# codePush实践方案及注意事项

一：前言

****CodePush****是微软提供的一套可用于React Native和Cordova的热更新服务。CodePush相当于一个中心仓库，开发者可以推送当前的更新（包括JS/HTML/CSS/IMAGE等）到CoduPush，然后应用将会查询是否有更新。

二：实现步骤

1. 安装 ****CodePush CLI****。管理 CodePush 账号需要通过 NodeJS-based CLI.  
   只需要在控制台输入 npm install -g code-push-cli，就可以安装了。  
   安装完毕后，输入 code-push -v查看版本代表成功。
2. 创建CodePush账号。在控制台输入 code-push register 后，将会打开一个网页进行注册CodePush账号支持 github和 Microsofe，选其中一个就可以了。授权成功后，网页将会显示一个token,复制到控制台就成功了。
3. 在codePush服务器注册APP。为了让codePush服务器知道你的app，我们需要向它注册app： code-push app add <appName>，就可以了。
4. 在app上添加CodePush SDK，配置升级相关代码。

****第一步****。在应用中安装react-native插件，npm install --save react-native-code-push

****第二步****。在Anroid project中安装插件。  
CodePush提供了两种方式：**[RNPM](https://github.com/rnpm/rnpm" \t "http://bbs.reactnative.cn/topic/725/_blank)** 和 ****Manual****。  
如果你不想依赖其他工具或者愿意走多几步额外的步骤，可以使用 Manual。不过像我这么懒的代码从业者，毫不犹豫地选择了 ****RNPM**** 这个实用工具。

****第三步**** 运行 npm i rnpm 安装 RNPM。

****第四步**** 运行 rnpm link react-native-code-push。这条命令将会自动帮我们在anroid文件中添加好设置（其实就是通过Manual的安装步骤）

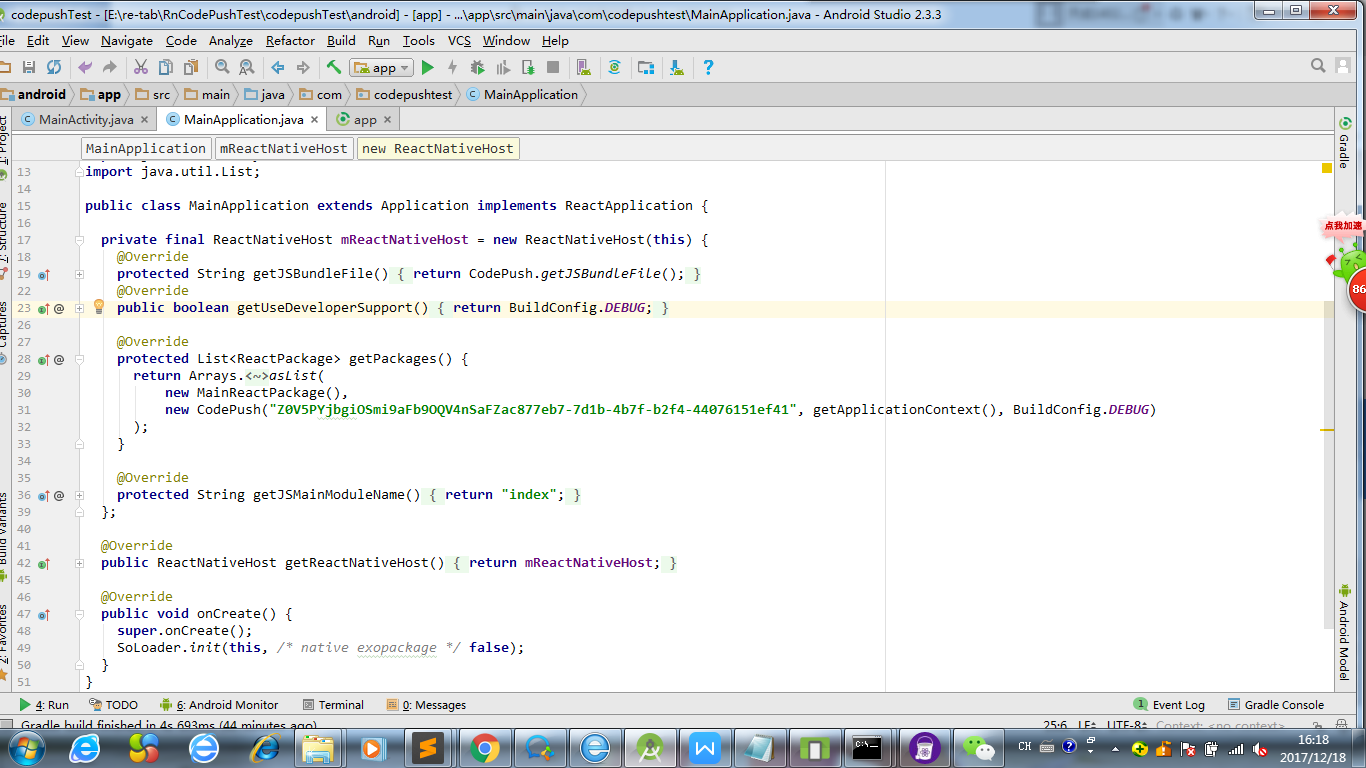
****第五步**** 在 android/app/build.gradle文件里面添加额为的创建任务：

