

Speak React Native

React Native course by **U+₋**

Overview

- Build process
- Signing app package
- Distributing to testers
- Deploying to production
- Building with Expo
- Screen orientation

Build process

Building Android debug package

```
$ cd android
```

```
$ ./gradlew assembleDebug
```

will generate output file:

/android/app/build/outputs/apk/debug/app-debug.apk

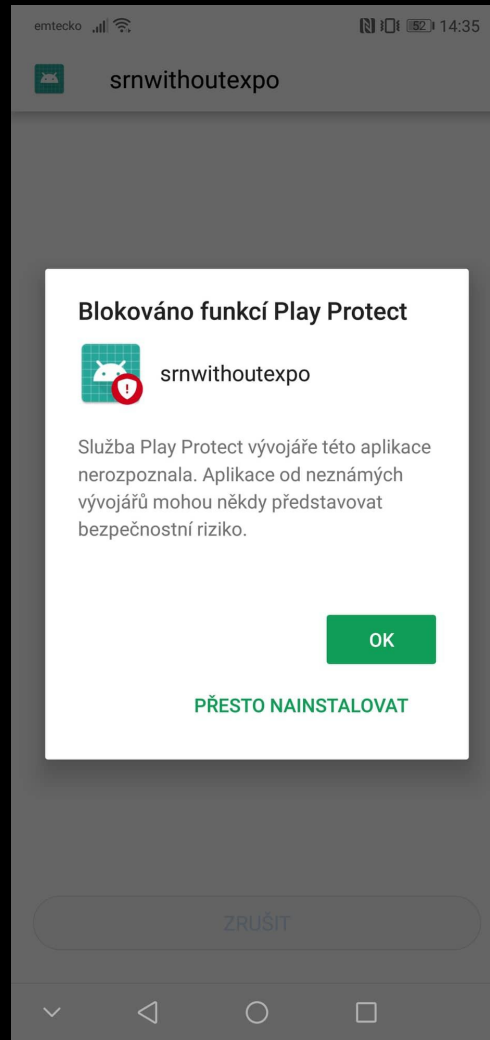
- Automatically signed with a debug key

app-debug.apk

- What if I send the apk to a friend to install it?

app-debug.apk

- What if I send the apk to a friend to install it?



app-debug.apk

- What if I send the apk to a friend to install it?



JS bundle

- Assemble JS bundle for standalone test apk

```
react-native bundle --platform android --dev
false --entry-file index.js --bundle-output
android/app/src/main/assets/index.android.bundl
e --assets-dest android/app/src/main/res/
```

- Building debug apk with `./gradlew assembleDebug` should work after this step

Building Android release package

- First, increase **version code** and **version number**
- Build the apk

```
$ cd android
```

```
$ ./gradlew assembleRelease
```

will generate output file:

```
/android/app/build/outputs/apk/release/app-release-unsigned.apk
```

app-release-unsigned.apk

- What if I send the apk to a friend to install it?

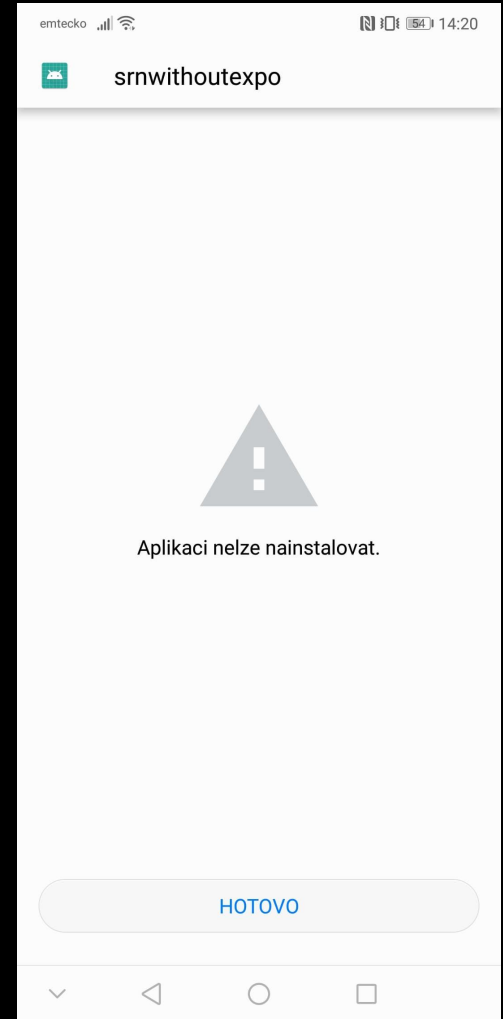
app-release-unsigned.apk

- What if I send the apk to a friend to install it?



app-release-unsigned.apk

- What if I send the apk to a friend to install it?
- Must be signed with your private key!



