Date: 04.12.2015

.

ACRA AND ACRALYZER AN ANDROID CRASH REPORTING CLOUD SERVICE

Authors:

Md. Shohrab Uddin
shobuzch@gmail.com

Sneha Gupta
gupta17.sneha@gmail.com

.

1. SERVER SIDE CONFIGURATION

With ACRA, custom back-end (with options ranging from free to commercial solutions) is by far the most used cloud service. Since google has stopped the support of google docs used in ACRA for the error data persistence, the other option include usage of **acralyzer**. Also Acralyzer is the new open source backend created by the author of ACRA, and is the official backend support from ACRA. It is an open source backend reports analysis webapp built on a full open stack.

Hosting the ACRA reports using Acralyzer is done in 2 steps:

Step1: Hosting your ACRA reports on a Cloudant back-end

Step2: Visualizing the data with acralyzer

Step 1: Hosting your ACRA reports on a Cloudant (from IBM) back-end.

i. Open an account at IBM Cloundant from https://cloudant.com/sign-up/. This will provide space on the cloud where your error reports are stored. Login after registration.

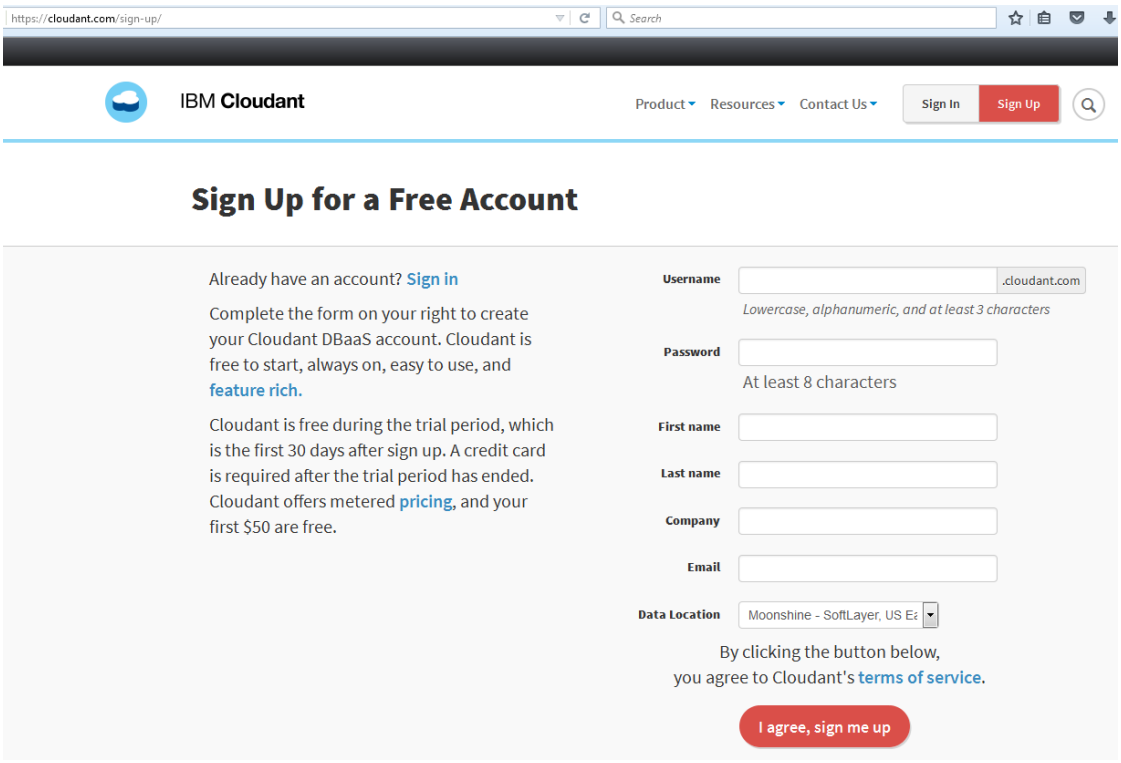

Figure 1: Registration at IBM Cloundant

.

ii) This back-end consists of two components – **Storage Database** (apache CouchDB) where errors from your app are stored and **Web app** (for analysis – CouchApp) where reports are generated.

iii) Replicate two databases in Cloudant: Click on Replicate button from dashboard. In SROUCE DATABASE, click on remote database tab section, and in TARGET DATABASE, click on new database tab section. Enter the following details to replicate two databases by clicking on replicate button.

