Android et le Cloud Computing

Plan de l'exposé

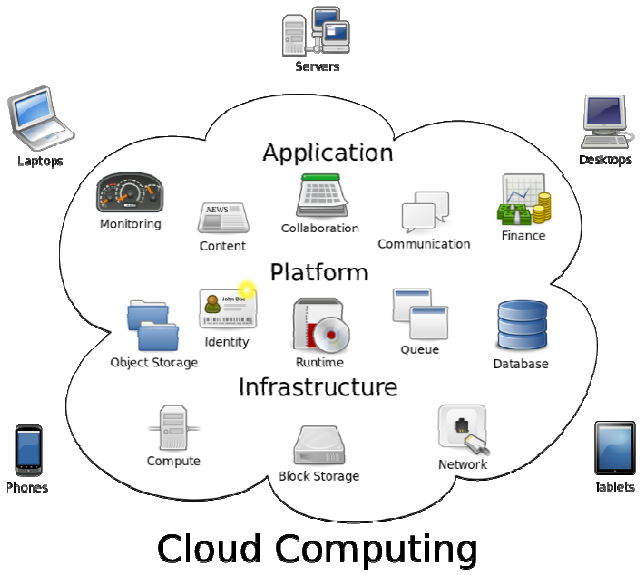
- Rappels sur Android et le cloud
- Présentation de GCM
- Notions fondamentales de GCM
- Construire un projet GCM
- Ecrire un client GCM (récepteur GCM)
- Ecrire un serveur GCM (émetteur GCM)

Cloud computing : rappels

- Android est (aussi) un environnement d'execution
- Un logiciel Android = une application Android
- Une application Android est constituée de composants
- => Ces composants sont pris en charge par l'environnement d'exécution
- Les applications Android sont souvent informées par réception de messages (envoyés par l'environnement d'exécution) : les `Intent`
- Une application Android est décrite par son `AndroidManifest.xml`

Cloud computing : rappels

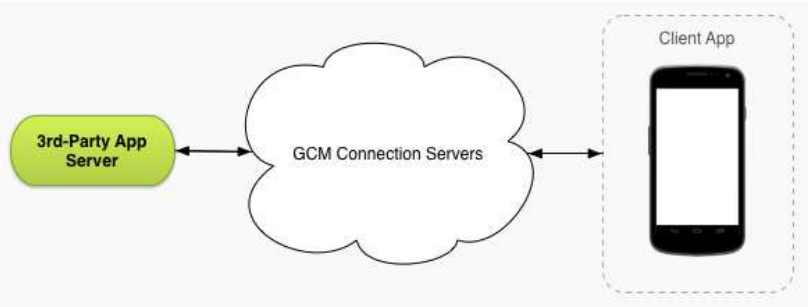
- "Cloud computing is internet-based computing in which large groups of remote servers are networked to allow sharing of data-processing tasks, centralized data storage, and online access to computer services or resources"
- C'est clair, non ?



- source : `http://en.wikipedia.org/wiki/Cloud_computing`

Google Cloud Messaging (GCM)

- Permet d'échanger des messages entre des servers applicatifs et des et des smartphones en utilisant le cloud Google
- L'ancienne version (avant le 26 juin 2012) était C2DM (Cloud to Device Messaging)
- De plus le mode est asynchrone (cf. le mail) : le smartphone n'a pas besoin d'être allumé pour recevoir des messages. Il les recevra lors de son prochain démarrage
- Nécessite Android 2.2 au moins
- GCM est la partie intermédiaire (middleware ?) entre un (des) smartphone(s) et un serveur
- ~ architecture JMS (Java Messaging Service)
- est gratuit et sans quota



~ Architecture JMS

- Design pattern "publish-subscribe"
- Asynchrone : pas besoin d'être à l'écoute pour recevoir (plus tard, après le redémarrage du mobile) les messages
- => L'architecture GCM stocke le message à envoyer au mobile tant que le mobile ne l'a pas reçu (mais délai de garde par défaut 4 semaines)
- "If the device is not connected to GCM, the message will be stored until a connection is established .... When a connection is established, GCM will deliver all pending messages to the device, ... If the device never gets connected again ..., the message will eventually time out and be discarded from GCM storage. The default timeout is 4 weeks..."

