



# Montez votre site dans le cloud avec Google App Engine

Par Mathieu Nebra (Mateo21)



## Sommaire

Sommaire .....	2
Partager .....	1
Montez votre site dans le cloud avec Google App Engine .....	3
Partie 1 : Débutez avec Google App Engine .....	3
Qu'est-ce que Google App Engine ? .....	4
Comment est né le cloud ? .....	4
Des datacenters et des serveurs par milliers .....	4
Que faire des serveurs inutilisés ? .....	6
Comment fonctionne le cloud ? .....	7
Les différents types de cloud .....	7
Le calcul du coût .....	8
Cloud or no cloud ? .....	9
Google App Engine, le cloud selon Google .....	9
Comment fonctionne App Engine ? .....	10
Quels langages sont supportés par App Engine ? .....	10
Installer Google App Engine avec Eclipse .....	12
Installer Eclipse .....	12
Installer le plugin Google App Engine .....	12
Créer et déployer sa première application .....	16
Création de l'application .....	17
Structure des fichiers d'un projet App Engine .....	18
MaPremiereAppServlet.java .....	19
appengine-web.xml .....	19
web.xml .....	22
Exécuter l'application App Engine .....	23
Déployer l'application App Engine sur les serveurs de Google .....	24
Déclarer l'application auprès de Google .....	25
Déploiement : direction les serveurs de Google ! .....	26
Et maintenant .....	28



# Montez votre site dans le cloud avec Google App Engine

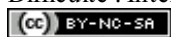
Par



[Mathieu Nebra \(Mateo21\)](#)

Mise à jour : [11/04/2013](#)

Difficulté : Intermédiaire



Depuis un moment, on vous bassine avec le même mot : *cloud*. Vous entendez des choses comme : « Utilise le cloud, et tous tes problèmes seront réglés ! ». Et vous, vous vous demandez ce qui leur prend à parler de *cloud* à tout va... Le pire, c'est que vous ne savez même pas vraiment ce que c'est !

Je vous propose de découvrir le fonctionnement du *cloud* avec Google App Engine. Vous n'avez jamais rêvé d'utiliser les serveurs de Google ? De stocker des informations dans leurs gigantesques bases de données ? D'utiliser des centaines de processeurs pour faire vos calculs les plus gourmands ?

Google App Engine vous permet d'héberger votre site sur les serveurs de Google et de bénéficier d'une puissance qui peut s'adapter au trafic de votre site. Besoin de plus de serveurs ? Il suffit de demander, ça se met en place tout seul ! Et le mieux dans tout ça, c'est qu'on peut s'en servir gratuitement tant que le trafic ne dépasse par 5 millions de pages vues par mois, ce qui laisse de la marge !

Direction les nuages !

## Partie 1 : Débutez avec Google App Engine

### Qu'est-ce que Google App Engine ?

Cloud, cloud, cloud... Les gens n'ont que ce mot-là à la bouche en ce moment ? Qu'est-ce que c'est ? A quoi ça sert ?

Google App Engine est justement une de ces solutions cloud. C'est un service qui vous permet d'utiliser les mécanismes du "cloud computing" et de bénéficier de leurs avantages pour votre prochain site web.

Comme vous devez avoir une bonne compréhension globale des principes du cloud avant toute chose, je vais justement vous expliquer dans ce premier chapitre **ce qu'est le cloud** et comment ça fonctionne. Puis, je vous parlerai plus précisément de **Google App Engine**, le cloud « façon Google », pour que vous puissiez décider si vous ferez votre prochain site avec ces nouvelles technologies. 😊

#### Comment est né le cloud ?

Mettons les choses au clair : « cloud » est d'abord un *terme marketing*. Voilà pourquoi vous l'entendez souvent. On aimerait que vous disiez « je veux utiliser le cloud pour mon site web »... Bien qu'il s'agisse d'un tutoriel sur le cloud (et donc quelque part à la gloire du cloud !), j'aimerais aiguïser votre esprit critique avant d'aller plus loin.

Comment est né le mot *cloud* ? Comment a-t-on découvert et mis en place ces nouvelles techniques ? Laissez-moi vous raconter la petite histoire du cloud à travers l'histoire d'Amazon Web Services, qui a précédé le lancement de Google App Engine.

Remontons quelques années en arrière. Juillet 2002. Amazon est un site de vente en ligne qui cartonne. Simple site de vente de livres à l'origine, on peut aujourd'hui tout acheter : jeux vidéo, matériel hi-fi, vêtements, chaussures... Pour évoluer et gérer de plus en plus de clients, Amazon a dû construire lui-même une très grosse infrastructure technique. Eh oui, il faut des serveurs, beaucoup de serveurs.

#### Des datacenters et des serveurs par milliers

Un site comme Amazon est donc hébergé sur d'innombrables serveurs, eux-mêmes regroupés dans des grands entrepôts appelés datacenters. Tous les très gros sites ont leurs propres datacenters, et Google ne fait évidemment pas exception à la règle.



Un datacenter avec des baies de serveurs

On se croirait dans un décor de science-fiction, et pourtant c'est bien dans des endroits comme celui-ci que tous les plus gros sites web du monde sont hébergés (bon ok la photo a peut-être *légèrement* été retouchée sur Photoshop 🤖).

Google a diffusé une vidéo de l'un de ses datacenters, c'est assez impressionnant et ça mérite vraiment le coup d'oeil. On peut aussi y naviguer avec [Google Street View](#) !



Vidéo



d'un datacenter de Google

Chaque colonne que vous voyez est appelée une baie de serveurs. A l'intérieur, on peut trouver facilement 15, 20, 30 serveurs. Zoomons un peu sur eux :



Zoom sur les serveurs

Les serveurs sont des ordinateurs comme les autres. Ils ont des ports USB, des ports Ethernet, et évidemment un bouton Power.



## Que faire des serveurs inutilisés ?

Pour faire face à la demande qui grandissait de jour en jour, Amazon a dû installer des dizaines de milliers de serveurs dans le monde, eux-mêmes répartis dans de multiples datacenters qui lui appartiennent : aux Etats-Unis, en Irlande, en Asie... Ces datacenters ont poussé comme des champignons ces dernières années.

Tous ces serveurs ne sont pas utilisés en même temps. Certains sont prêts pour faire face à la demande lors des pics de vente (comme Noël). Mais alors, que faire de ces serveurs qui dorment ?

Les ingénieurs d'Amazon ont alors eu l'idée de les louer à d'autres développeurs web. Mais pas comme n'importe quel hébergeur web qui louerait des serveurs dédiés, attention : Amazon s'est dit « on ne va pas louer la machine physique elle-même mais la puissance de nos machines ». Ainsi, les entreprises n'auront plus besoin d'acheter des tonnes de serveurs comme eux juste pour être prêt en cas de pic de trafic : ils devront simplement demander d'utiliser temporairement plus de serveurs les jours où ils ont plus de visiteurs.



Quelle est la différence avec un hébergeur qui loue des serveurs ?



Dans le cas d'un hébergement traditionnel, vous achetez ou louez vos propres serveurs. Vous pouvez dire, si vous allez dans le datacenter : « ces serveurs-là, ce sont les miens ! » (et si vous les avez achetés, vous pouvez même graver votre nom dessus 🤪).

Les ingénieurs d'Amazon ont eu l'idée de « cacher » le fonctionnement de leurs serveurs et de vendre uniquement de la puissance de calcul et du stockage. Bien sûr, au final, c'est toujours un vrai serveur qui répond à vos requêtes (un serveur branché sur une prise électrique avec un disque dur et tout !). Mais la différence est que vous ne savez pas quel est le serveur qui répond aux requêtes. Parfois, le serveur change dans la journée et vous n'êtes pas au courant, mais ça n'est pas grave : ce qui compte, c'est que votre site fonctionne toujours !

L'intérêt de ce fonctionnement, c'est que vous n'avez plus à vous préoccuper des problèmes physiques des machines. Si un disque dur tombe en panne, vous passez automatiquement à un autre serveur pendant que des techniciens vont le réparer et le remplacer.

Voilà comment le cloud (ou cloud computing) est né. Ca n'a rien de magique : il y a toujours des serveurs, des processeurs et des disques durs. Sauf qu'au lieu de vous louer un serveur précis avec son numéro de série, on vous loue « la puissance d'un serveur » et on vous garantit que le service fonctionnera toujours, même en cas de panne matérielle.



Le cloud est généralement matérialisé par un nuage qui dit « vous ne voyez pas ce qu'il y a à l'intérieur mais ne vous inquiétez pas ça fonctionne ». Soyons un peu critiques toutefois : les problèmes, même s'ils sont rares, peuvent toujours survenir. Amazon, Google et Microsoft ont tous les trois connu des ennuis avec leur infrastructure de cloud, qui parfois n'a pas fonctionné pendant plusieurs heures à cause d'un gros bug technique. Heureusement, cela reste rare, mais il faut quand même le savoir !



Encore un disque grillé... j'aurais dû passer au cloud

!

