Android and Cloud Computing

Schedule

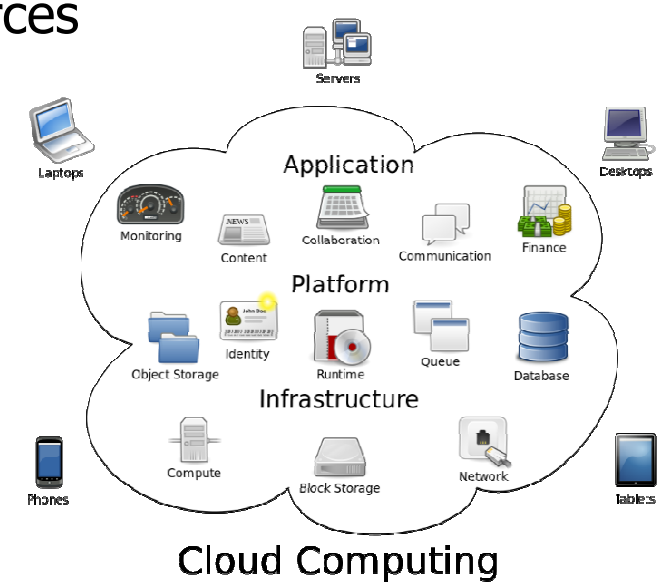
- Reminders on Android and Cloud
- GCM presentation
- GCM notions
- Build a GCM project
- Write a GCM client (receiver)
- Write a GCM server (transmitter)

Android : reminders ?

- Android = execution environment (too)
- A sofware in Android = Android application
- Android application = components into this environment
- => components' life are supported by the environment
- Android application are informed by this messages (sent by environment) : the `Intent`
- An Android application is described by its `AndroidManifest.xml`

Cloud computing : reminders

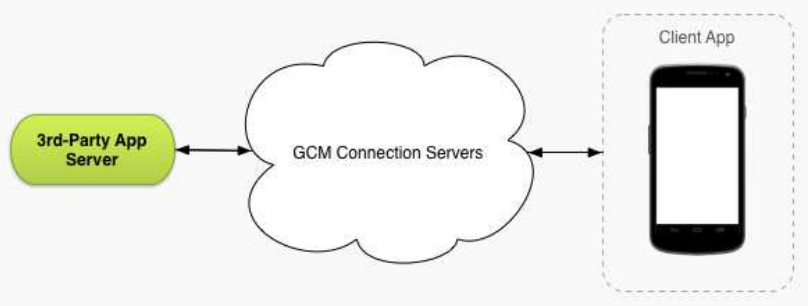
- Cloud computing is internet-based computing in which large groups of remote servers are networked to allow sharing of data-processing tasks, centralized data storage, and online access to computer services or resources



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

Google Cloud Messaging (GCM)

- Allow messages exchanges between application server and Android devices using Google cloud
- New version of C2DM (Cloud to Device Messaging) (stopped at 26 june 2012)
- asynchronous mode (= mail) : devices don't need in use to receive messages. It will received it when it will be used
- Need Android 2.2 at least (very old version ;-), 1% found now)
- GCM = middleware between devices and server
- ~ JMS architecture (Java Messaging Service)
- free and without quota



GCM architecture

- Design pattern "publish-subscribe"
- Asynchronous : not need to be listening to receive
- => GCM architecture store the message to send to the device until the device receive it (but deadline = 4 weekd)
- "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`

GCM middleware

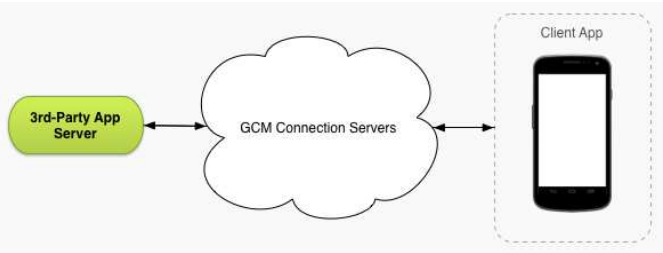
- "The application server sends a message to GCM servers.
- Google enqueues and stores the message in case the device is offline.
- When the device is online, Google sends the message to the device.