Please note that, the target database naming convection is acra-<your_db_name> and for acralyzer database the target database name will be acralyzer.

Source database	Target Database
http://get.acralyzer.com/distrib-acra-storage	acra-[your_app_name] (Figure 2)
http://get.acralyzer.com/distrib-acralyzer	acralyzer (Figure 3)

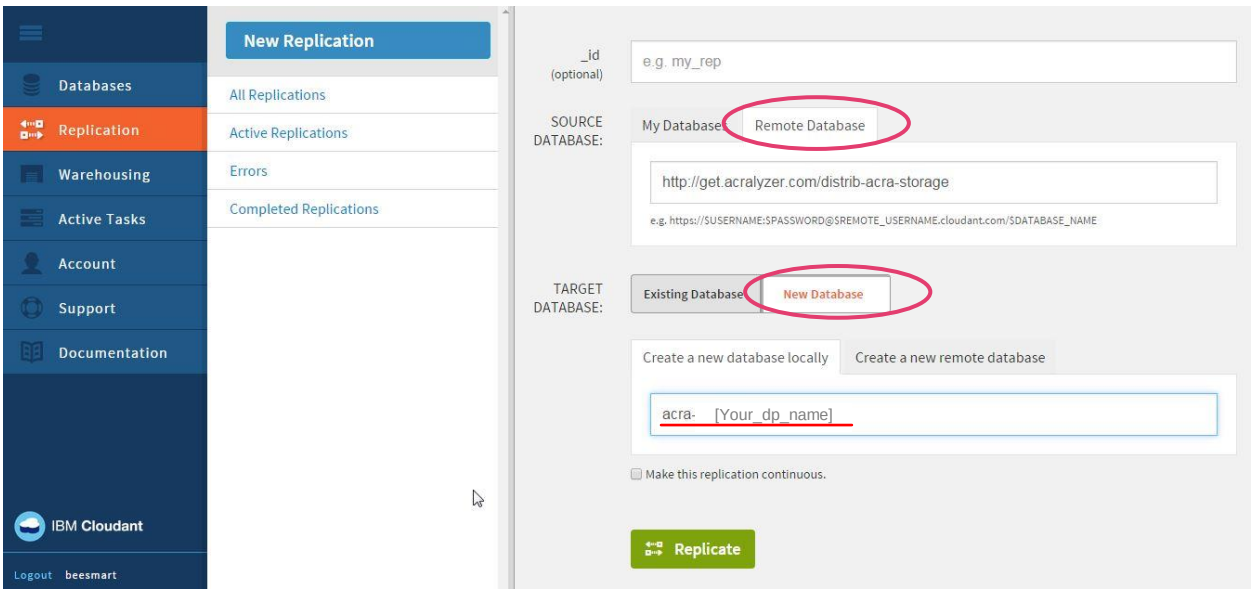

Figure 2: Database Replication for your app

.