- source : `http://developer.android.com/google/gcm/adv.html`

Scénario d'envoi d'un message

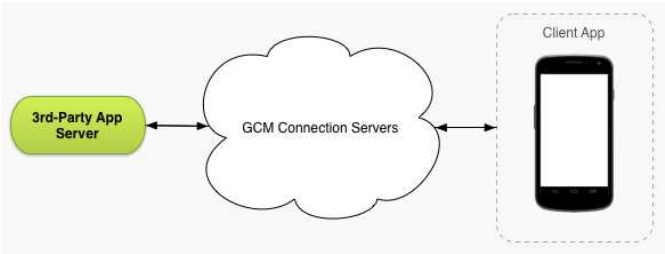
- "The application server sends a message to GCM servers.
- Google enqueues and stores the message in case the device is offline.
- When the device is online, Google sends the message to the device.

- On the device, the system broadcasts the message to the specified Android application via Intent broadcast with proper permissions, so that only the targeted Android application gets the message. This wakes the Android application up. The Android application does not need to be running beforehand to receive the message.
- The Android application processes the message."

- C'est clair, non ?
- source : `http://developer.android.com/google/gcm/server.html#send-msg`

Architecture GCM

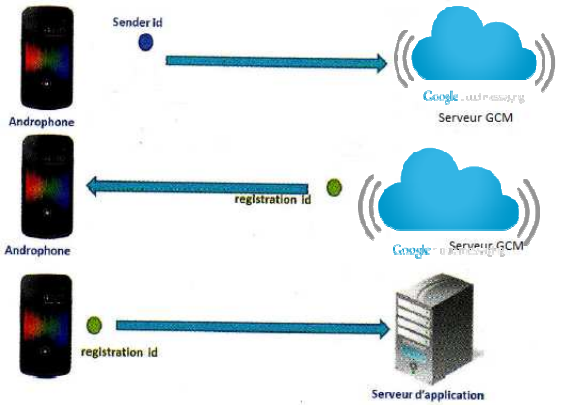
- Un smartphone est informé d'arrivées de messages par `Intent` (comme d'hab)
- Peut fonctionner sur des émulateurs
- Pas besoin de déposer l'application dans Google Play Store
- Utilise les Google Services
- Utilise les notions de composants (Components) et d'identifiants (Credentials)
- Les composants sont les 3 entités :
  - Une application serveur (3rd-party Application Server)
  - Les smartphones (Client App)
  - GCM c'est à dire des serveurs GCM



- source : `http://developer.android.com/google/gcm/gcm.html`

Architecture GCM : les identifiants

- Le Sender Auth Token : l'API Key donné à l'application serveur l'autorisant à utiliser l'architecture GCM. Est obtenu à partir de la "Google Developers Console"
- Le Sender ID : le project number du projet GCM. Est obtenu à partir de la "Google Developers Console"
- Le Registration ID : l'identifiant donné par les serveurs GCM caractérisant le couple (mobile, application cliente) : " In other words, a registration ID is tied to a particular Android application running on a particular device."



- source : `http://developer.android.com/google/gcm/gcm.html`

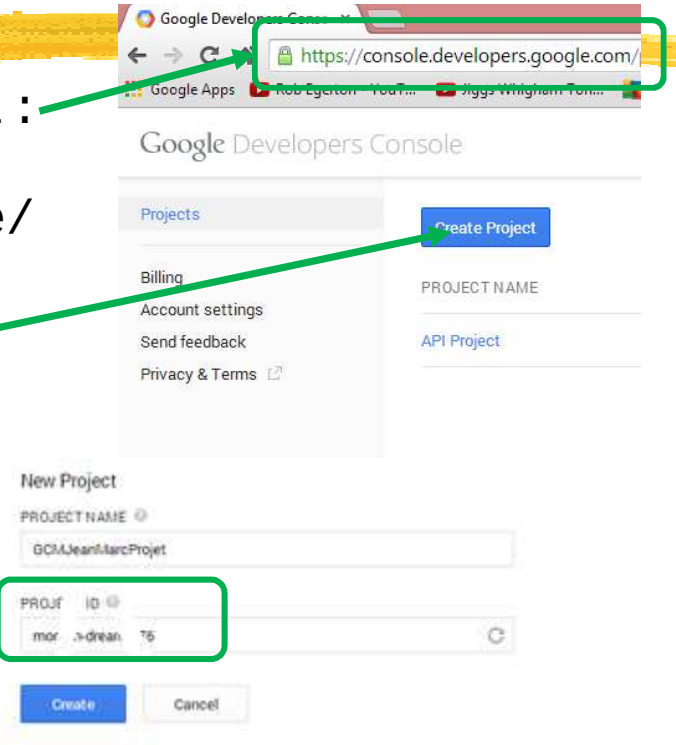
Une application utilisant GCM : les étapes