## Comment fonctionne le cloud ?

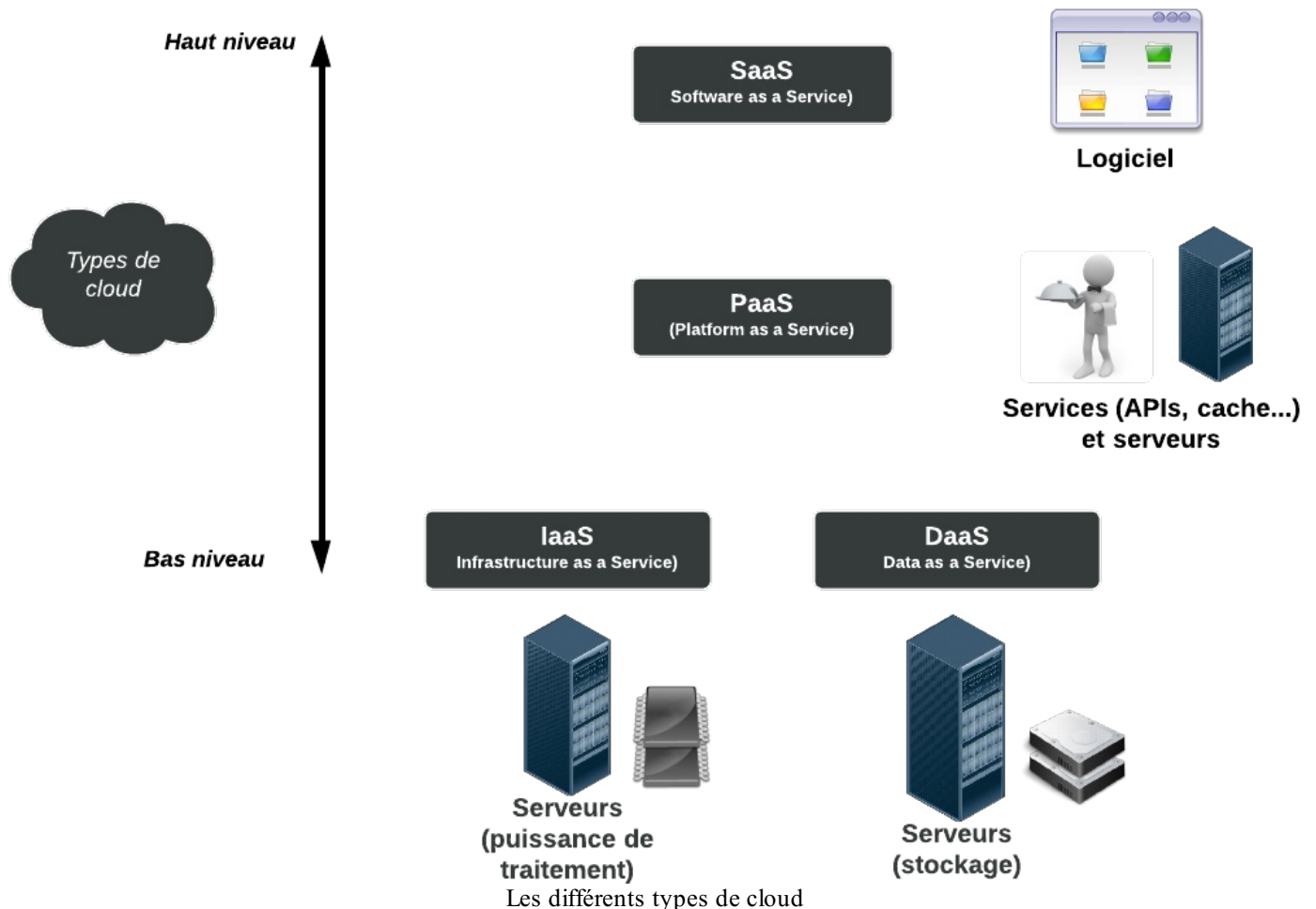
Bon, c'est bien beau tout ça, mais si on rentre un peu dans le vif du sujet ? Par exemple, comment fonctionne le cloud ? Ah... Ca, c'est une question délicate, parce qu'il existe plusieurs types de cloud !

## Les différents types de cloud

Si le cloud est difficile à saisir pour beaucoup de gens, c'est parce que c'est un terme très large qui englobe beaucoup de concepts. Je vous propose pour commencer de retenir les termes suivants, qui font tous partie de la grande famille du cloud :

- **IaaS** (Infrastructure as a Service) : un prestataire vous fournit un accès à tout ou partie de son infrastructure technique, c'est-à-dire à ses serveurs. C'est ce que faisait Amazon à ses débuts dans l'histoire que je viens de vous raconter (ils font d'autres services cloud aussi maintenant).  
C'est un peu comme si vous louiez une chambre d'un hôtel, peu importe laquelle. Si la chambre a un problème technique, comme une fuite d'eau, on vous redirige automatiquement vers une chambre propre qui fonctionne correctement.
- **PaaS** (Platform as a Service) : on vous fournit non seulement un accès à l'infrastructure, mais aussi à des fonctionnalités comme par exemple des bases de données, des serveurs de cache, des serveurs d'e-mail... C'est le cas de Google App Engine : vous bénéficiez des serveurs de Google pour stocker votre site, mais aussi des fonctionnalités très puissantes des bases de données Google, de leurs serveurs permettant d'envoyer des e-mails, etc.  
Pour reprendre mon analogie avec l'hôtel, non seulement vous avez tout le temps accès à une chambre qui fonctionne sans problème, mais en plus vous avez le Room Service qui vous propose la livraison de plats, l'envoi et la réception de messages...

- **SaaS** (Software as a Service) : on vous fournit l'accès à un logiciel sous forme de service. Avant, vous deviez installer le logiciel sur votre machine (ex : Microsoft Office). Aujourd'hui, le logiciel se présente sous la forme d'application web (qui n'est rien d'autre qu'une sorte de super site web !). Vous devez juste vous rendre à une adresse et vous pouvez l'utiliser (ex : Microsoft Office 360, Google Apps...).
- **DaaS** (Data as a Service) : on vous fournit un espace de stockage pour vos fichiers. C'est le cas d'Amazon S3 (un service d'Amazon Web Services) qui vous propose autant d'espace de stockage que nécessaire. C'est un peu comme si vous achetiez un entrepôt de stockage pour vos cartons, dont les murs pourraient s'étendre en fonction de la place dont vous avez besoin !



## Le calcul du coût

Une des grandes innovations du cloud computing, c'est qu'on vous facture uniquement la puissance dont vous avez besoin. Avant, vous deviez prendre et payer 50 serveurs au cas où il y ait un pic de trafic sur votre site. Aujourd'hui, le nombre de serveurs que vous utilisez peut changer d'une minute à l'autre. On vous facture les serveurs à l'heure, et parfois même à la minute ou à la seconde !

Si vous avez un pic de trafic sur votre site, vous paierez un petit peu plus pendant quelques heures mais vous n'aurez pas besoin d'acheter et de monter des serveurs dont vous n'aurez peut-être plus besoin ensuite.

La plupart des services de cloud peuvent être utilisés gratuitement pour commencer. Si vous avez besoin d'un tout petit peu de puissance, c'est donc gratuit !



Comment ?! Mais il faut bien payer à un moment non ?

Oui, mais seulement si votre site grossit et commence à accueillir beaucoup de visiteurs. Dans ce cas, vous aurez besoin de plus de puissance et il faudra commencer à payer.



Chez Google App Engine, vous pouvez par exemple utiliser gratuitement le service avec 1 Go de stockage et l'équivalent de 5 millions de pages vues par mois.

A vous de surveiller votre consommation ensuite car les services de cloud peuvent finir par être coûteux si votre site commence à grossir beaucoup !

## Cloud or no cloud ?

Faut-il partir sur une infrastructure cloud pour votre prochain site ? Il n'y a que vous qui puissiez le décider. Voici quelques éléments vous permettant de faire votre choix en toute objectivité si vous hésitez à vous lancer dans une plateforme PaaS comme Google App engine :

- Le cloud vous permet de commencer gratuitement dans la plupart des cas.
- Vous n'avez (presque) rien à faire si le trafic de votre site grossit : il vous suffit de demander d'utiliser plus de serveurs (et faire chauffer la CB 🍷).
- Vous n'avez pas à gérer les problèmes techniques « bas niveau » comme la perte d'un disque dur ou même d'un serveur entier. Tout cela est transparent pour vous.
- Dans le cas d'un PaaS comme Google App Engine, vous bénéficiez de fonctionnalités très pratiques qui vous évitent d'avoir à installer et maintenir un serveur de base de données, d'e-mails, etc.

En revanche :

- Vous devez adapter votre site pour qu'il fonctionne avec les limitations et fonctionnalités offertes par votre PaaS. Il y a des règles à suivre quand vous développez : vous n'avez en général pas accès au système d'exploitation du serveur et vous ne pouvez pas y stocker des fichiers. Il faut les stocker sur d'autres serveurs en faisant appel à un *service* de stockage : un DaaS (Data as a Service).
- Si vous voulez changer de prestataire, c'est compliqué. Il faut parfois recoder tout ou partie de votre site pour qu'il fonctionne sur un autre PaaS.
- Les bugs techniques sont rares mais peuvent toujours survenir, quoique le service marketing veuille bien tenter de vous faire croire. Ne faites pas une confiance aveugle dans votre cloud et demandez des SLA (Service Level Agreement) pour garantir financièrement le bon fonctionnement du site si vous êtes une entreprise et que le site est critique pour vous. Avec des SLA, le fournisseur aura des pénalités si votre site ne fonctionne pas pendant plusieurs heures par sa faute.

## Google App Engine, le cloud selon Google

Rentrons un peu plus dans le détail du fonctionnement de Google App Engine. Pour commencer, il faut savoir que Google propose plusieurs services cloud :

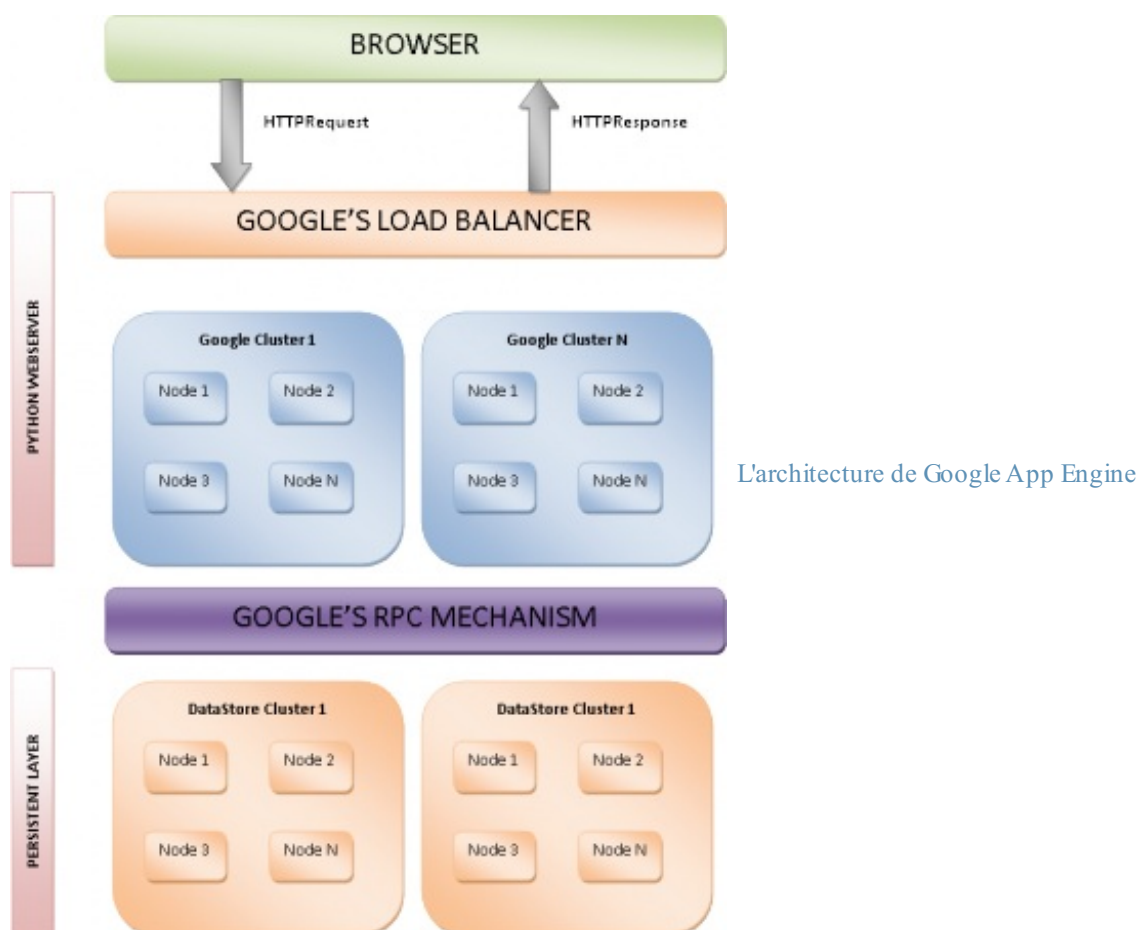
- **App Engine** : le service PaaS de Google, la star de la maison. De gros sites comme Khan Academy ou Pulse l'utilisent. *C'est ce service que nous étudierons dans ce tutoriel.*
- **Compute Engine** : le service IaaS de Google. Si vous avez besoin de puissance de calcul brute (pour calculer des modèles mathématiques par exemple), c'est ce qu'il faut utiliser.
- **Cloud Storage** : un service potentiellement « illimité » de stockage de fichiers dans le cloud. C'est donc un... DaaS. C'est bien, je vois que vous suivez. 🍷
- **Big Query** : des fonctionnalités permettant d'analyser de grosses quantités de données en peu de temps (on parle de Big Data). Très utile pour les entreprises qui veulent faire de la Business Intelligence (BI) pour avoir des indicateurs de tendance basés parfois sur plusieurs Tera Octets de données.
- **Cloud SQL** : une base de données MySQL distribuée dans le cloud. Vous n'avez pas besoin d'installer ni de mettre à jour MySQL. Pas même besoin de le configurer. Vous pouvez créer des serveurs SQL répliqués en quelques clics si vous avez besoin de plus de puissance.