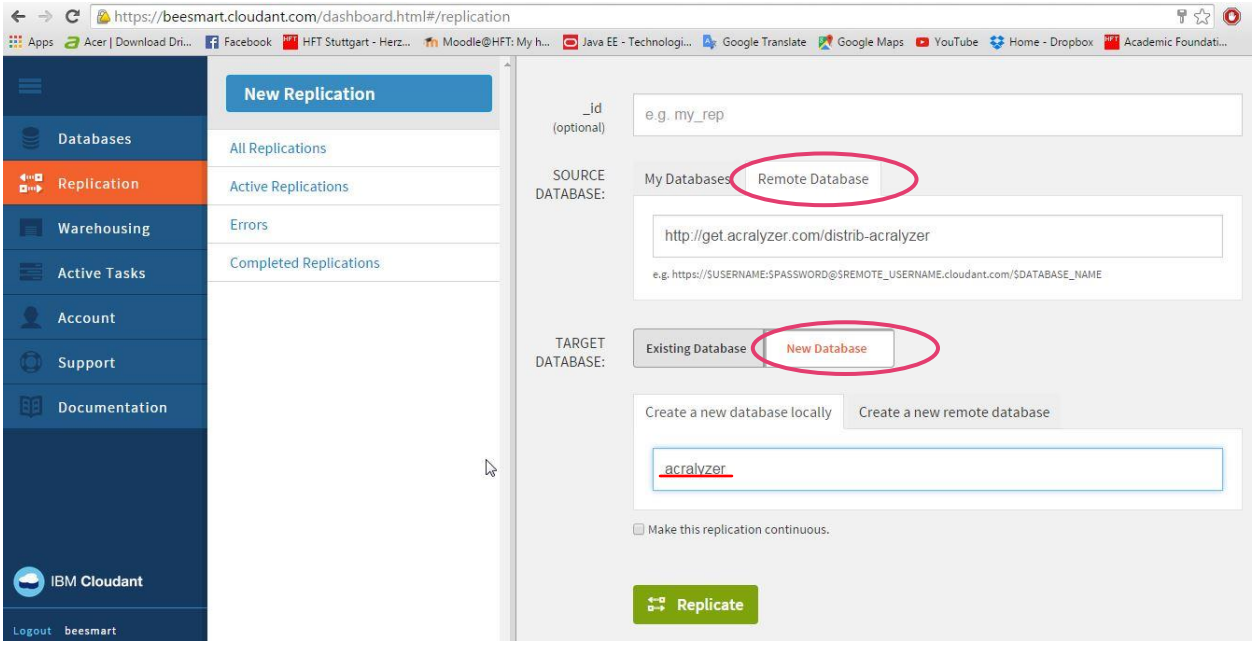

Figure 3: Database Replication for acralyzer

iv) Select Databases->Permission section for acra-[your_app_name] database and generate API keys. It is automated key/password pair, which is used in android for basic user and password authentication.

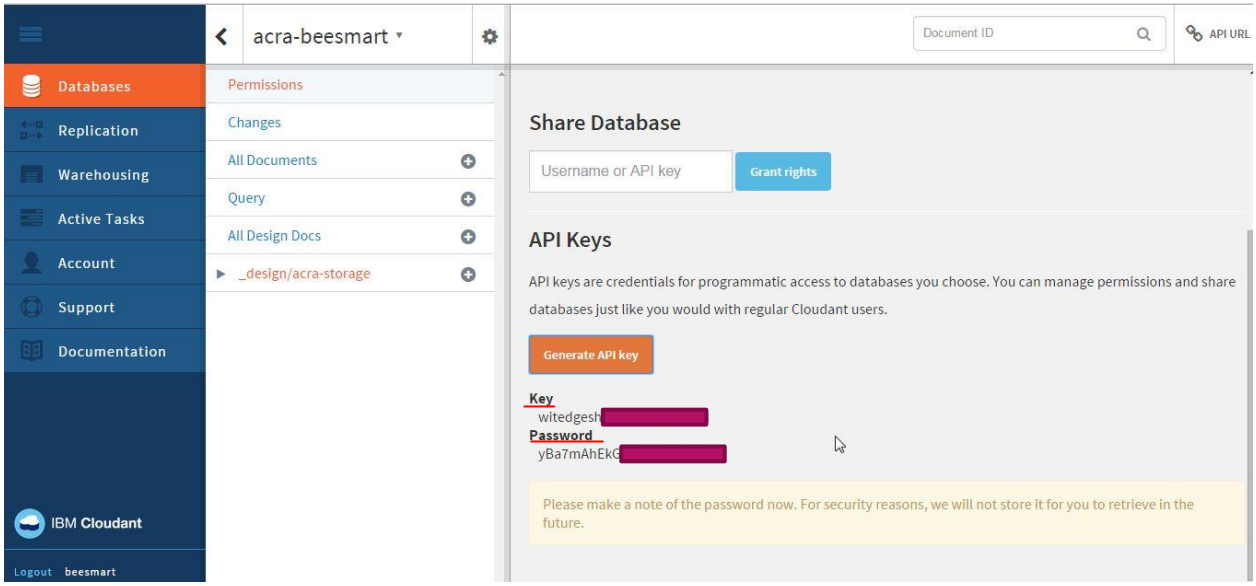

Figure 4: API Key Generation

.