- 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."

- source : `http://developer.android.com/google/gcm/server.html#send-msg`

GCM on devices

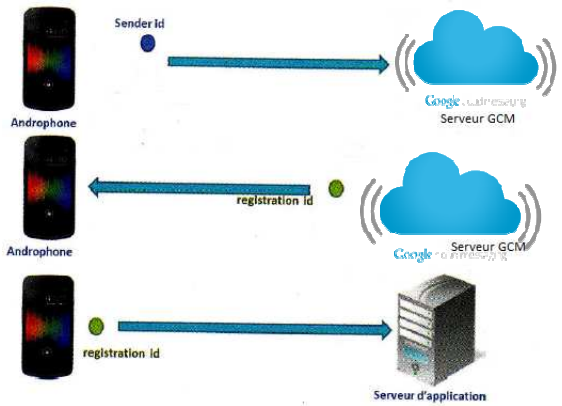
- Android applications know GCM incoming messages by receiving an `Intent` (= Android architecture)
- Can be used on AVD
- No need to put the application in the Google Play Store
- Use Google Services
- Two important notions: component and credential (= identificator)
- Components = the 3 entities:
  - A 3rd-party Application Server
  - Android devices (= Client App)
  - GCM = infrastructure GCM = Google SaaS



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

GCM Architecture : credentials

- The Sender Auth Token called API Key = API Key given to application server needed to use GCM = obtained in the "Google Developers Console"
- The Sender ID : the project number of the GCM project. Obtained from the "Google Developers Console"
- The registration ID : credentials given dynamically by GCM which characterizes the pair (device, android application) : " 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`

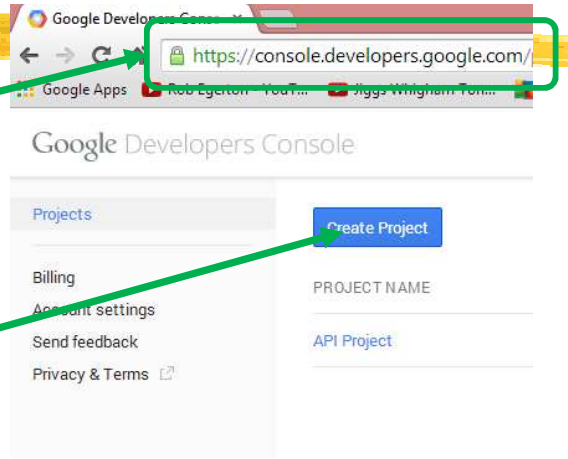
Build a GCM applications: the steps

- First build a GCM project = a GCM middleware which we will use
- 1°) Go to the "Google Developers Console" and build it

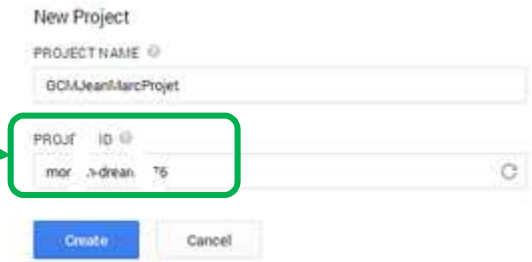
- 2°) Write a server (transmitter) application using this GCM project
- 3°) Write a customer (receiver) Android application

Create a GCM project (1/4)

- Go to the "Google Developers Console" which URL :
  It is often
  `https://code.google.com/apis/console/`
  but write "Google Developers Console" in
  Google is good too ;-)
- If you haven't already a project, create it