apply from "react.gradle"

apply from "../../node\_modules/react-native-code-push/android/codepush.gradle"

****第六步**** 运行 code-push deployment ls <appName>获取 ****部署秘钥****。默认的部署名是 ****staging****，所以 ****部署秘钥（deployment key ）**** 就是 staging的可以。

****第七步**** 添加配置。我们需要让app向CodePush咨询JS bundle的所在位置，这样CodePush就可以控制版本。更新 MainApplication.java文件：



****第八步**** 在 android/app/build.gradle中有个 android.defaultConfig.versionName属性，我们需要把 应用版本改成 1.0.0（默认是1.0，但是codepush需要三位数）。

android{

defaultConfig{

versionName "1.0.0"

}

}

****第九步**** CodePush 插件下载和关联完毕后，就剩下在应用中部署更新控制策略：

检查更新时机,更新是否强制，更新是否要求即时等等.

检查更新可以打开app检查，在后台运行是检查，生成组件前更新

如果是强制更新需要在发布的时候指定，发布命令中配置--m true

在更新配置中通过指定installMode来决定安装完成的重启时机，亦即更新生效时机

****codePush.InstallMode.IMMEDIATE****：表示安装完成立即重启更新

****codePush.InstallMode.ON\_NEXT\_RESTART****：表示安装完成后会在下次重启后进行更新

****codePush.InstallMode.ON\_NEXT\_RESUME****：表示安装完成后会在应用进入后台后重启更新

5：更新代码后，发布一个应用更新到服务器

发布更新之前，需要先把 js打包成 bundle，以下是anroid的做法：

****第一步**** 在 工程目录里面新增 bundles文件：mkdir bundles

****第二步**** 运行命令打包 react-native bundle --platform 平台 --entry-file 启动文件 --bundle-output 打包js输出文件 --assets-dest 资源输出目录 --dev 是否调试。  
这是我的打包命名： react-native bundle --platform android --entry-file index.android.js --bundle-output ./bundles/index.android.bundle --dev false  
需要注意的是：

* 忽略了资源输出是因为 ****输出资源文件后，会把bundle文件覆盖了****。
* 输出的bundle文件名不叫其他，而是 index.android.bundle，是因为 ****在debug模式下，工程读取的bundle就是叫做 index.android.buundle****。
* 平台可以选择 android 或者 ios。
* 具体的命令可以参考[这里](http://reactnative.cn/docs/0.22/signed-apk-android.html" \l "%E7%94%9F%E6%88%90%E5%8F%91%E8%A1%8Capk%E5%8C%85" \t "http://bbs.reactnative.cn/topic/725/code-push-%E7%83%AD%E6%9B%B4%E6%96%B0%E4%BD%BF%E7%94%A8%E8%AF%A6%E7%BB%86%E8%AF%B4%E6%98%8E%E5%92%8C%E6%95%99%E7%A8%8B/_blank)