## Comment fonctionne App Engine ?

Le rôle d'App Engine est de "masquer" le fonctionnement et la complexité des serveurs de Google. Lorsqu'un visiteur se connecte à votre site, il arrive sur le *load balancer* (répartiteur de charge) de Google, qui va chercher un serveur disponible et pas trop chargé pour gérer la demande de votre visiteur. Si votre site a besoin d'accéder à des données, ce qui est fréquent, il fera appel à une autre zone de serveurs appelée DataStore (c'est en quelque sorte la base de données).

Tout ceci est résumé dans ce schéma fourni par Google :



Pas besoin de le comprendre et de le retenir par coeur. 🤖

Retenez simplement ce que je viens de vous expliquer, ce sera déjà pas mal !

## Quels langages sont supportés par App Engine ?

Les plateformes PaaS comme Google App Engine supportent un nombre limité de langages de programmation. En effet, elles fournissent des fonctionnalités supplémentaires sous la forme de bibliothèques, il faut donc que Google ait développé les fonctionnalités correspondantes pour ces langages.

Voici les 3 langages supportés par Google à l'heure actuelle (cette liste est susceptible de s'agrandir à l'avenir) :

- Java
- Python
- Go



En réalité, tout langage pouvant utiliser la machine virtuelle Java peut aussi fonctionner : c'est le cas de Ruby et de Scala par exemple. Dans la pratique, la plupart des gens utilisent soit Java, soit Python. Go est un langage créé par Google assez spécifique et encore expérimental, nous n'en parlerons pas ici.

Dans ce cours, je supposerai que vous savez déjà créer des sites web avec un de ces langages. Mes exemples seront basés sur un site Java EE développé en Java. Allez donc lire les cours sur [Java](#) et [Java EE](#) (ou sur [Python](#) et [Django](#)) pour apprendre la création de sites web dynamiques avec ces langages.

Lorsque vous saurez créer des sites en Java ou Python, vous pourrez ensuite découvrir avec moi comment on fait des sites dans le cloud avec Google App Engine !

Allez, vous en savez assez sur le cloud. 😊

Dans le prochain chapitre, nous allons voir comment mettre en place les outils de développement pour créer un site web en Java en utilisant Google App Engine. Nous allons en fait simuler le fonctionnement des serveurs de Google sur notre propre machine pour créer notre site !

## Installer Google App Engine avec Eclipse

Maintenant que nous savons un peu mieux ce qu'est le Cloud et en quoi consiste Google App Engine, si nous passions un peu à la pratique ? Avant de commencer à développer notre premier site utilisant App Engine, il va nous falloir transformer notre ordinateur en "mini-serveur Google" pour pouvoir faire plus facilement nos tests par la suite.

Heureusement, Google nous fournit tous les outils dont nous avons besoin pour développer. Ces outils varient en fonction du langage que vous utilisez : Java, Python ou Go. Comme je vous l'ai dit, je ne peux pas présenter tous les langages à la fois, j'ai donc choisi de faire mes exemples en Java.

*(insérez ici le traditionnel troll sur Java une bonne fois pour toutes, qu'on puisse continuer tranquillement ensuite 🤪)*

La plupart des développeurs Java saints de corps et d'esprit (oui ça existe) utilisent un IDE comme Eclipse. Google y a pensé et nous fournit justement un kit de développement App Engine sous la forme d'un plugin Eclipse !



Pour information, Google fournit aussi le [support de Maven](#) pour les projets App Engine. Cela intéressera ceux qui ont l'habitude de l'utiliser pour générer leurs projets Java EE.

### Installer Eclipse

Si vous avez l'habitude de développer en Java et de faire du Java EE, vous devriez déjà utiliser Eclipse régulièrement. Vous pouvez donc sauter cette étape qui ne vous apprendra rien de nouveau.

Cependant, si vous n'avez pas déjà Eclipse, et puisqu'un rappel ne fait jamais de mal, voici ce qu'il faut faire : il faut télécharger Eclipse (nooon ?). Allez sur la section [téléchargement d'Eclipse](#) et sélectionnez la version d'Eclipse de votre choix.

La version la plus téléchargée est "Eclipse for Java EE developers", c'est celle qui convient le mieux dans notre cas. Si vous utilisez une autre version vous ne devriez pas rencontrer de problème rassurez-vous.

Lors du premier lancement, Eclipse vous demandera de choisir un "workspace", un dossier dans lequel il déposera vos projets de développement Java. Vous pouvez laisser la valeur proposée par défaut et continuer.



Eclipse étant lui-même un programme Java, vous devez avoir installé Java auparavant. Vous aurez besoin du JDK (Java Development Kit) pour l'utiliser. [Téléchargez le JDK](#) sur le site d'Oracle si vous ne l'avez pas déjà.



Vous m'excuserez, je passe volontairement un peu vite sur tout ça, car ce ne sont normalement que des rappels pour vous. Si vous êtes perdus, c'est que vous ne connaissez probablement pas Java et Java EE. Lisez les tutoriels correspondants avant d'utiliser Google App Engine. 😊

### Installer le plugin Google App Engine

Installer le plugin Google App Engine dans Eclipse est vraiment très simple. Allez dans le menu **Help > Install New Software**.

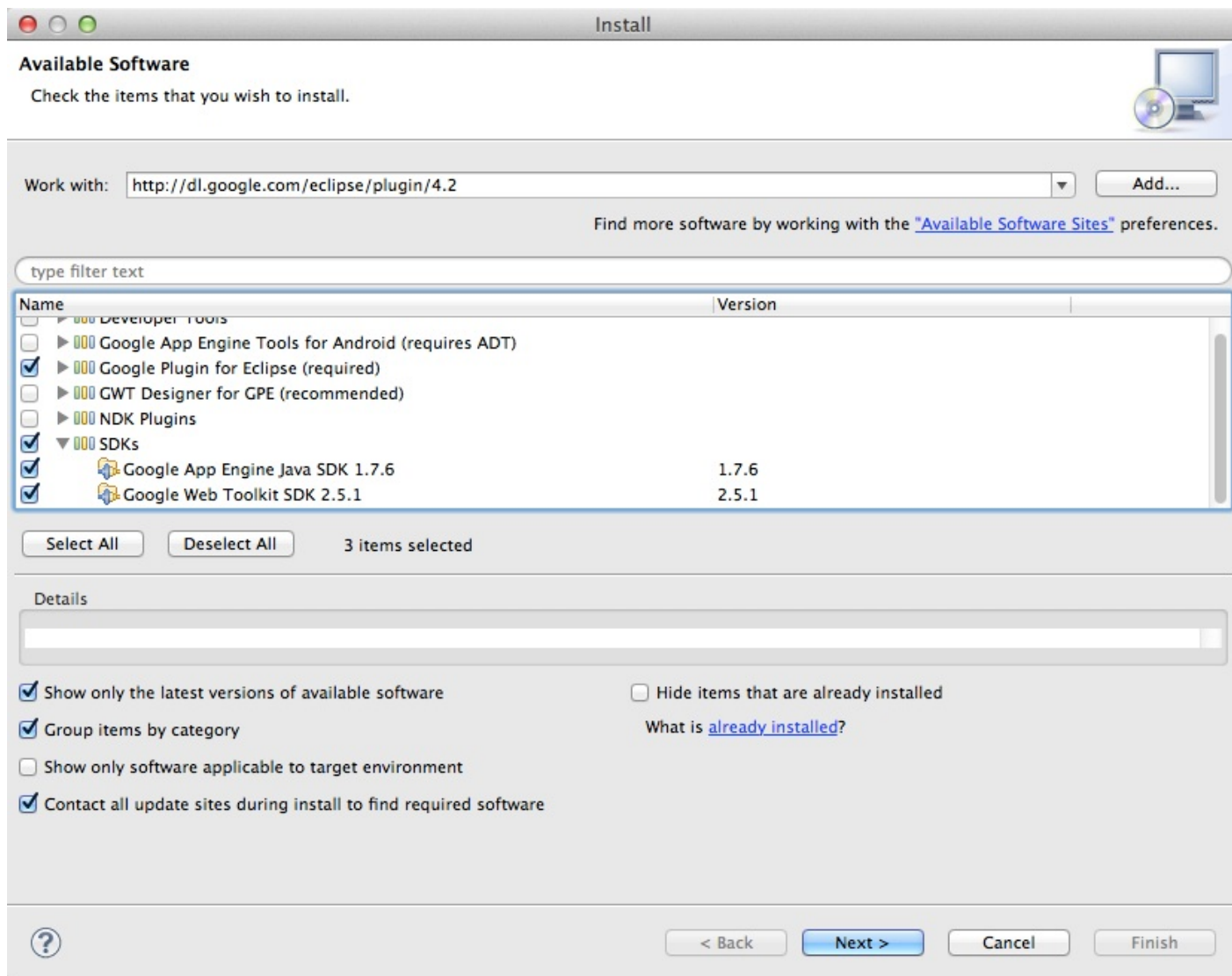
Une fenêtre apparaît, où l'on vous demande où trouver de nouveaux plugins. Rentrez l'adresse suivante (si vous avez la version 4.2 d'Eclipse comme moi, sinon changez l'URL) :