A name is proposed



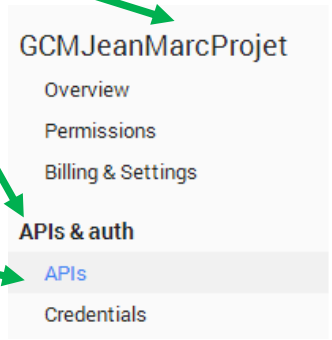
- We will obtained a project number (long integer) which will be the sender ID
- source :
  `http://developer.android.com/google/gcm/gs.html#next`

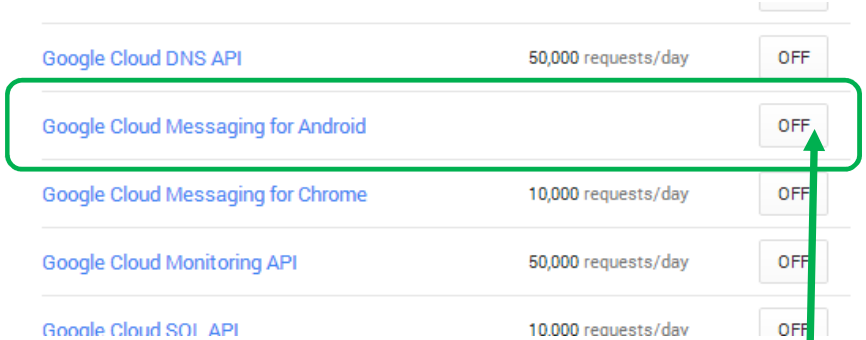
Create a GCM project (2/4)

- You must ask for using GCM Service for this Google project. So select the project

Select APIs & auth

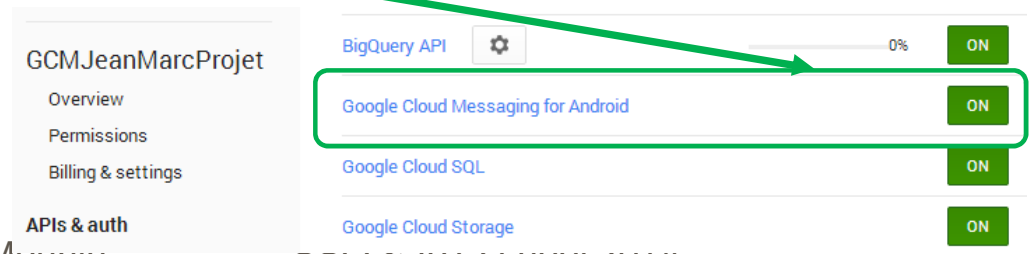
then APIs





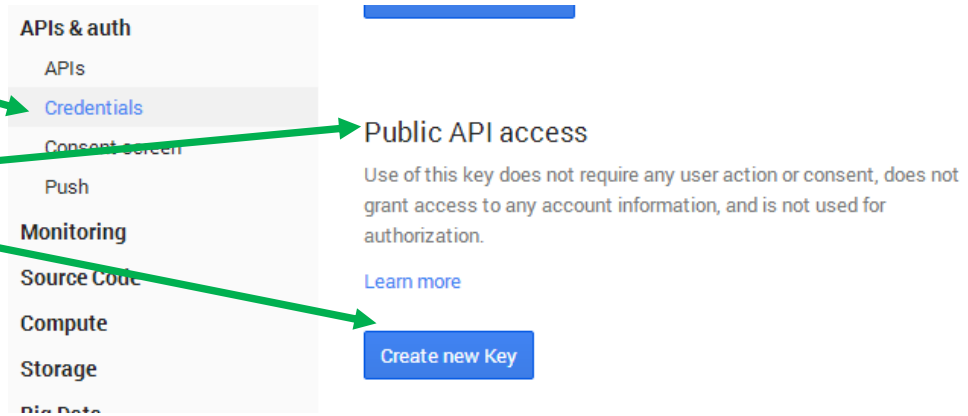
then select the button (Off for now) of Google Cloud Messaging for Android

- The line disappears (thanks ergonomics !). It is at the beginning of the page (ON put) !