- D'abord construire un projet GCM = le middleware GCM qu'on va utiliser
- 1°) On le fait dans la "Google Developers Console"

- 2°) Construire une application serveur (émetteur) utilisant GCM
- 3°) Construire une application Android cliente (réceptrice)

Créer un projet GCM Google dans l'API Google (1/4)

- Aller dans la "Google Developers Console" d'URL :
  Euh c'est parfois
  `https://code.google.com/apis/console/`
  mais tapper "Google Developers Console" dans Google c'est bien aussi ;-)
- Si vous n'avez pas de projet déjà crée, il faut en créer un
  Le nom peut être quelconque et un project Name est proposé



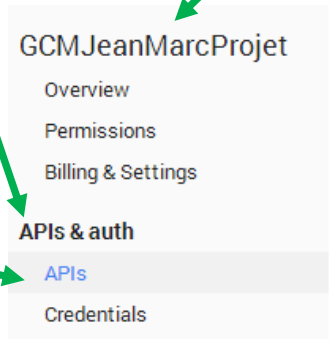
- A l'aide de ce project name on va avoir un project ID (un entier long). Ce project ID sera le GCM Sender ID. Euh parfois les deux sont confondus et c'est bien dommage !
- source :
  `http://developer.android.com/google/gcm/gs.html#next`

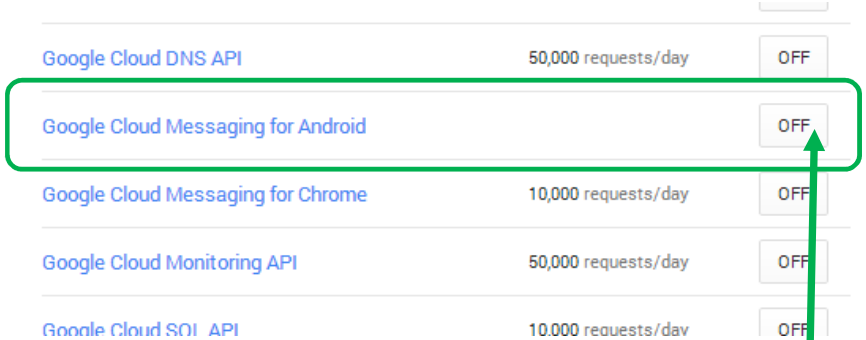
Créer un projet GCM Google dans l'API Google (2/4)

- Il faut ensuite demander d'utiliser le GCM Service pour ce projet. Pour cela sélectionner le projet