<http://dl.google.com/eclipse/plugin/4.2>

Appuyez sur Entrée. La liste des plugins disponibles se charge. Je vous invite à cocher ceux-ci :

- Google Plugin for Eclipse
- SDK > Google App Engine Java SDK
- SDK > Google Web Toolkit SDK (facultatif)

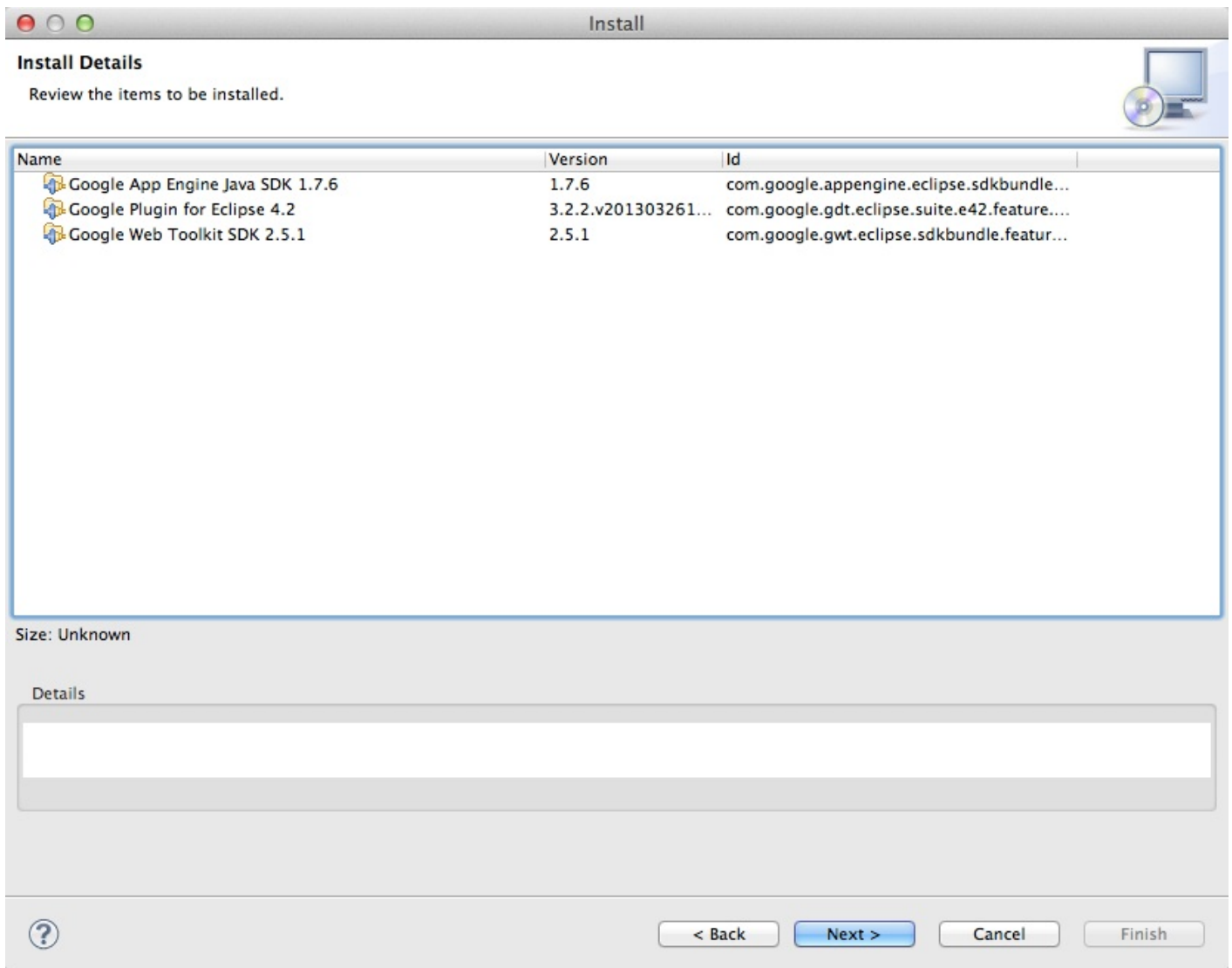
Votre fenêtre devrait alors ressembler à ceci :



Sélection des plugins à installer

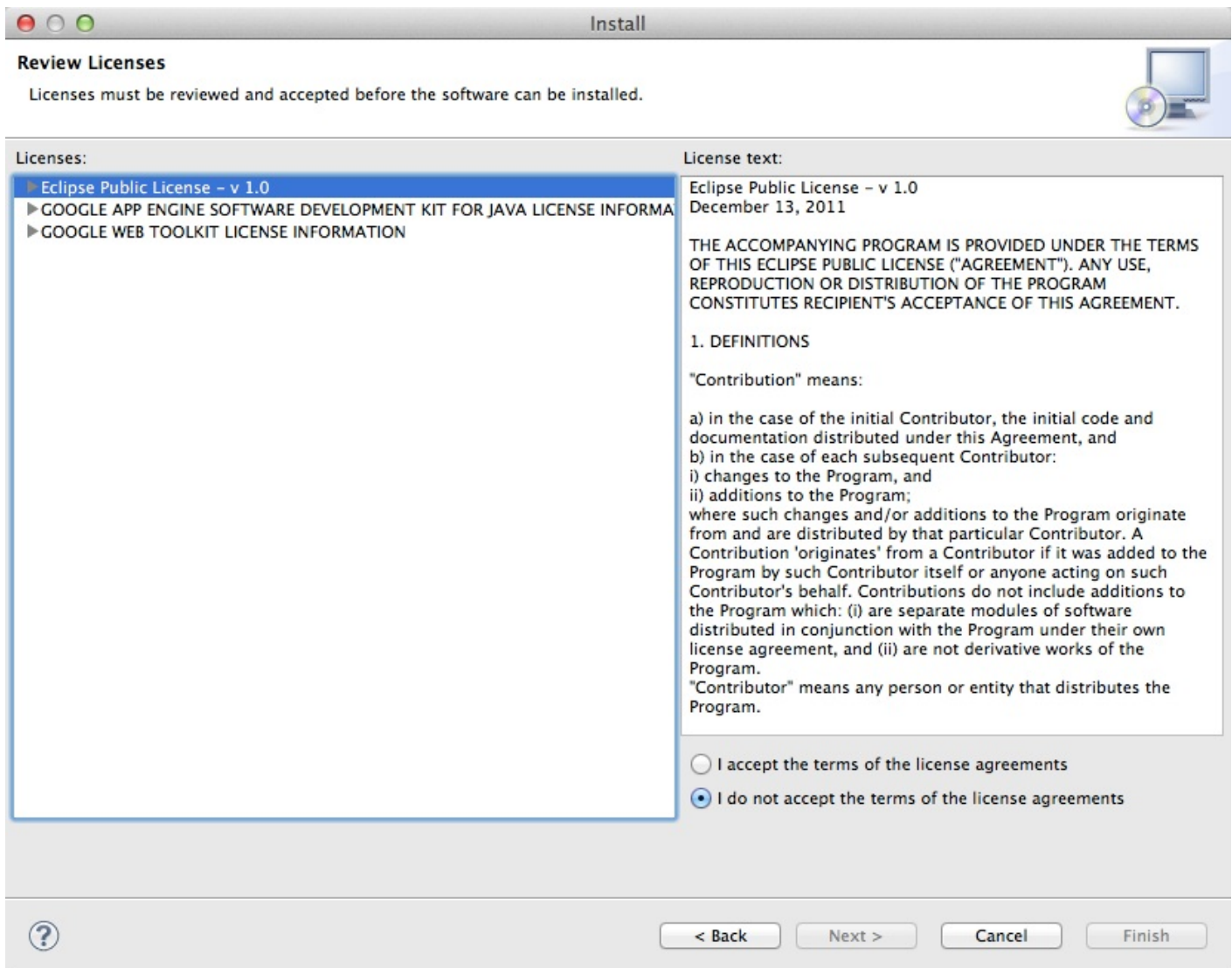
Vous pouvez installer d'autres plugins si vous le voulez mais ça ne nous sera pas nécessaire ici. Cliquez ensuite sur Next.





Validation des éléments à installer

On vous demande de vérifier ce que vous allez installer. Cliquez encore sur Next.



Validation de la licence

Vous devez valider la licence d'utilisation. Cliquez sur "I accept the terms of the license agreements" et continuez. L'installation devrait commencer.

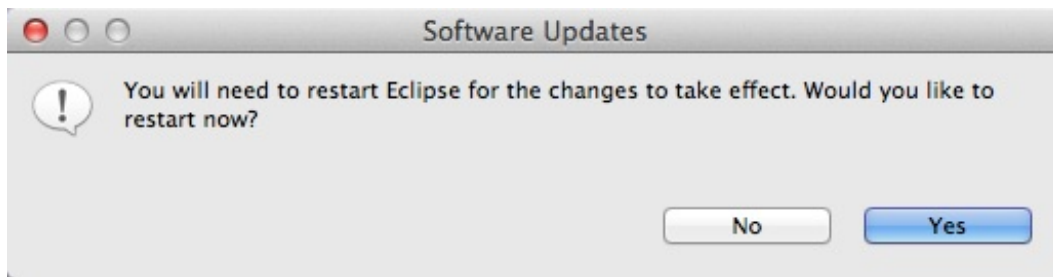
Au bout d'un moment, vous verrez le message suivant :



Validation de la signature

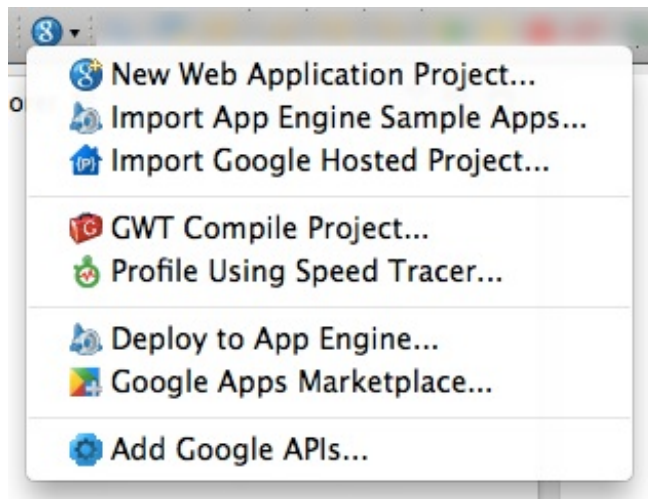
Pas de panique. Eclipse avertit qu'il y a du contenu non signé (qui pourrait ne pas provenir de Google) dans ce que vous téléchargez. En réalité il n'y a pas de problème, cliquez sur OK pour continuer l'installation.