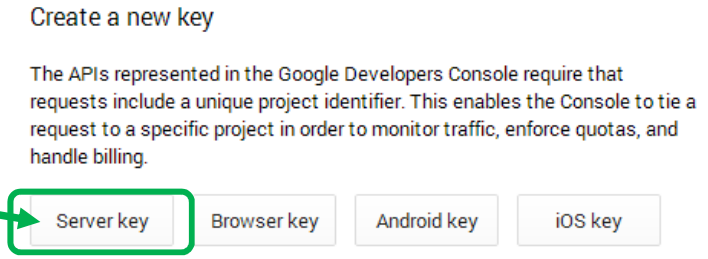


Create a GCM project (3/4)

- Select Credentials then,
- Public API Access part, and click Create new Key

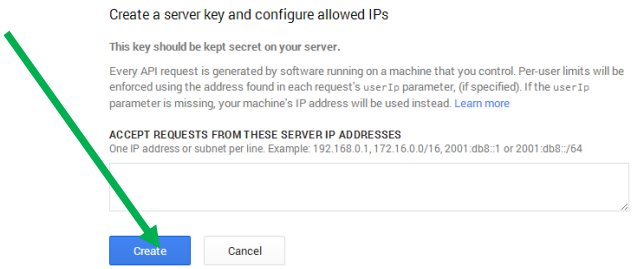


- In the dialog box click, Server key

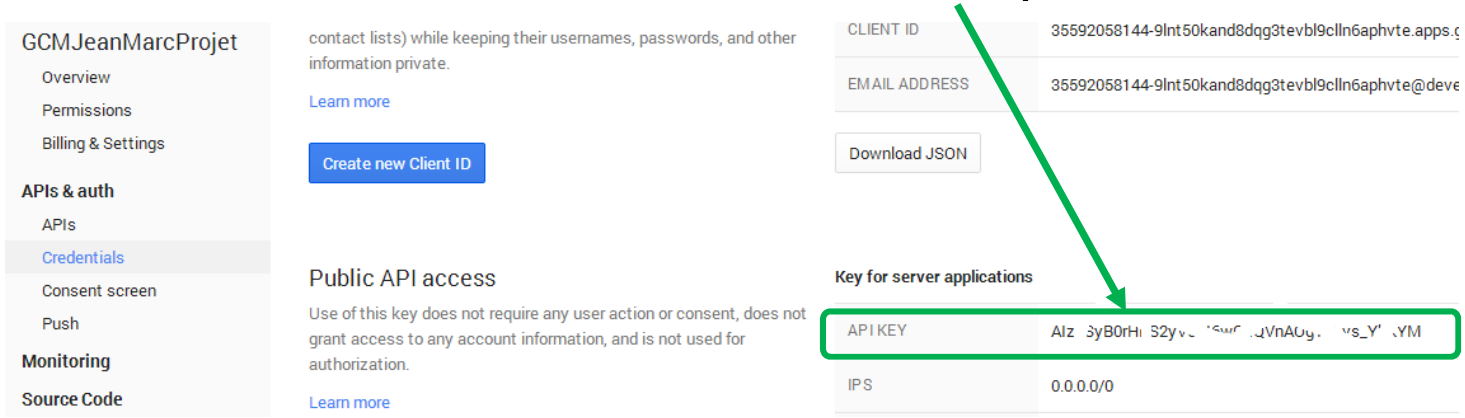


Create a GCM project (4/4)

- In the next dialog box, put the IP address of the application server which will use this GCM project. Click Create
- We can put `0.0.0.0/0` for the tests