Step 2: Visualizing the data with acralyzer - once database is replicated successfully, the acralyzer dashboard can be easily accessed by following URL:

https://<cloudant_user_name>.cloudant.com/acralyzer/_design/acralyzer/index.html#/dashboard/

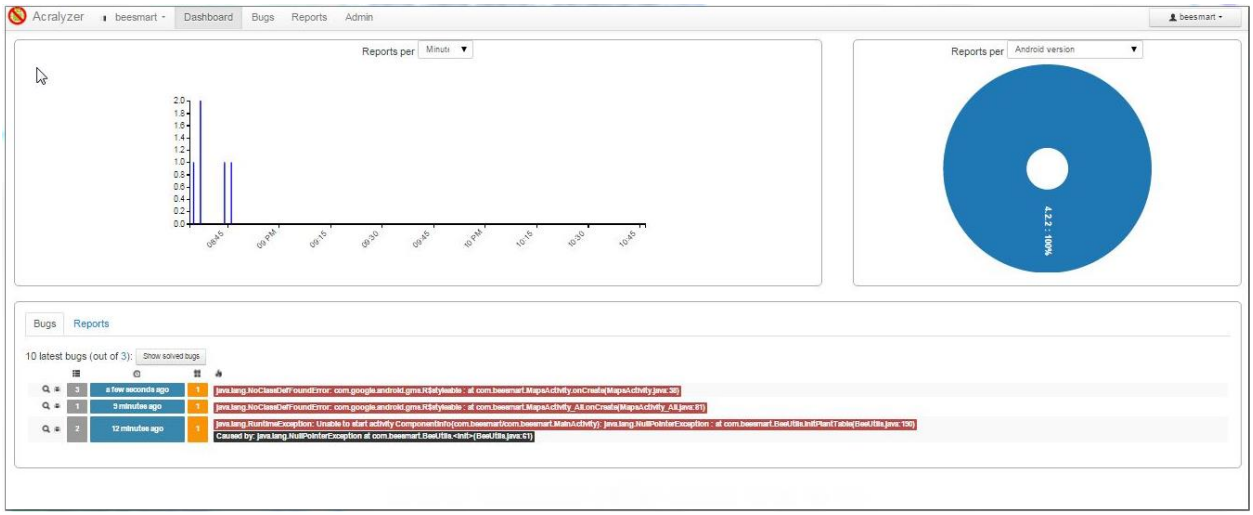

Figure 5: Acralyzer Dashboard

The error which happens in your app can be directly viewed here. It can also be used to generate reports. The real time pooling in by default is on. So you can view errors as soon as it occurs.

There are various screens in the dashboard to view the bugs in your app.

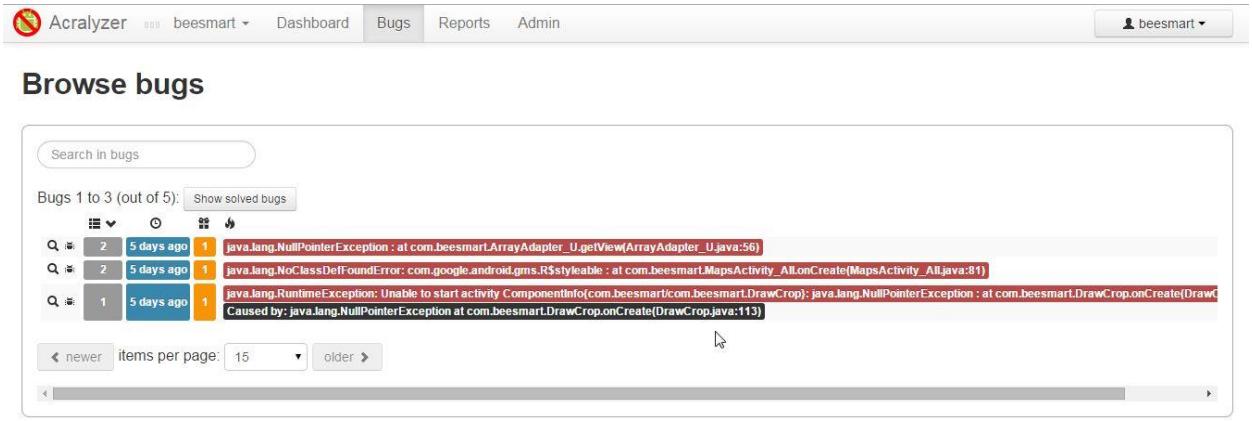

Figure 5: Acralyzer Bugs Tracing

.