A la fin, on vous demandera de redémarrer Eclipse pour finaliser l'installation :



Redémarrage d'Eclipse

Une fois Eclipse redémarré, c'est fini ! Si vous lancez un nouveau projet ou que vous fermez simplement l'écran de bienvenue, vous devriez voir une icône Google (représentée par un "g") qui vous donne accès aux fonctionnalités de Google App Engine que nous avons installé :



Le menu Google dans Eclipse

Voilà, vous avez installé Google App Engine sur votre ordinateur et vous êtes maintenant prêts à programmer ! 😊

## Créer et déployer sa première application

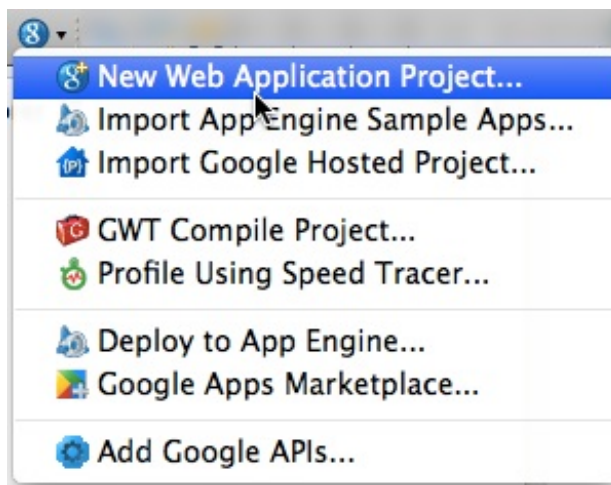
Maintenant que nous avons installé Eclipse et le plugin Google App Engine, tout est prêt pour que nous commençons à créer notre première application App Engine !

Dans ce chapitre, nous allons nous familiariser avec les outils que nous donne le plugin App Engine. Vous allez voir que créer une première application est vraiment très très simple. Nous irons même jusqu'à la déployer (c'est-à-dire la mettre en ligne) sur les serveurs de Google !

Je vous rappelle que vous devez déjà avoir des connaissances sur Java EE pour suivre ce tutoriel (je voudrais bien tout vous réexpliquer mais ce serait vraiment trop long, et puis on a un [super tutoriel sur le Site du Zéro sur Java EE](#) !). Je montrerai rapidement certains concepts, comme les servlets et les JSP, sans trop m'attarder dessus. Je vous montrerai surtout ce qui *change* par rapport à une application Java EE classique.

### Création de l'application

Pour créer une nouvelle application App Engine, utilisez le bouton "g" de Google qui a été ajouté lorsque vous avez installé le plugin. Sélectionnez "New Web Application Project".



Nouvelle application App Engine

Un assistant de création de projet s'ouvre. A l'intérieur :

- Donnez un nom à votre premier projet (par exemple "MaPremiereApplication")
- Donnez un nom au Package que vous allez créer (par exemple "maPremiereApplication")
- Vérifiez que les cases "Use Google App Engine" et "Generate project sample code" sont cochées.
- En revanche, décochez "Use Google Web Toolkit" si vous l'avez installé, il ne nous sera pas utile ici et alourdira inutilement le projet.

**New Web Application Project**

Create a Web Application project in the workspace or in an external location

Project name: MaPremiereApp

Package: (e.g. com.example.myproject) maPremiereApp

Location

☒ Create new project in workspace

☐ Create new project in:

Directory: /Users/mateo21/dev/workspaces/MaPremiereApp [Browse...](#)

Google SDKs

☐ Use Google Web Toolkit

☒ Use default SDK (GWT - 2.5.1) [Configure SDKs...](#)

☐ Use specific SDK: GWT - 2.5.1

☒ Use Google App Engine

☒ Use default SDK (App Engine - 1.7.6) [Configure SDKs...](#)

☐ Use specific SDK: App Engine - 1.7.6

The project will use App Engine's [High Replication Datastore \(HRD\)](#) by default.

Google Apps Marketplace

☐ Add support for listing on Google Apps Marketplace

Sample Code

☒ Generate project sample code

[?](#) [Cancel](#) [Finish](#)

Préparation de l'application

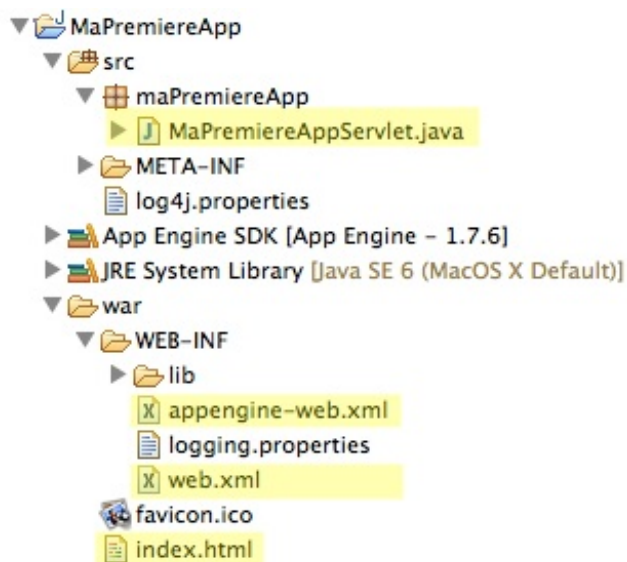
Cliquez sur Finish. Le plugin App Engine se charge de créer et d'organiser les fichiers de votre application tout seul !

Et si on regardait un peu comment ces fichiers sont organisés maintenant ? 😊

### Structure des fichiers d'un projet App Engine

En générant notre premier projet, le plugin App Engine a créé toute une architecture de fichiers et de dossiers. Elle ne devrait pas trop vous dépayser si vous avez l'habitude de développer en Java EE.





Structure des fichiers d'un projet App Engine

J'ai surligné les fichiers qui méritent qu'on y jette un oeil. Je vous propose qu'on en parle un petit peu. 😊

## MaPremiereAppServlet.java

C'est un fichier de code source Java. On l'appelle *servlet* quand on développe des applications web Java EE.

Ce fichier contient une classe qui peut interagir avec le serveur web. Elle peut récupérer les données envoyées par le visiteur depuis un formulaire et les traiter par exemple.

Le code généré par le plugin ressemble à ceci :

### Code : Java

```
package maPremiereApp;

import java.io.IOException;
import javax.servlet.http.*;

@SuppressWarnings("serial")
public class MaPremiereAppServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws IOException {
        resp.setContentType("text/plain");
        resp.getWriter().println("Hello, world");
    }
}
```

Tout se passe dans la méthode `doGet()` qui est appelée quand le visiteur demande la page. Ici, la servlet ne fait rien de particulier : elle affiche "Hello, world" sur la page sous forme de texte brut. Elle pourrait néanmoins faire beaucoup d'autres choses, comme récupérer les données envoyées par un formulaire, demander à les stocker dans une base de données de Google, envoyer des emails à l'aide des serveurs utilisés par GMail et bien d'autres choses. Nous verrons comment faire tout cela dans les prochains chapitres !

Cette servlet ressemble en tous points à une servlet d'application Java EE classique, si ce n'est qu'on peut accéder à tous les services offerts par les bibliothèques d'App Engine.

## appengine-web.xml

C'est un fichier de configuration propre à App Engine. C'est en fait le seul de la liste que je vous présente qui diffère d'un projet

Java EE classique (si on ne tient pas compte des nombreuses bibliothèques d'App Engine qui ont été chargées mais que nous n'avons aucun besoin d'ouvrir).

Eclipse a la fâcheuse tendance à afficher ce fichier dans une interface spécifique, alors que ce n'est en fait qu'un simple fichier XML :

Node	Content	
?? xml	version="1.0" encoding="utf-8"	
▼ [e] appengine-web-app		
[a] xmlns	http://appengine.google.com/ns/1.0	
[e] application		
[e] version	1	
[--]		Eclipse affiche les
[e] threadsafe	true	
[--]	Configure java.util.logging	
▶ [e] system-properties		
[--]		

XML dans une interface spécifique

Pour changer de mode de visualisation, cliquez sur l'onglet "Source" en bas à gauche. Vous devriez alors voir le vrai code XML :

#### Code : XML

```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application></application>
  <version>1</version>

  <!--
Allows App Engine to send multiple requests to one instance in parallel:
-->
  <threadsafe>true</threadsafe>

  <!-- Configure java.util.logging -->
  <system-properties>
    <property name="java.util.logging.config.file" value="WEB-INF/logging.properties"/>
  </system-properties>

  <!--
HTTP Sessions are disabled by default. To enable HTTP sessions specify:

<sessions-enabled>true</sessions-enabled>

It's possible to reduce request latency by configuring your application to
asynchronously write HTTP session data to the datastore:

<async-session-persistence enabled="true" />

With this feature enabled, there is a very small chance your app will see
stale session data. For details, see
http://code.google.com/appengine/docs/java/config/appconfig.html#Enabling_Sessio
-->

</appengine-web-app>
```

Le fichier est assez simple en fait. Google a laissé quelques commentaires pour nous aiguiller si on veut compléter le fichier.



A quoi sert ce fichier ?



Il donne des informations générales sur votre application App Engine : son nom, sa version... Il permet d'activer certaines fonctionnalités comme les sessions, des services de Google, etc.

### *Le nom et la version*

Pour le moment, l'application n'a pas de nom. On devrait d'ailleurs plus exactement parler d'identifiant, car l'application doit avoir un nom unique sur les serveurs de Google, nous verrons cela un peu plus tard.

Le numéro de version permet, lui, d'identifier la version de votre application. Vous pouvez tester une nouvelle version de votre application en ligne pendant que vos visiteurs sont sur une ancienne version. Et lorsque vous trouvez que votre nouvelle version fonctionne bien, il ne vous faudra que quelques clics dans l'interface d'administration de Google pour rediriger vos visiteurs vers la nouvelle version !