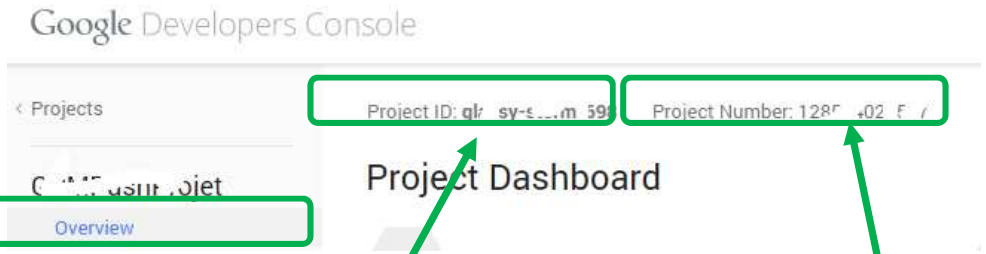


- Then in the next window we obtain the API Key :



The GCM project : a summary

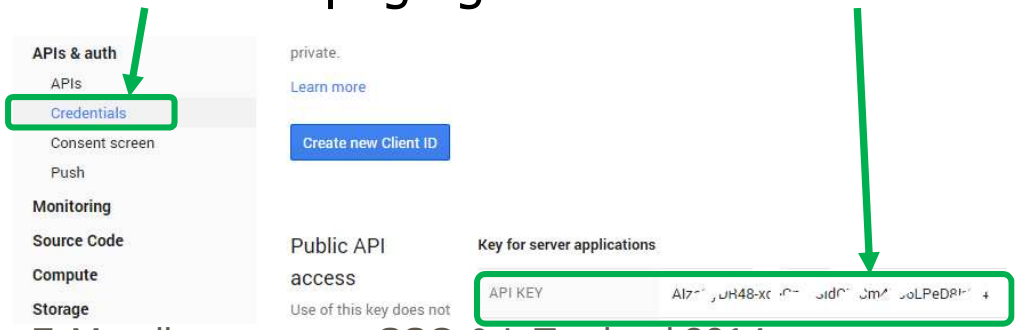
- We always can obtain the informations from a GCM project by going to the Google Developers Console
- Go to `https://code.google.com/apis/console/` (= the Google Developers console)
- Sélect the projet



then Overview

At the top of the page there is the project ID (not used) and the Project Number

- Click Credentials. The page gives the API KEY



Type of connection

- We can use HTTP or XMPP
- With HTTP, we can have only application server -> Cloud -> devices routes
- With HTTP, this sendings are synchronous (so blocking)

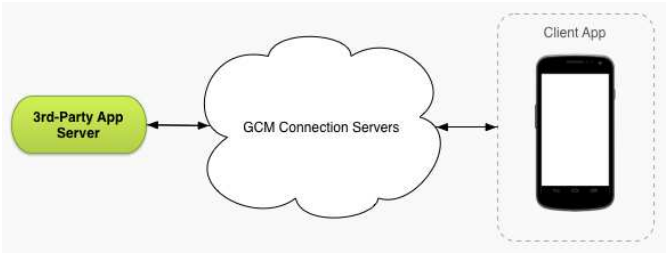
- With XMPP, we can have the two routes and the asynchronous sendings
- This type of connection (with XMPP) is called CCS (Cloud Connection Server)
- source : `http://developer.android.com/google/gcm/ccs.html`
- We can use both types in a same application

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

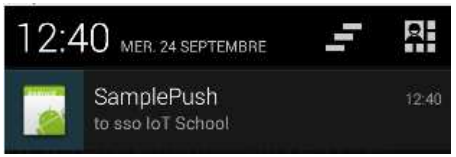
A demo

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

Java SE application



Android application



- Coming from the tutorial : `http://android.amolgupta.in/2012/07/google-cloud-messaging-gcm-tutorial.html`

Demo : the "client"

- "client" = receiver of messages coming from the cloud
- = a Android application exécuted on a Android device (eventually AVD)
- The code is:
  - the device is registered in the cloud
  - checking of this registration
  - obtention of its regId associated to the pair (project, device)
  - it's all folks !
- Euh, and when we receive a message ?
- We must know it. By Android
- So Android architecture :
  - a Android application has a special service for the cloud which is used when a message from the cloud is coming
  - messages are marshalled into Intents
  - when a message is received, it is displayed

