React Native ❤ CodePush

Prabakaran Marimuthu Follow
Jan 30, 2018 · 7 min read

React Native ❤️
CodePush

It’s started two years ago, Love with React Native and still it continues. Whenever new app or product enters into our court, we are learning something new.

This time **“CodePush”** — a Pain Killer for frustrated mobile app updates (I know it’s too late for me).

In this blog, I will be sharing my experience with CodePush in React Native. Before getting started with CodePush, here is the quick intro about React Native and it’s architecture.

React Native

> Build native mobile apps using JavaScript and React

- Not a “mobile web app”, an “HTML5 app”, or a “hybrid app”.

- Uses the same fundamental UI building blocks as native **iOS** and **Android** apps.

Features

1. ***Hot & Live Reloading:*** When you need to make small changes in your app , you can use Hot Reloading to changes reflect in your app. For bigger changes impacting the logic of the app, it's better to use Live Reloading that reloads your app entirely as you make a change in the code.

2. ***Native code usage:*** Sometimes it's necessary to access to a platform API that corresponding RN module didn't exist. Using React native , we can easily combine native code components from Java, Objective-C and Swift.

3. ***Debugger Tools:*** With this tools, we can debug logic and GUI parts of the app. RN provides lot of debugging options from it's developer menu like Reload, GUI Inspector, Chrome Debugging and Performance Monitor.

Simple Architecture:

A RN app is composed of JavaScript files and any images, which are bundled together by the **packager** (a file called "android.js.bundle") .

This file packed within the platform-specific binary (.ipa or .apk file).

A simple React Native Architecture (Source: RN team)

Problem with updates 😩

In one of our projects, we faced this problem with updates, that is for an event specific app, we finished all features and uploaded app to respective stores, but later we realized that we forgot to add timezone to event time. Meh!!

> *For example, we coded* as “29 JAN 2018 07:10 AM” *instead of* “29 JAN 2018 07:10 AM IST” *. For this minor but important mistake, we had to rebuild the app and push it to the corresponding stores which took one more day to reach the users*

This is one of the biggest problem with mobile updates. Once the app is released in store, updating either the JavaScript code (e.g. making bug fixes, adding new features) or image assets,

- requires you to rebuild and redistribute the entire binary through respective stores.
- a time taking process to upload new app to stores and painful app review process.

Solution

CodePush

Microsoft CodePush (Source: Microsoft)

An **App Center** cloud service from Microsoft.

> *→ allows Cordova and RN developers to deploy mobile app updates directly to their user’s devices without help of Stores(OTA updates)*
>
> *→ more useful for fixing bugs or adding/removing small features that don’t require us to rebuild binary and redistribute through respective stores.*

How it works ?

How CodePush works?

- It is acting as a central repository that developer can publish updates and that apps can query and downloads from it — with help of their respective SDK
- This plugin helps end users to get app improvements/bug fixes instantly, by keeping our JavaScript and images synchronized with updates that we release to the CodePush server.
- As it's maintains a copy of the previous update, we can automatically revert back to previous state if we released new update with bug/ crash.

Important !

- *Any product changes which touch native code (e.g. modifying your AppDelegate.m/MainActivity.java file, adding a new plugin) cannot be distributed via CodePush, and therefore, must be updated via the appropriate store(s).*

Also it support from iOS (7+) Android (4.1+)

Setup CodePush:

1. Install CodePush CLI :

```
npm install -g code-push-cli
```

2. Create CodePush account using CLI :

```
code-push register
```

3. Register app with CodePush:

```
code-push app add MyApp android react-native
```

Once you successfully integrated , you will end-up with two default deployment keys for Staging and Production.

Using these keys , your app can communicate with CodePush server to download your updates for respective build type (Staging or Production).

Integrating with React Native:

Microsoft open source community provides best React Native wrapper for code push called react-native-code-push. Using this we can easily integrate CodePush service to our React Native app.

1. Installing CodePush RN Wrapper:

```
npm install --save react-native-code-push
```

2. Integrate with React Native,

a. using RNPM,

```
react-native link react-native-code-push
```

b. CocoaPods (for iOS only) ,

```
pod 'CodePush', :path => '../node_modules/react-native-code-push'
```

c. Manually (by gradle for android, adding lib package to iOS)

For full setup and integration document, please refer this https://github.com/Microsoft/react-native-code-push

> ***Note:*** *Don't forgot to add deployment keys to* ***AndroidManifest.xml*** *in Android and* ***Info.plist*** *in iOS*

Example:

In this example, the app mobile app communicate with CodePush server frequently, downloads updates silently and installed it. So the new updates available to the user whenever app gets started again.

