().getPersistenceManager();

Employee e = new Employee("Mr", "true", new Date());

pm.makePersistent(e);
```

S. Salva

36

PaaS Google App engine

Datastore en Java (classes metier annotées)

Lecture:

```
//reading data
Query q = pm.newQuery(Employee.class);
q.setFilter("lastName == lastNameParam");
q.setOrdering("height desc");
q.declareParameters("String lastNameParam");

try {
    List<Person> results = (List<Employee>) q.execute("Smith");
    if (!results.isEmpty()) {
        for (Person p : results) {
            // Process result p
        }
    } else {
        // Handle "no results" case
    }
} finally {
    q.closeAll();
}
}
```

S. Salva 37

---

---

---

---

---

---

---

---

PaaS Google App engine

Datastore en Java (classes metier annotées)

Lecture:

```
//reading data
Query q = pm.newQuery("select from Employee " +
    "where lastName == lastNameParam " +
    "parameters String lastNameParam " +
    "order by height desc");

List<Employee> results = (List<Employee>) q.execute("Smith");
```

S. Salva 38

---

---

---

---

---

---

---

---

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

S. Salva 39

---

---

---

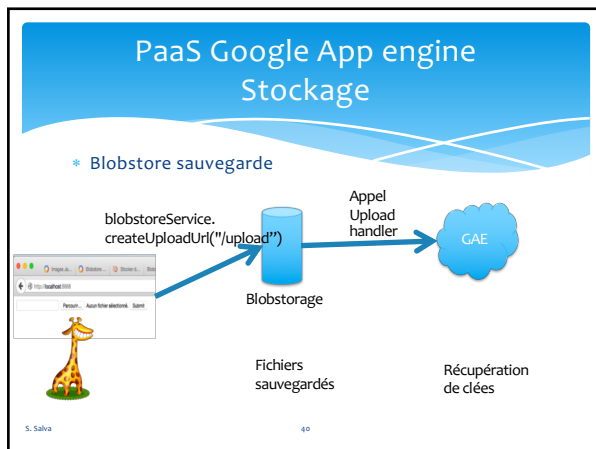
---

---

---

---

---




---

---

---

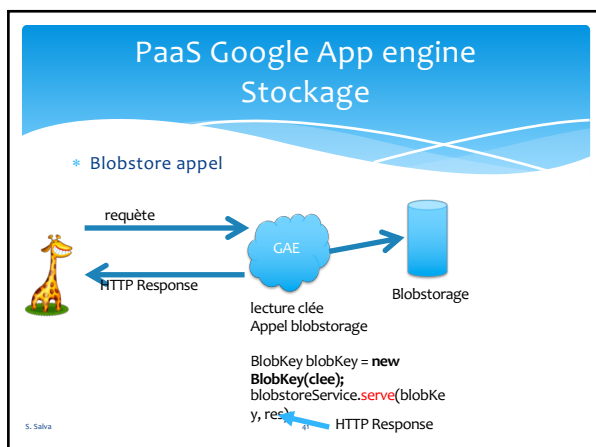
---

---

---

---

---




---

---

---

---

---

---

---

---




---

---

---

---

---

---

---

---

## 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?cle=" + blobKeys.get(0).getKeyString());
}
```

**Objets**  
 req=htt  
 pservlet  
 Request  
  
 Res=htt  
 pservlet  
 Respons  
 e

S. Salva 43

---

---

---

---

---

---

---

---

## PaaS Google App engine Stockage

- \* Blobstore
- \* Principe:
  - \* Récupération, Servlet avec URL Serve

```
public void doGet(HttpServletRequest req, HttpServletResponse
res)
throws IOException {
    BlobKey blobKey = new BlobKey(req.getParameter("cle"));
    blobstoreService.serve(blobKey, res);
}
```

S. Salva 44

---

---

---

---

---

---

---

---

## 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 getClichedMessage() {
        return "Hello World";
    }
}
```

S. Salva 45

---

---

---

---

---

---

---

---

## PaaS Google App engine

### 4. Modifier Web.xml