GUI of the client : regId reception

- The big part of the principal activity is:

```
...
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);
     ...
```

When receiving a cloud message

- We use a `Service`, not a `BroadcastReceiver`. It seems strange
- When a message is received from the cloud, we must treate it
- A Service class of the GCM API treates the reception of the message and the treatment : `GCMBaseIntentService`
- The Android application has this (`Intent`)`Service` :

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

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

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

The AndroidManifest of the client

- Using cloud, internet, etc. need permissions
- Put them in the Android Manifest of the Android application
- Voir à http://developer.android.com/google/gcm/client.html, Step 2: Edit Your Application's Manifest
- So a part of the Manifest is:

```
<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
```

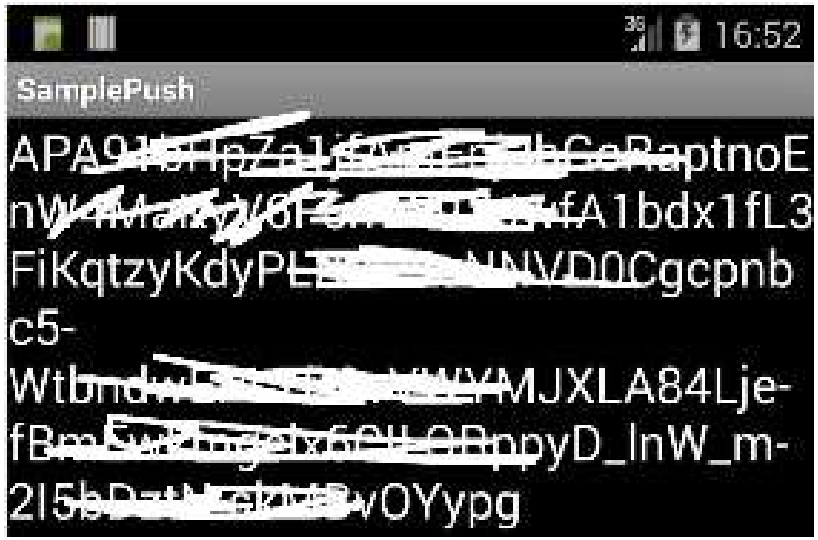
receiver part of the AndroidManifest

- The receiver part is:

```
<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 : execution of the client

- You must obtain something as:



- But not scrawled ;-)

Demo : "server"

- "server" = the sender of messages into the cloud
- = a Java application J2SE
- The code is:
  - build an sender of messages to the cloud
  - build a un message
  - send it to a list of devices

Code of the server

- Essentially we have:

```
...
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 = "THE_REG_ID_OF_A_RECIPIENT_DEVICE";
          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);
```

An exercise

- Build an Android application to receive messages from the cloud and a Java application to push messages in the cloud

Use GCM with HTTP

- source: `http://developer.android.com/google/gcm/http.html`
- If we want to use HTTP for GCM, we must:
  - send a HTTP POST request to `https://android.googleapis.com/gcm/send`
  - write an HTTP message with an header having a content type and the API keyFor example: `Content-Type: application/json` pour JSON, `application/x-www-form-urlencoded;charset=UTF-8` for plain text
  - Example :

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

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

- Above, a list of regIds and datas for a GCM application with API Key `AIz...`

Bibliography for GCM

- http://android.amolgupta.in/2012/07/google-cloud-messaging-gcm-tutorial.html: a good example
- http://developer.android.com/google/gcm/index.html: reference site for GCM

- http://androidexample.com/Android_Push_Notifications_using_Google_Cloud_Messaging_GCM/index.php?view=article_discription&aid=119&aaid=139 and http://androidexample.com/Device_To_Device_Messaging_Using_Google_Cloud_Messaging_GCM_-_Android_Example/index.php?view=article_discription&aid=122&aaid=142: two other tutorials

- http://hmkcode.com/android-google-cloud-messaging-tutorial/: another example

The end