Cliquer APIs & auth

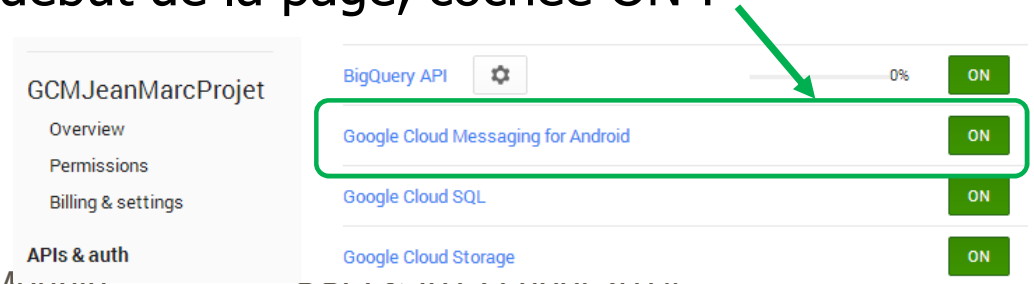
puis APIs





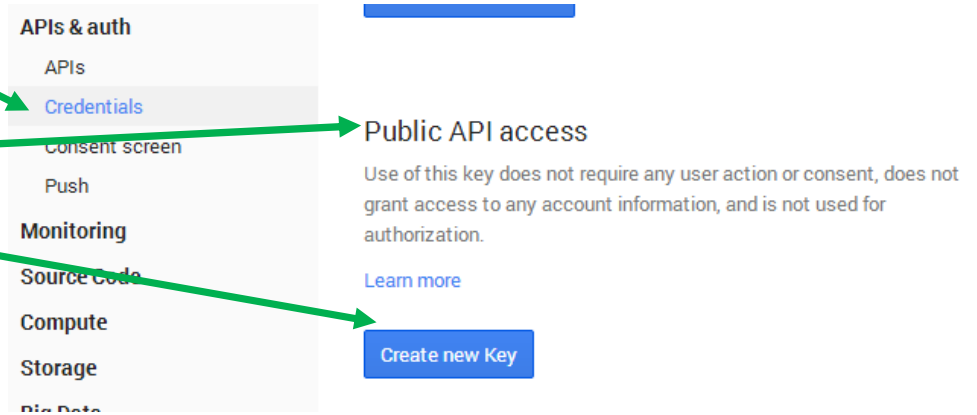
puis cliquer le bouton (Off pour l'instant) de Google Cloud Messaging for Android

- La ligne disparait des propositions (bravo l'ergonomie !) mais apparaît au début de la page, cochée ON !

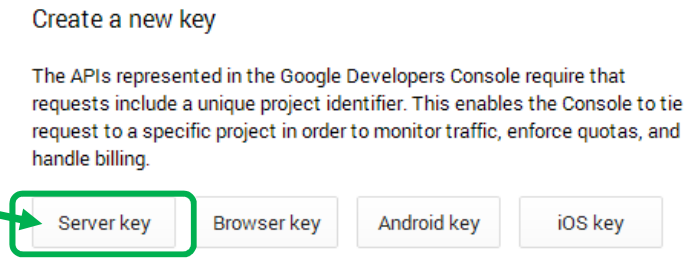


Créer un projet GCM Google dans l'API Google (3/4)

- Sélectionner Credentials puis, dans la partie Public API Access, cliquez Create new Key

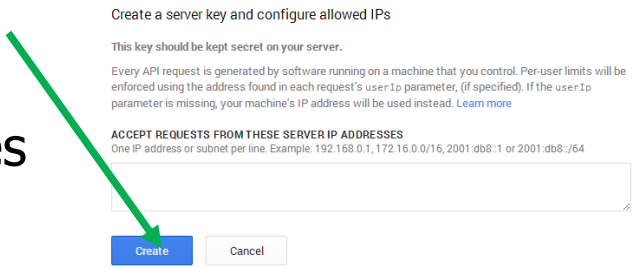


- Dans la boite de dialogue qui apparaît, cliquer Server key

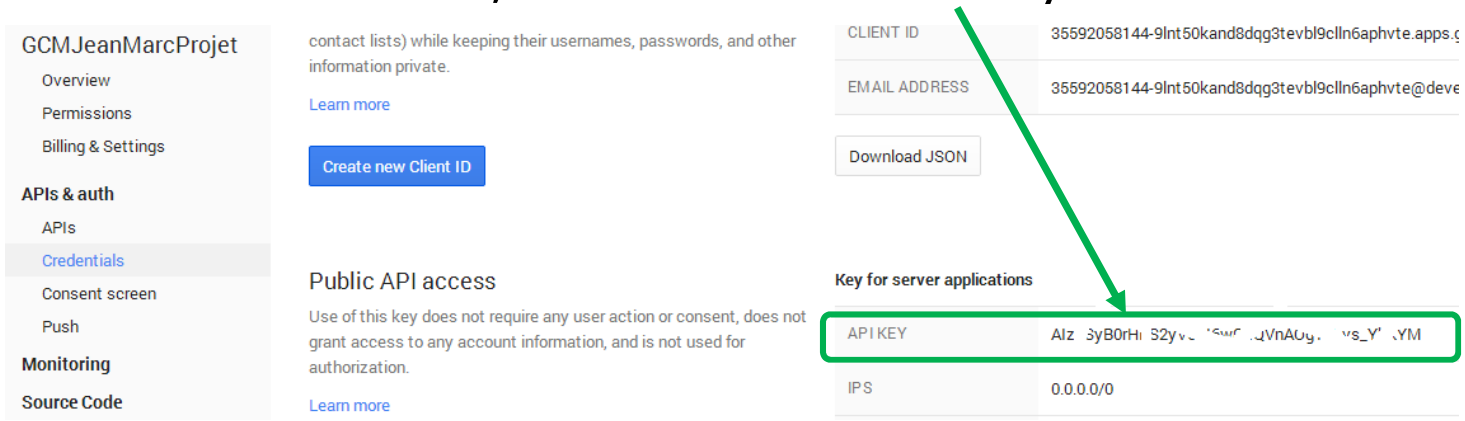


Créer un projet GCM Google dans l'API Google (4/4)