Par exemple, si votre site a pour nom "monsupersite", il sera accessible par défaut à cette adresse :

<http://monsupersite.appspot.com>

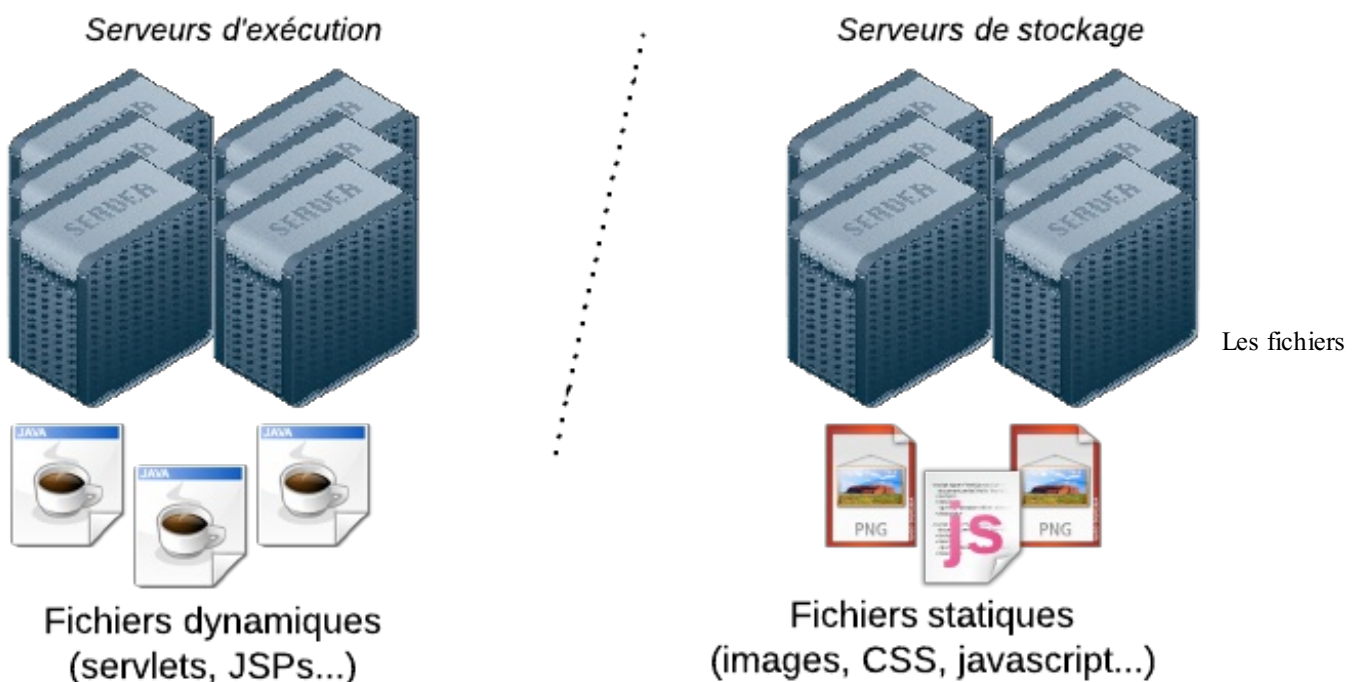
Et si vous voulez tester spécifiquement la version 57 de votre application, il vous suffira d'aller à l'adresse :

<http://57.monsupersite.appspot.com>

### *La déclaration des fichiers statiques*

Le fichier appengine-web.xml permet aussi d'indiquer où se trouvent les fichiers statiques de votre site, c'est-à-dire les images (PNG, JPEG...), les CSS, les Javascript, etc. En gros, tout ce qui n'est pas du code Java.

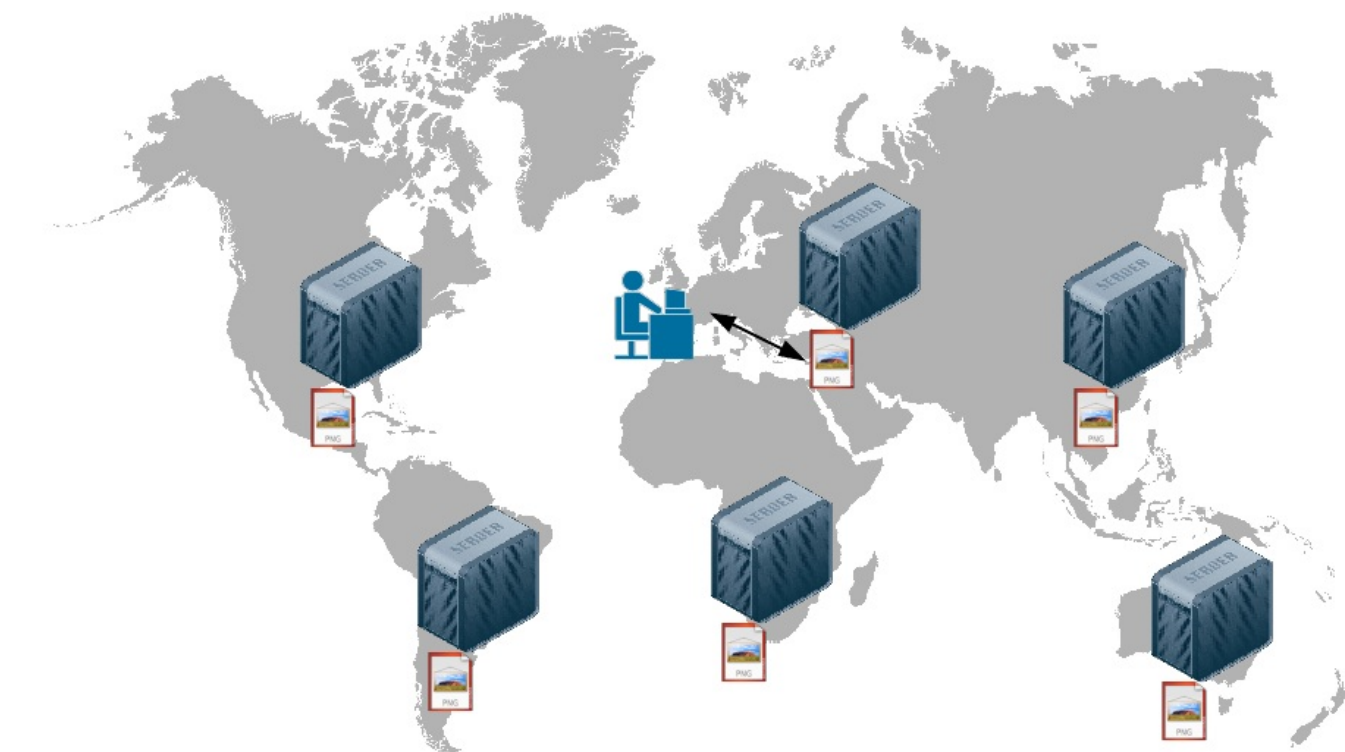
Tous les fichiers de votre site n'ont pas besoin de subir de traitement particulier par les serveurs de Google. Un PNG est un PNG, il n'est (normalement) pas modifié par le serveur avant envoi au visiteur, alors qu'une servlet requiert une exécution par un processeur. On vous demande donc d'indiquer où sont ces fichiers statiques pour que Google puisse les mettre à part sur des serveurs spécifiques qui n'ont pas besoin d'autant de puissance processeur.



statiques sont stockés sur des serveurs différents des servlets et JSPs

L'avantage de mettre les fichiers statiques sur des serveurs spécifiques, c'est que Google peut ainsi les dupliquer sur plusieurs serveurs via un CDN (Content Delivery Network). Ainsi, chaque visiteur ira récupérer les fichiers statiques sur le serveur le plus

proche de lui, ce qui rendra leur téléchargement plus rapide. 😊



Avec un CDN, le visiteur télécharge les fichiers depuis le serveur le plus proche

Pour indiquer à Google quels sont les fichiers statiques, vous devrez ajouter des balises XML incluant ou excluant des fichiers. Par exemple :

Code : XML

```
<static-files>
  <include path="/**/*.png" />
  <include path="/javascript/*.js" />
  <exclude path="/generator/**/*.png" />
</static-files>
```

On a 2 règles d'inclusion et 1 règle d'exclusion dans cet exemple :

- Tous les fichiers .png seront considérés comme statiques et entreposés dans des serveurs de stockage (à une exception près que je vous indiquerai juste après). Les deux étoiles \*\* signifient que Google peut aller chercher les .png dans tous les sous-dossiers.
- Tous les fichiers .js dans le dossier /javascript uniquement seront considérés statiques.
- En revanche, les fichiers .png du dossier "generator" seront exclus des fichiers statiques (ils seront bien placés sur les serveurs d'exécution). C'est utile si, par exemple, le dossier generator contient en réalité du code Java qui génère des PNG et qui les présente sous la forme de fichiers PNG pour le navigateur.

## web.xml

C'est le fichier de configuration "classique" de votre application Java EE. Il ne faut pas le confondre avec appengine-web.xml :

- appengine-web.xml contient la configuration de votre application Java EE App Engine, c'est-à-dire de tout ce qui est spécifique à App Engine dans votre application (comme la définition de fichiers statiques pour que Google sache sur

quels serveurs les placer).

- web.xml contient la configuration générale de votre application Java EE, comme par exemple la liste des servlets et l'adresse correspondant à chacune d'elle sur votre site.