Pros and cons of using Acralyzer

Pros:

- Easy to install
- Free and open-source. (for small usage, simple pricing structure)
- Decent UI and search functionality.

Cons:

- Acralyzer is free and open-source, you still need to use couch hosting (IrisCouch, apache CouchDB) but small usage is basically free.

.

2. CLIENT SIDE CONFIGURATION

1. STEPS FOR USING ACRA IN ANDROID:

- Add the file acra-4.6.2.jar or latest version in the libs folder of your android project. You can download the jar from this site http://repo1.maven.org/maven2/ch/acra/acra/4.6.2/ or add `compile 'ch.acra:acra:X.Y.Z'` dependency in your app's build.gradle file.
- Next add a class which extends Application class of android. If you already have a class which extends Application and being used in Manifest then update that class as shown below. The class should be annotated with **@ReportsCrashes** annotation of acra with following configurations.

```
1. @ReportsCrashes(
2.     formUri = "https://{cloudant_username}.cloudant.com/acra-
   {your_cloundant_db_name}/_design/acra-storage/_update/report",
3.     reportType = HttpSender.Type.JSON,
4.     httpMethod = HttpSender.Method.POST,
5.     formUriBasicAuthLogin = "THE_KEY_FROM_CLOUDANT",
6.     formUriBasicAuthPassword = "THE_PASSWORD_FROM_CLOUDANT",
7.         customReportContent = {
8.             ReportField.APP_VERSION_CODE,
9.             ReportField.APP_VERSION_NAME,
10.            ReportField.ANDROID_VERSION,
11.            ReportField.PACKAGE_NAME,
12.            ReportField.REPORT_ID,
13.            ReportField.BUILD,
14.            ReportField.STACK_TRACE
15.    },
16.    mode = ReportingInteractionMode.SILENT,
17.)
18.
19.public class MyApp extends Application {
20.
21.    @Override
22.    public void onCreate() {
23.        super.onCreate();
24.        // The following line triggers the initialization of ACRA
25.        ACRA.init(this);
26.    }
27.}
```

There are lot more configuration options. You can experiment a bit here.

Figure 6: Code Snippet to configure Acralyzer in Android

.

- In AndroidManifest.xml file do the following,
  a) Add permission android.permission.INTERNET and android.permission.READ_LOGS
  b) Add the class name in which you have configured Acralyzer.

```
<application
    android:name=".MyApp"
    android:theme="@style/AppTheme">
    ...
```

Save all the files and build the application. You will be able to view all the occurring errors in the acralyzer dashboard for your android application.

3. Other Backend Frameworks

1. **Crittercism:** Commercial. (With free plan also available)

Pros:

- They have a free plan (not recommended - cumbersome).
- The crash data includes stacktraces for all threads.
- They support crash report aggregation.

Cons:

- Logs not supported properly: only logcat support, which is an issue for android versions prior to 4.1
- Support for handled exceptions requires premium membership.
- Support for extra report data is cumbersome (you can only see it by downloading a JSON file)
- You can't customize the exception handler: when a crash happens, the app closes as usual. With ACRA, there's a variety of responses possible, including attempting to stay inside the app.

How to use:

http://docs.crittercism.com/quickstart.html

.

2. **HockeyApp:** Entry-level pricing ($10/month) out of the commercial solutions.

Pros:

- They do crash report aggregation, by stacktrace and by version.
- They have a dedicated feedback mechanism.
- They seem to have quite a complete REST query API.
- The UI is fairly nice.

Cons:

- Not free.

How to use:

http://support.hockeyapp.net/kb/client-integration-android-other-platforms/hockeyapp-for-android-sdk

Thank you and Best of luck for Crash hunting!!