- Dans la boite de dialogue suivante, indiquer l'adresse IP du serveur applicatif qui utilisera le service GCM. Cliquer Create
- On peut mettre `0.0.0.0/0` pour des tests

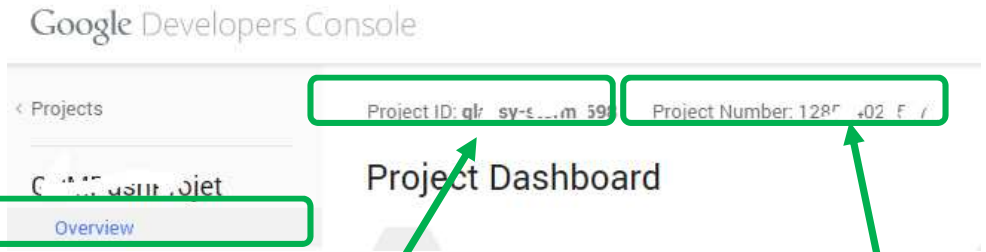


- Dans la fenêtre suivante, on obtient une API Key :



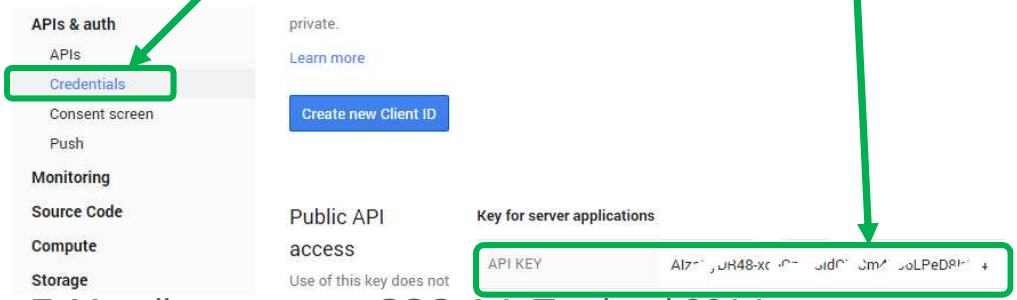
Le projet GCM : un résumé

- Par la suite, on peut toujours récupérer les informations d'un projet en allant à la Google Developers Console
- Aller à l'URL `https://code.google.com/apis/console/` (la Google Developers console)
- Sélectionner le projet
  Eventuellement sélectionner Overview



  Il apparaît en haut de la page, le project ID et le Project Number

- Cliquer sur Credentials. La page affiche l'API KEY



Choix de la connexion

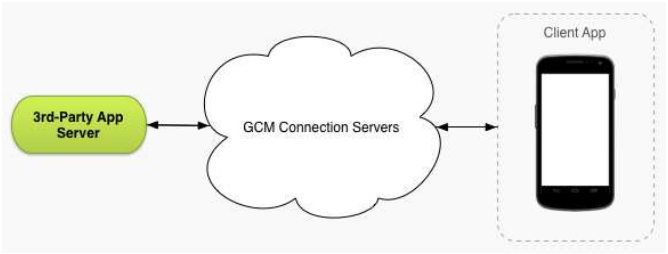
- On peut utiliser comme couche transport HTTP ou XMPP
- Si on choisit HTTP, on ne pourra avoir que des connexions serveur applicatif -> Cloud -> smartphone
- Avec HTTP les envois serveur applicatif -> cloud (= serveurs cloud) sont bloquants : envois synchrones

- Avec XMPP, on peut avoir les deux sens
- La connexion XMPP est appelée aussi CCS (Cloud Connection Server) voir à `http://developer.android.com/google/gcm/ccs.html`
- On peut utiliser ces deux types de connexion en même temps
- envois asynchrones avec CCS
- source : `http://developer.android.com/google/gcm/server.html`