Ce fichier est couramment modifié quand on développe une application Java EE. Il devrait vous être familier. Ici, pour le moment, le fichier contient ceci :

#### Code : XML

```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
  <servlet>
    <servlet-name>MaPremiereApp</servlet-name>
    <servlet-class>maPremiereApp.MaPremiereAppServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MaPremiereApp</servlet-name>
    <url-pattern>/mapremiereapp</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

On voit par exemple que la servlet MaPremiereApp (dont on a vu le code Java tout à l'heure) est appelée quand le visiteur essaie d'accéder à la page /mapremiereapp de votre site.

Le fichier index.html est appelé par défaut à la racine de votre site. C'est donc lui qui sera chargé comme page d'accueil.

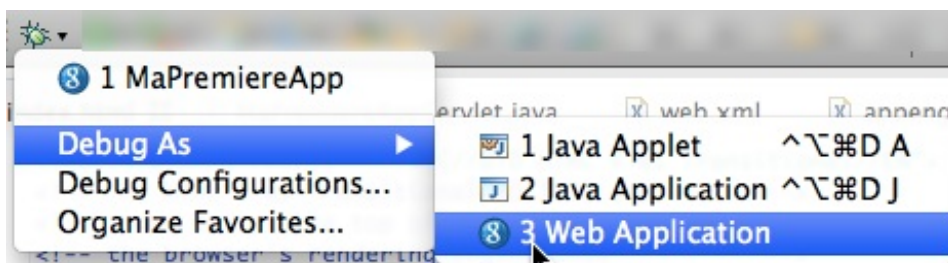
#### index.html

C'est la page qui sera affichée à votre visiteur à l'accueil de votre site. Il s'agit d'une page HTML tout ce qu'il y a de plus basique.

Nous pouvons bien entendu la modifier ou même demander au fichier web.xml d'appeler une servlet à la place de cette page statique pour l'accueil de notre site.

### Exécuter l'application App Engine

Pour tester votre site, il suffit de cliquer sur le bouton "Debug" (ou "Run") de la barre d'outils, et de sélectionner : "Debug as" > "Web Application" :



Exécution de l'application depuis

Eclipse

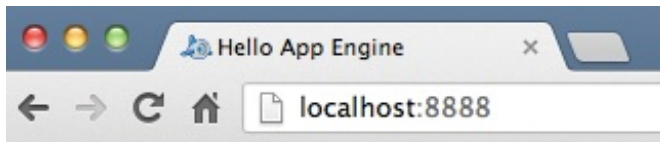
La console en bas d'Eclipse devrait afficher des messages pendant quelques secondes avant d'indiquer : "INFO: Dev App Server is now running". Cela signifie que le mini-serveur Google tourne maintenant sur votre machine et que vous pouvez tester votre site !

Si vous regardez bien, on vous indique à quelle adresse locale vous pouvez tester votre site :

INFO: Server default is running at http://localhost:8888/



Allez donc à l'adresse `http://localhost:8888` dans votre navigateur, vous devriez voir la page d'accueil s'afficher :

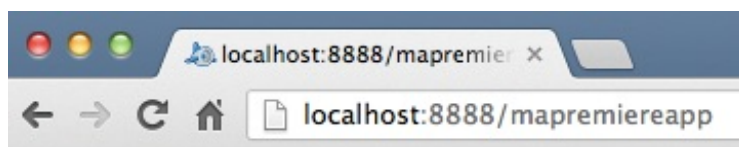


Ouverture du site dans le navigateur (sur localhost:8888)

#### Available Servlets:

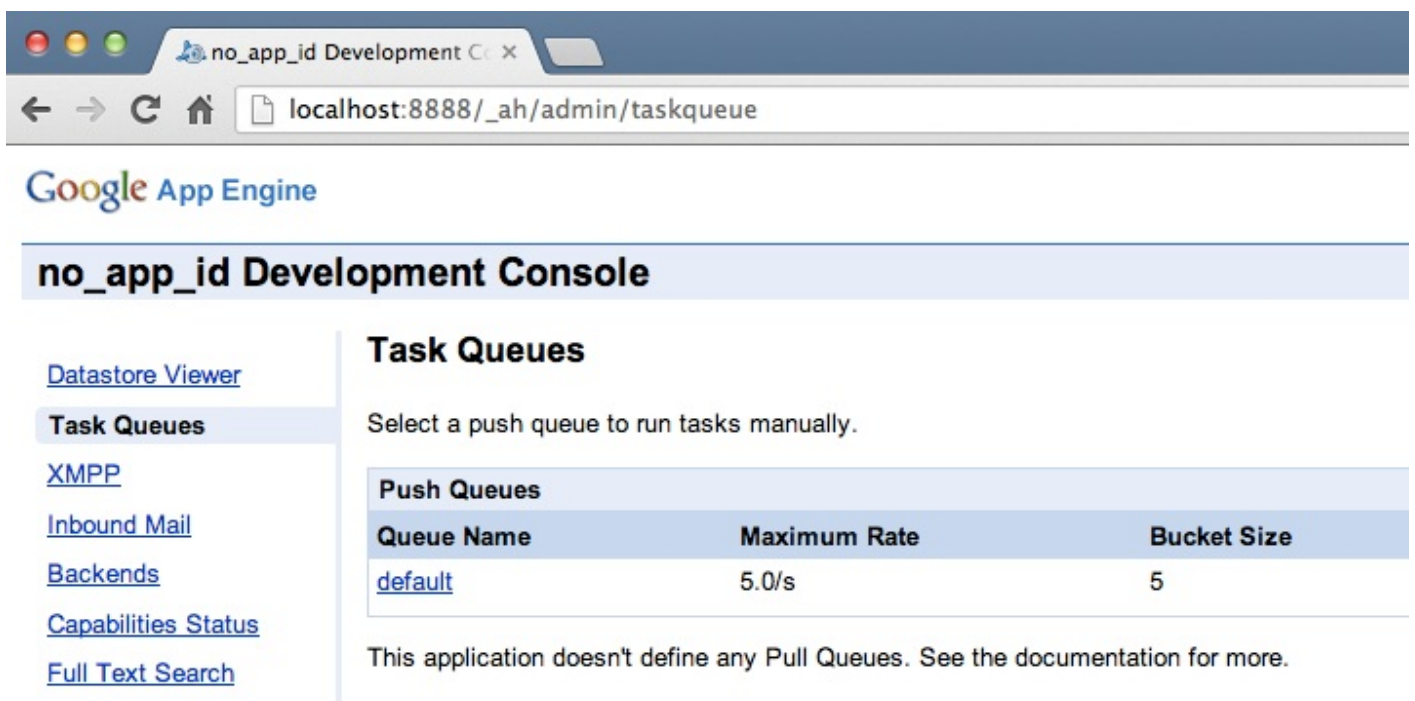
[MaPremiereApp](#)

Il s'agit là du fichier `index.html` que nous avons vu tout à l'heure dans le projet sous Eclipse. Si vous cliquez sur le lien "MaPremiereApp", la servlet `MaPremiereApp` sera chargée et s'exécutera. Comme je vous l'ai dit, cette servlet ne fait rien de bien excitant pour le moment : elle affiche juste "Hello, world" !



Affichage de la servlet dans le navigateur

Il faut aussi savoir que Google génère automatiquement une administration pour contrôler et surveiller votre site. Vous pouvez y accéder en allant à l'adresse `http://localhost:8888/_ah/admin` :

A screenshot of the Google App Engine Administration Console. The browser title is 'no\_app\_id Development Console'. The address bar shows 'localhost:8888/\_ah/admin/taskqueue'. The page has a sidebar with links: 'Datastore Viewer', 'Task Queues' (selected), 'XMPP', 'Inbound Mail', 'Backends', 'Capabilities Status', and 'Full Text Search'. The main content area is titled 'Task Queues' and says 'Select a push queue to run tasks manually.' Below this is a table for 'Push Queues' with columns 'Queue Name', 'Maximum Rate', and 'Bucket Size'. The table has one row: 'default' with a rate of '5.0/s' and a bucket size of '5'. At the bottom, it says 'This application doesn't define any Pull Queues. See the documentation for more.'

L'administration automatiquement générée par Google pour gérer votre application

D'ici, vous pourrez surveiller votre application : voir les données stockées dans le Datastore (la "base de données Google", que nous étudierons plus tard), simuler des emails entrants, désactiver des fonctionnalités pour voir le comportement de l'application en mode dégradé, etc.

Bon, nous avons exécuté l'application qui a été générée par Google, ce n'était pas bien difficile. Je vous propose de la déployer maintenant sur les serveurs de Google. 😊

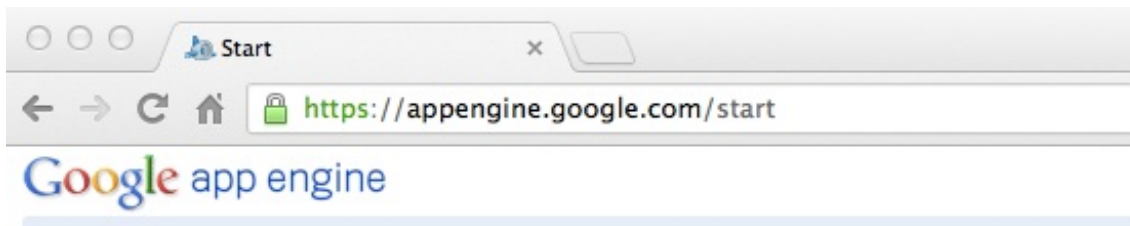
## Déployer l'application App Engine sur les serveurs de Google

Pour pouvoir envoyer notre application App Engine sur les serveurs de Google, nous devons d'abord réserver un identifiant d'application. Nous pourrons ensuite dans un second temps déployer l'application directement depuis Eclipse !

## Déclarer l'application auprès de Google

Commencez par vous rendre à l'adresse suivante : <https://appengine.google.com/>  
C'est ici que vous pourrez créer et gérer vos application App Engine.

Si vous n'avez pas déjà de compte Google (un compte GMail par exemple) il vous faudra en créer un et vous connecter. Une fois que c'est fait, vous devriez voir l'écran suivant :



L'interface

### Welcome to Google App Engine

Before getting started, you want to learn more about developing and deploying applications. Learn more about Google App Engine by reading the [Getting Started Guide](#), the [FAQ](#), or the

Create Application

d'administration de vos projets App Engine chez Google

Cliquez sur "Create Application" pour commencer la déclaration de votre application chez Google. La première fois, on vous demandera sûrement une authentification supplémentaire, pour "éviter les abus" vous dit-on. Il suffit d'indiquer votre numéro de téléphone portable, d'attendre de recevoir un SMS contenant un code secret et de l'écrire sur la page.

Une fois que c'est fait, un simple formulaire vous demande de créer l'identifiant de votre application et de lui donner un nom :



## Create an Application

You have 10 applications remaining.

### Application Identifier:

.appspot.com

Check Availability

Yes, "sdz-mapremiereapp" is available!

All Google account names and certain offensive or trademarked names may not be used as Application Identifiers.

You can map this application to your own domain later. [Learn more](#)

### Application Title:

Displayed when users access your application.

Création de l'application

Vous devrez utiliser un identifiant d'application libre. Cliquez sur "Check Availability" pour vérifier si celui que vous voulez prendre est disponible.

Le formulaire vous demande si vous voulez restreindre les utilisateurs qui se connecteront sur votre site à des comptes Google spécifiques à un domaine (par exemple le domaine d'une entreprise si vous faites une application pour votre entreprise). N'y touchez pas, ça ne nous sert à rien ici.

Validez la licence... et voilà !

On vous propose un lien vers le dashboard (tableau de bord) de votre application. D'ici, vous pouvez voir tout ce qui se passe sur votre site et contrôler toute votre application !

**Application:** sdz-mapremiereapp [High Replication] [Community Support](#) « [My Applications](#)

**Main**

- Dashboard**
- [Instances](#)
- [Logs](#)
- [Versions](#)
- [Backends](#)
- [Cron Jobs](#)
- [Task Queues](#)
- [Quota Details](#)

**Data**

- [Datastore Indexes](#)
- [Datastore Viewer](#)
- [Datastore Statistics](#)
- [Blob Viewer](#)
- [Prospective Search](#)
- [Text Search](#)
- [Datastore Admin](#)
- [Memcache Viewer](#)

**Charts** ?

No Data Available

**Instances** ?

Number of Instances - <a href="#">Details</a>	Average QPS	Average Latency	Average Memory
<b>Billing Status:</b> Free - <a href="#">Settings</a> <span style="float: right;">Quotas reset every 24 hours. Next reset: 16 hrs ?</span>			
Resource	Usage		

Administration de l'application



Quelle est la différence entre cette zone admin et celle que nous avons vue un peu plus tôt après avoir testé le site en local ?

L'admin du site local permet de faire des tests précis pour déboguer temporairement votre application. En revanche, l'admin que vous avez maintenant sous les yeux est beaucoup plus complète : elle vous donne des indications sur l'application qui est en ligne et disponible à tous. Vous pouvez voir le trafic, mettre en maintenance votre site, augmenter la puissance des serveurs, et bien d'autres choses ! 😊

## Déploiement : direction les serveurs de Google !

Maintenant que Google nous connaît et que nous avons réservé un identifiant, nous pouvons retourner dans Eclipse et envoyer notre site !

Il faut d'abord indiquer l'identifiant de l'application dans le fichier appengine-web.xml dans la balise `<application></application>` :

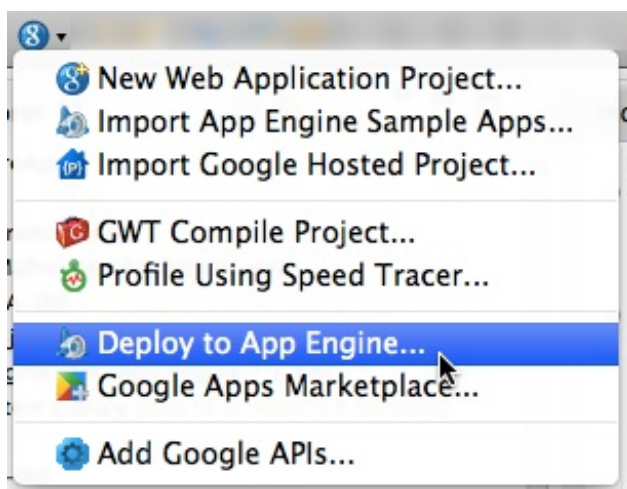
**Code : XML**