打包bundle结束后，就可以通过CodePush发布更新了。在控制台输入

code-push release <应用名称> <Bundles所在目录> <对应的应用版本>

--deploymentName 更新环境

--description 更新描述

--mandatory 是否强制更新

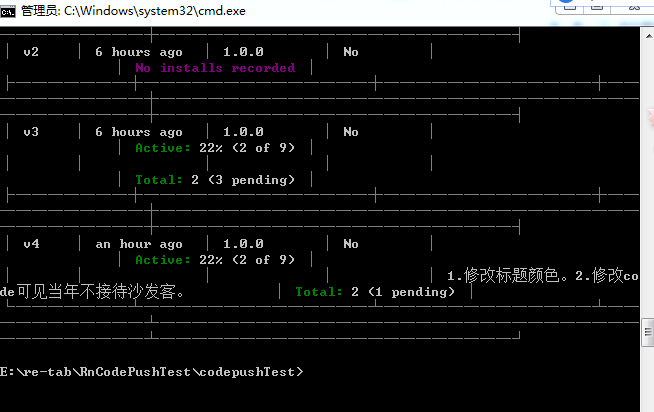
注意：

* CodePush默认是更新 ****staging**** 环境的，如果是staging，则不需要填写 deploymentName。
* 如果有 ****mandatory**** 则会让客户端强制更新
* 对应的应用版本（targetBinaryVersion）是指当前app的版本，而不是你填写的更新版本。譬如客户端版本是 1.0.0，如果我们需要更新客户端，那么targetBinaryVersion填的就是 1.0.0。（踩了坑，半夜调试到哭了- -）
* 在控制台输入 code-push deployment history <appName> Staging 可以看到版本更新的时间、描述等等属性。

这是我的输入

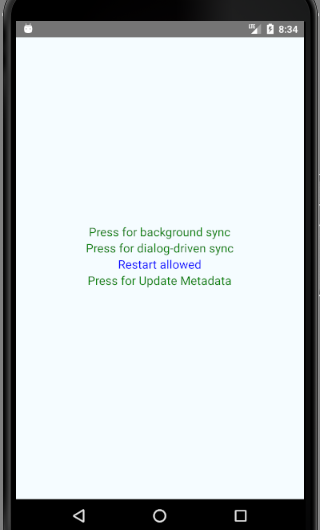
code-push release codepushTest2 ./bundles 1.0.1

这是测试查看升级的记录：



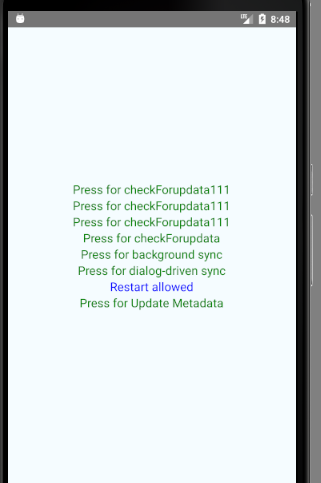
**6：**app收到升级推送

下面是测试的升级前，点击默认可以更新的弹窗以及升级后的截图：





这里就能看见测试的时候写的描述信息了。



7：codePush更新服务流程图：

8：注意事项：

A: 模拟机运行测试时候，需要注意将端口改为程序不认识的，这样确保下载到最新的安装包，不然是node服务器的jsbundle文件

B: 生成bundle文件的命令里面的版本信息是当前客户端的版本，不是需要更新的版本。

9： 常用命令：

##### **CODE-PUSH APP相关命令**

* add 在账号里面添加一个新的app
* remove 或者 rm 在账号里移除一个app
* rename 重命名一个存在app
* list 或则 ls 列出账号下面的所有app
* transfer 把app的所有权转移到另外一个账号

##### **部署APP相关命令**

* code-push deployment add <appName> 部署
* code-push deployment rename <appName> 重命名
* code-push deployment rm <appName> 删除部署
* code-push deployment ls <appName> 列出应用的部署情况
* code-push deployment ls <appName> -k 查看部署的key
* code-push deployment history <appName> <deploymentNmae> 查看历史版本(Production 或者 Staging)

##### **发布APP相关命令**

code-push release <应用名称> <Bundles所在目录> <对应的应用版本>

--deploymentName 更新环境

--description 更新描述

--mandatory 是否强制更新

比如我的测试：

生成环境：

code-push release codepush01 ./bundles/ 1.0.0 --deploymentName Production --description "1.修改标题颜色。" --mandatory false

测试环境：

code-push release codepushTest2 ./bundles/ 1.0.0

其他命令：

code-push deployment ls 您的应用名 -k

code-push deployment history 查看历史版本(Production 或者 Staging)如：

测试版本更新：code-push deployment history 应用名 Staging

生产版本更新：code-push deployment history 应用名 Production

codePush.InstallMode，有三种模式，

                 一个是立即启动，nstallMode.IMMEDIATE

                一个是下次启动安装：InstallMode.ON\_NEXT\_RESTART

                一个是程序在前台，并没有从后台切换到前台的情况下用的InstallMode.ON\_NEXT\_RESUME

 10： js 常用函数：

CodePush.disallowRestart();//页面加载的禁止重启，在加载完了可以允许重启

  CodePush.allowRestart();//在加载完了可以允许重启

 CodePush.sync（）

CodePush.checkForUpdate()

CodePush.getUpdateMetadata（）