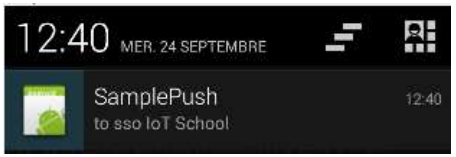
Une démo

- dans workspace D:\CNAM\DevMobiles\Android\JMF\SupportCours\GCM\TravailPushGCM
- client projet `ReceiveFromCloudInAndroidDeviceProject`, application Android
- serveur projet `PushIntoTheCloudProject`, application Java SE `EmetteurDansLeCloud.java`

Java SE application



Android application



- Tout cela inspiré du tutorial à `http://android.amolgupta.in/2012/07/google-cloud-messaging-gcm-tutorial.html`

Démo : le "client"

- Le "client" = le récepteur de messages venant du cloud
- = une application Android qui sera exécutée dans un smartphone
- Essentiellement le code est :
  - on enregistre le smartphone dans le cloud Google
  - on vérifie le bon déroulement de cette phase d'enregistrement
  - on récupère le regId associé au couple (projet, smartphone)
  - et puis voilà !
- Euh, et quand recevons nous le message ?
- Il faut en être informé. Par l'environnement d'exécution qui, lui, le reçoit
- D'où architecture Android :
  - l'application Android possède un service spécial dédié au cloud qui est déclenché lors de la réception de messages provenant du cloud
  - les messages reçus sont encapsulés dans des Intent
  - à réception d'un message, celui-ci est affiché

IHM du client : réception du regId

- Essentiellement, l'activité principale de l'application Android est :

```
...
import com.google.android.gcm.GCMRegistrar;

public class SamplePushActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
     ...
     // On demande d'enregistrer le smarphone dans le cloud
     GCMRegistrar.checkDevice(this);

     // On vérifie que l'Android Manifest est correct pour utiliser GCM
     // (toutes les permissions nécessaires)
     GCMRegistrar.checkManifest(this);
     if (GCMRegistrar.isRegistered(this)) {
          // Si l'enregistrement dans le cloud est correct, on affiche le regId
          // (dans LogCat)
          Log.d("info", GCMRegistrar.getRegistrationId(this));
     }
     // On récupère le regId qu'on affiche dans l'IHM du smartphone
     final String regId = GCMRegistrar.getRegistrationId(this);
     ...
```

Le récepteur de messages diffusés du client

- Il est utilisé un `Service` et pas un `BroadcastReceiver`. Cela peut paraître étrange !
- Conceptuellement lorsqu'on reçoit un message du cloud, il faut lancer un traitement de ce message. Une classe Service de l'API encapsule la réception du message et le lancement du service. C'est la classe `GCMBaseIntentService`
- L'appli Android possède donc un tel `(Intent)Service` :

```
...
import com.google.android.gcm.GCMBaseIntentService;

public class GCMIntentService extends GCMBaseIntentService {
     public GCMIntentService() {
          super("LE_PROJET_GCM_NUMBER");
     }

     protected void onMessage(Context arg0, Intent lIntentRecu) {
          // Traitement du message reçu.
          // On peut récupérer le message par
          // lIntentRecu.getStringExtra("message")
     }
     ...
}
```

L'AndroidManifest du client

- Pour contacter le cloud, internet, etc. il faut avoir des permissions
- Il faut les indiquer explicitement dans l'Android Manifest de l'application Android
- Voir à `http://developer.android.com/google/gcm/client.html`, Step 2: Edit Your Application's Manifest
- Une partie du Manifest est alors :

```
<manifest ...
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <permission
        android:name="fr.cnam.ssoiot.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />

    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
    <uses-permission android:name="fr.cnam.ssoiot.permission.C2D_MESSAGE" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
   <uses-permission android:name="android.permission.USE_CREDENTIALS" />

    <application ...
     activité principale
```