```

<servlet>
  <servlet-name>mvrest</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
  <init-param>
    <param-name>jersey.config.server.provider.packages</param-name>
    <param-value>wrest</param-value>
  </init-param>
  <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>mvrest</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>

```

Nom du service

Nom du package englobant le SW

S. Salva 46

---

---

---

---

---

---

---

---

## GAE authentication

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

Not logged in

Email: test@example.com

☐ Sign in as Administrator

- \* Utilise Google account
- \* en déployé

S. Salva 47

---

---

---

---

---

---

---

---

## GAE authentication

- \* Authentification : Donnez les permissions à vos applications

S. Salva 48

---

---

---

---

---

---

---

---



## GAE authentication

- \* 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

S. Salva

49

---

---

---

---

---

---

---

---

## Introduction a Heroku

S. Salva

50

---

---

---

---

---

---

---

---

## Présentation héroku

- \* Offre PaaS (repose sur AWS) depuis 2007
- \* Langues:
  - \* 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



S. Salva

51

---

---

---

---

---

---

---

---

## 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.)

S. Salva

52

---

---

---

---

---

---

---

---

## 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)

S. Salva

53

---

---

---

---

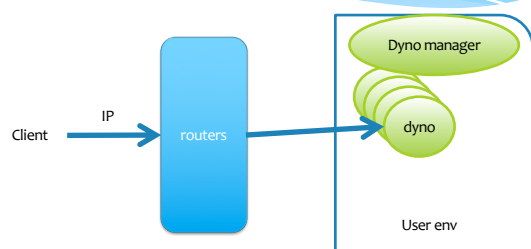
---

---

---

---

## Présentation héroku



S. Salva

54

---

---

---

---

---

---

---

---

## Hérouku

- \* 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

S. Salva

55

---

---

---

---

---

---

---

## Hérouku

- \* 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

S. Salva

56

---

---

---

---

---

---

---

## Hérouku, 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

S. Salva

57

---

---

---

---

---

---

---

## Hérou, 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>
</dependencies>
```

S. Salva

58

## Hérou, applications Java

- \* Gestion des dépendances dans pom.xml
- Ajout de postgres:

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

</dependencies>
```

S. Salva

59

## Hérou, 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
-DpackageName=com.example \
-DarchetypeVersion=2.14
```

S. Salva

60

## Hérouku, 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

S. Salva

61

---

---

---

---

---

---

---

---

## Hérouku, interface d'administration

S. Salva

62

---

---

---

---

---

---

---

---

## Hérouku, interface d'administration

S. Salva

63

---

---

---

---

---

---

---

---

21

# Hérouku, interface d'administration

Add-on Categories

Data Stores

Data Store Utilities

Monitoring

Logging

Email/SMS

Caching

Errors and Exceptions

Content Management

Search

Metrics and Analytics

Testing

Messaging and Queuing

Network Services

Alerts and Notifications

User Management

Development Tools

Security

Dynamic

Content

Document Processing

Image Processing

Video Processing

Deployment

Utilities

Payments

ClearDB MySQL

The high-speed database for your MySQL-powered applications.

JavaDB MySQL

The database you trust, with the power and reliability you need.

Compose MongoDB

Hosted, optimized and super fast MongoDB databases.

Instaclustr

6.4

mLab MongoDB

Cloud hosted MongoDB

Citus

Horizontally scalable PostgreSQL

Redis To Go

40 Redis Provider with over 50,000 Redis instances.

openredis

Redis Cloud

Enterprise Class Redis for Developers

Heroku Postgres

Reliable and powerful database as a service based on PostgreSQL.

ObjectRocket for MongoDB

High-Performance MongoDB with Fullstack Support™ and DBAs

Treasure Data

S. Salva

<https://dailymilestones.heroku.com/addons/>

---

---

---

---

---

---

---

---