Some of CodePush options

This wrapper provides many of methods to configure metrics like installatiom method, syncing frequency etc. Here is some important metrics.

1. **checkFrequency:** option to denote when you would like to check for updates. The options like `` `*ON_APP_START, ON_APP_RESUME, MANUAL*` ``

2. **installMode:** option to denote when you would like to install updates. `IMMEDIATE, ON_NEXT_RESTART, ON_NEXT_RESUME, ON_NEXT_SUSPEND`
3. **sync() :** This method synchronizes our app's JavaScript bundle and image assets with the latest release to the configured deployment. Works on two different mode '*Silent*' and '*Active*'.
4. **syncStatus :** You can find status of the synchronization at anytime. It's includes following values like 'UP_TO_DATE, UPDATE_IGNORED, UPDATE_INSTALLED, SYNC_IN_PROGRESS'

Releasing updates 😍

Once app has been configured and distributed to your users, and we've made some JS and/or asset changes, it's time to instantly release them!

```
code-push release <appName> <updateContents> <targetBinaryVersion> [--deploymentName <deploymentName>] [--description <description>] [--disabled <disabled>] [--mandatory] --rollout <rolloutPercentage>]
```

Example for MyApp Android:

This is the command to push updates to CodePush server for an android app called "Myapp" only staging environment. It should be run from the root directory of the project and it would build the js bundle automatically.

```
code-push release-react MyApp android -d "Production" -m --description "Timezone text added"
```

Release Options

1. `deploymentName:` Using this, we can specify exact deployment like 'Staging or Production'.
2. `description:` option to add description notes related with current release.
3. `rollout:` option to increase the rollout percentage of the target release.
4. `mandatory:` option to indicate whether the release should be considered mandatory or not.
5. `disabled:` option to update whether the release should be disabled or not

Other features

CodePush also provides some other features through CLI.

1. **Multi-Deployment**

Using this feature we can maintain multiple deployment and it can be achieved using specific deployment keys.

→ configure the CodePush plugin using a specific deployment key.

→ maintain deployments like staging and Production deployments.

→ ensure your updates to end users are valid one.

2. Patching Update Metadata

After releasing an update, you need to modify one or more of the metadata attributes associated with app like "Mandatory, Rollout %", disabled etc.

You can update current release as mandatory as below,

`code-push patch MyApp Production -m`

3. Promoting Updates

We can easily promote updates from one deployment to another level deployment. For example, promoting updates from `Staging` to `Production` .

`code-push promote MyApp Staging Production`

4. Rolling Back Updates:

We can't delete or remove an update once it has been released. But we can easily roll back to prior version if current update contains bug or crash.

```
code-push rollback MyApp Production — targetRelease v3
```

5. Debugging CodePush Integration

Using this we can easily diagnose the behavior of the plugin like issues, update experience. Please ensure the app is running on device and must be connected with CLI while debugging.

```
code-push debug android
```

Store Guideline Compliance

- Google Play and internally distributed apps (e.g. Enterprise, Fabric) have no limitations .
- But iOS App Store and its corresponding guidelines have more precise rules. Here is the notes Official Apple Developer License Agreement.

> `Interpreted code may be downloaded to an Application but only so long as such code: (a) does not change the primary purpose of the Application by providing features or functionality that are inconsistent with the intended and advertised purpose of the Application as submitted to the App Store, (b) does not create a store or storefront for other code or applications, and © does not bypass signing, sandbox, or other security features of the OS`.

Again, It's important. If your update needs recompilation or rebuild of app, it must be delivered through respective stores.

That's all. Finally our fight with mobile app updates comes to end. Now we are on the way to implement stuff like Continuous Integration, Multi-Level app deployment etc. So we'll hit back again with our experience.

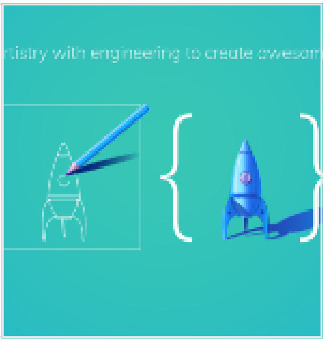
About Us:

We are people from **Sprite** and we ❤ to build products using Rails, Node JS, React Native etc .

Spritle software - Awesome Web and Mobile solutions

A software company focused on software design and delivery. We provide professional services and leading thought on...

www.spritle.com



React Native | Code Push | React | Android | iOS