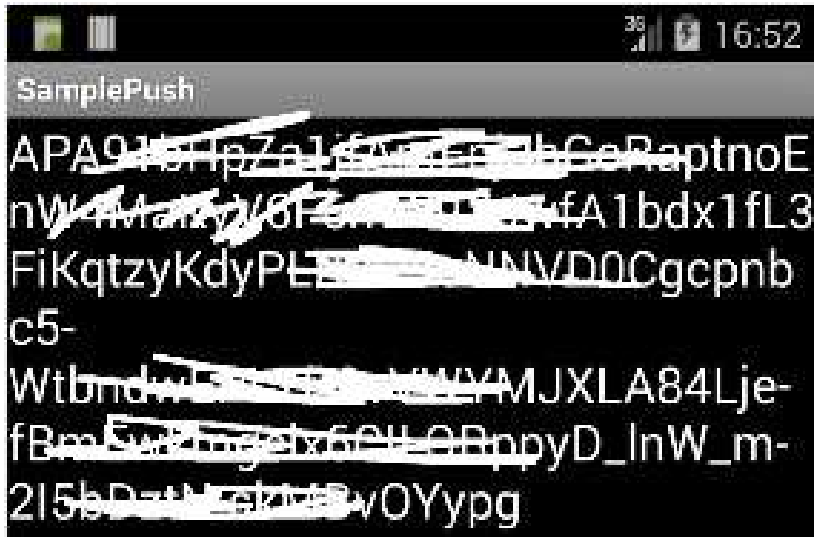
La partie receiver de l'AndroidManifest

- La partie receiver est :

```
<application ...
 activité principale

    <receiver
        android:name="com.google.android.gcm.GCMBroadcastReceiver"
        android:permission="com.google.android.c2dm.permission.SEND" >
        <intent-filter>
            <action android:name="com.google.android.c2dm.intent.RECEIVE" />
            <action android:name="com.google.android.c2dm.intent.REGISTRATION" />

            <category android:name="fr.cnam.ssoiot" />
        </intent-filter>
    </receiver>

    <service android:name="fr.cnam.ssoiot.GCMIntentService" />
</application>
```

Demo : exécution du client

- On doit obtenir le regId affiché comme :



- Euh, sans les gribouillis ;-)

Démo : le "serveur"

- Le "serveur" = l'émetteur de messages dans le cloud
- C'est une application Java J2SE
- Essentiellement le code est :
  - construire un émetteur de messages dans le cloud
  - construire un message pour le cloud
  - envoyer, dans le cloud, ce message pour une liste de smartphone

Code de l'émetteur (= serveur)

- L'activity principale est :

```
...
import com.google.android.gcm.server.Message;
import com.google.android.gcm.server.MulticastResult;
import com.google.android.gcm.server.Sender;

public class EmetteurDansLeCloud {
public static void main(String args[]) {
     try {
          Sender sender = new Sender("L_API_KEY_DU_PROJET_GCM");
          ArrayList<String> devicesList = new ArrayList<String>();

          String monRegIdDuNexus5 = "LE_REG_ID_D_UN_SMARTPHONE_DESTINATAIRE";
          devicesList.add(monRegIdDuNexus5);
          // construction d'un message à envoyer
          Message message = new Message.Builder()
                    .collapseKey("1")
                    .timeToLive(3*60)
                    .delayWhileIdle(true)
                    .addData("message",
                              "Hello sso and iOt school")
                    .build();

          MulticastResult result = sender.send(message, devicesList, 2);
          sender.send(message, devicesList, 1);
```

Un exercice

- Construire une application Android application qui reçoit des messages envoyés dans le cloud par une application Java

Publier dans GCM avec HTTP

- voir à `http://developer.android.com/google/gcm/http.html`
- Si on veut utiliser HTTP pour GCM, il faut :
  - envoyer une requête POST à `https://android.googleapis.com/gcm/send`
  - fabriquer un message HTTP avec un en-tête indiquant un type de contenu (`Content-Type: application/json` pour JSON ou `application/x-www-form-urlencoded;charset=UTF-8` pour du texte brut) et la clé (key=l'API key)
  - Par exemple :

```
Content-Type:application/json
Authorization:key=AIzaSyB-1uEai2WiUapxCs2Q0GZYzPu7Udno5aA

{
  "registration_ids" : ["APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx..."],
  "data" : {
    ...
  }
}
```

- Ci dessus une liste des regIds et des données pour une appli GCM de clé `AIz...`

Bibliographie pour GCM

- http://android.amolgupta.in/2012/07/google-cloud-messaging-gcm-tutorial.html: un bon exemple
- http://developer.android.com/google/gcm/index.html: le site de réference pour GCM
- http://androidexample.com/Android_Push_Notifications_using_Google_Cloud_Messaging_GCM/index.php?view=article_discription&aid=119&aaid=139 et http://androidexample.com/Device_To_Device_Messaging_Using_Google_Cloud_Messaging_GCM_-_Android_Example/index.php?view=article_discription&aid=122&aaid=142: deux autres tutoriaux

- http://hmkcode.com/android-google-cloud-messaging-tutorial/: un autre exemple

Fin