```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application>sdz-mapremiereapp</application>
  <version>1</version>
```



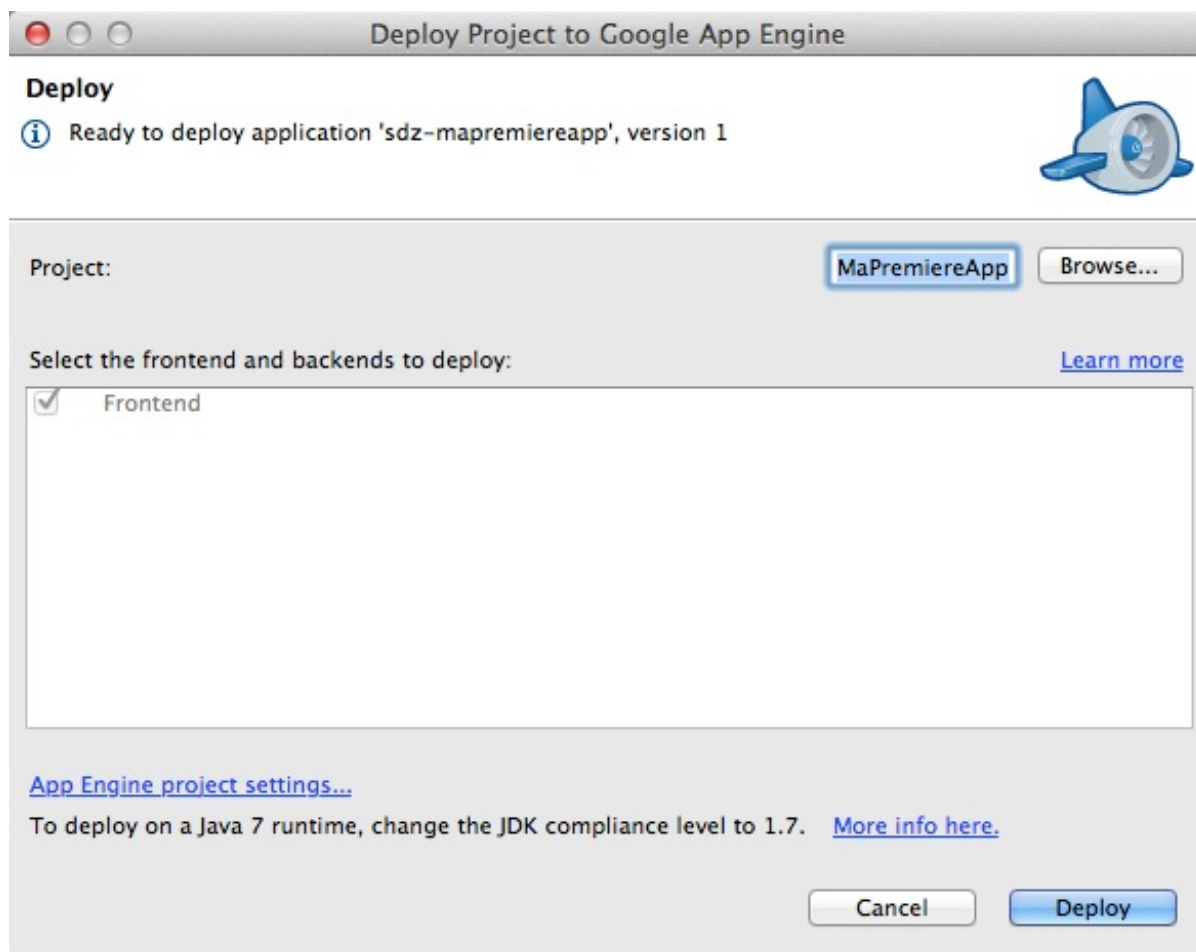
Ne copiez pas le même nom d'application que moi ! Je l'ai réservé en rédigeant ce tutoriel, vous ne pouvez donc pas avoir créé une application ayant le même nom que moi. Indiquez le nom que vous avez réservé à la place.

Vous pouvez maintenant déployer ! Allez dans le menu Google et cliquez sur "Deploy to App Engine" :



Demandez le déploiement depuis le menu Google

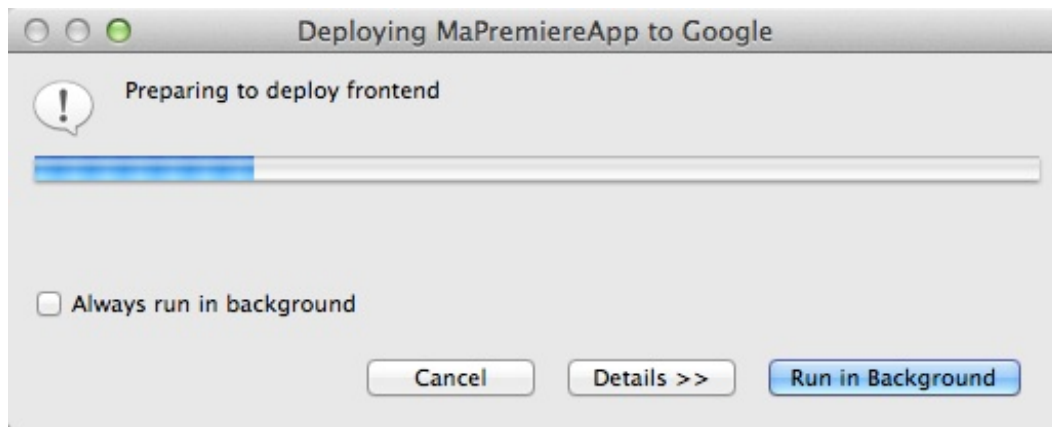
La première fois, on vous demandera de vous connecter à votre compte Google pour qu'on puisse vous identifier. Rentrez votre login et votre mot de passe. Une fois connecté, on vous demande de valider avant de déployer :



Validation avant

déploiement

Cliquez tout simplement sur "Deploy" : votre site part alors sur les serveurs de Google comme un grand !



Le déploiement est en cours,

vous n'avez rien à faire !



Le déploiement d'une application App Engine est une sorte de FTP amélioré. Vous n'avez pas besoin d'indiquer les fichiers qui ont changé, Google se charge de tout !

Voilà, votre site est maintenant disponible à une adresse de la forme : <http://applicationid.appspot.com> (remplacez *applicationid* par l'identifiant de votre application).

Dans mon cas, l'application est disponible à cette adresse :

<http://sdz-mapremiereapp.appspot.com/>

Allez-y, je l'ai envoyée au moment où j'ai écrit ce tutoriel et elle tourne toujours à l'heure où vous lisez ces lignes ! Google se charge de gérer les serveurs et de faire en sorte que les sites web soient toujours en ligne, c'est l'intérêt justement d'utiliser une solution de cloud PaaS (Platform as a Service) comme Google App Engine. 😊



Mais... Mon site est dans un sous-domaine de [appspot.com](http://appspot.com), je n'aime pas l'adresse que ça lui donne ! Il n'y a pas moyen de changer ça ? 😞

Bien sûr que si ! 😊

Vous pouvez tout à fait faire en sorte que votre site soit disponible depuis [www.monsupersite.com](http://www.monsupersite.com) (si vous avez acheté le nom de domaine), pour ne pas que vos visiteurs voient un "appspot.com" qui n'a rien à faire là pour eux.

Il faut aller dans l'administration de l'application, section "Administration" > "Application Settings" et cliquer sur le bouton "Add Domain" dans "Domain Setup". Il faudra associer au préalable le domaine avec Google Apps et modifier les entrées DNS auprès du registrar chez qui vous avez acheté le nom de domaine. Toutes les instructions sont données dans la [documentation de Google](#). Allez, ne faites pas cette tête, un peu de lecture de documentation ne peut pas vous faire de mal !

## Et maintenant

Bravo, vous avez réussi à déployer votre première application Google App Engine !

Dans la suite de ce cours, nous allons découvrir dans le détail toutes les fonctionnalités offertes par Google App Engine : comment stocker des données dans le Datastore, comment authentifier les utilisateurs depuis leur compte Google, envoyer des e-mails, gérer le cache et bien d'autres choses !