Keystore

- For signing apps, we need a keystore file which contains a private key
- Generate keystore and **make a backup** of it
- If you lose it, you won't be able to upload any updates

```
$ keytool -genkey -v -keystore my-release-key.keystore  
-alias alias_name -keyalg RSA -keysize 2048 -validity  
10000
```

Signing

- Sign app

```
$ jarsigner -sigalg SHA1withRSA -digestalg SHA1 -keystore  
my-release-key.keystore -storepass KEYSTORE_PASSWORD  
android/app/build/outputs/apk/release/app-release-unsigned.apk alias_name
```

- Verify signed app

```
$ jarsigner --verify app-release-unsigned.apk
```

Zipalign

- Optimization to consume less RAM

```
$ zipalign -v 4  
android/app/build/outputs/apk/release/app-release-unsigned.apk app-release.apk
```

- If you use Apksigner, zipalign must be performed **before** signing the app
- All of the previous steps can also be done using Android studio UI

Split APKs to reduce file size

- android/app/build.gradle
 - ndk {
 - abiFilters "armeabi-v7a", "x86"
 - }
 - def enableSeparateBuildPerCPUArchitecture = false
 - + def enableSeparateBuildPerCPUArchitecture = true

Split APKs to reduce file size

- android/app/build.gradle
 - ndk {
 - abiFilters "armeabi-v7a", "x86"
 - }
 - def enableSeparateBuildPerCPUArchitecture = false
 - + def enableSeparateBuildPerCPUArchitecture = true
- Output files `app-armeabi-v7a-release-unsigned.apk` (4.6 MB) and `app-x86-release-unsigned.apk` (5.8 MB)

Split APKs to reduce file size

- android/app/build.gradle
 - ndk {
 - abiFilters "armeabi-v7a", "x86"
 - }
 - def enableSeparateBuildPerCPUArchitecture = false
 - + def enableSeparateBuildPerCPUArchitecture = true
- Output files `app-armeabi-v7a-release-unsigned.apk` (4.6 MB) and `app-x86-release-unsigned.apk` (5.8 MB)
- `App-release-unsigned.apk` had 7.9 MB

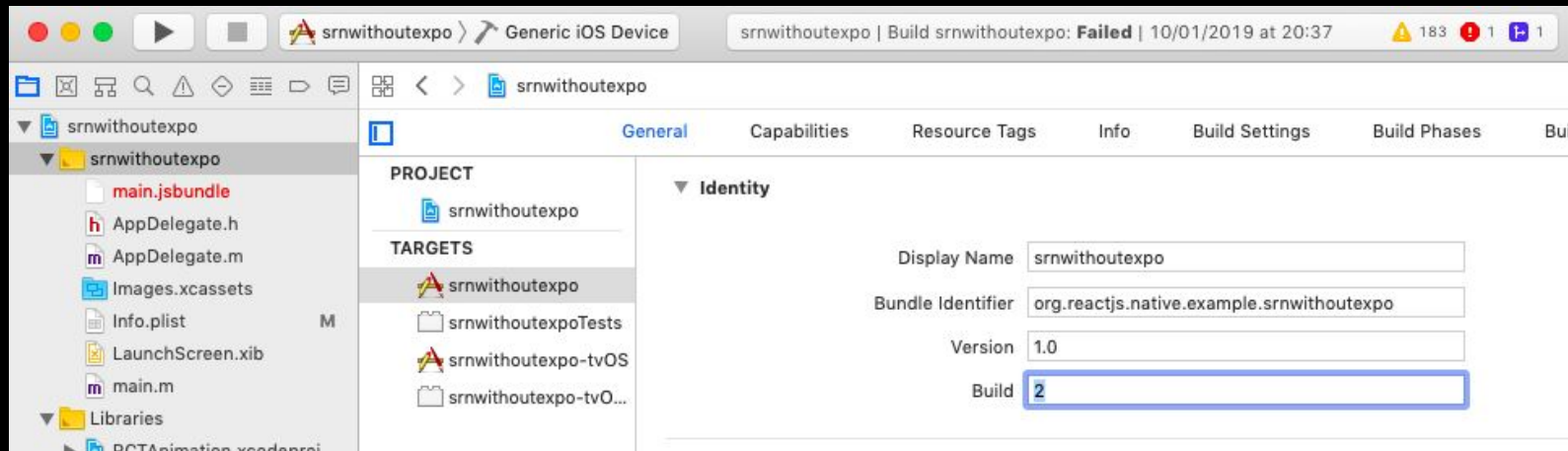
iOS account

- Developer account
 - \$99 / year
 - \$299 / year enterprise
- Running in iPhone without developer account

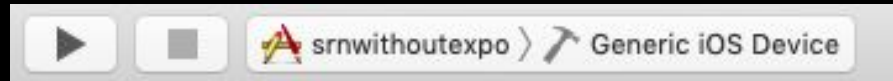
<https://blog.ionicframework.com/deploying-to-a-device-without-an-apple-developer-account/>

iOS build

- Increase build number



- Generic iOS Device
- Product -> Archive



Bitrise.io

- Cloud platform for building mobile apps
- Supports building, signing apps, uploading to stores automatically
- Supports git hooks - automatic builds on merge
- Has a friendly UI - also non-dev people can trigger new builds
- Free trial version

Distributing to testers

Local APK package

- Install test version to connected physical device

```
$ cd android
```

```
$ ./gradlew installDebug
```

HockeyApp

- A platform for distribution of APK files
- Similar to Testflight
- We can upload APKs to HockeyApp and send it to testers without having Play store app set up

Publishing APK

- <https://play.google.com/apps/publish>
- 25 USD
- Version must be increased before uploading to store
- We can use Alpha or Beta testing features directly in Play store

Adding Android testers

- In Play console, create Closed track testing channel
- Create user list
- Add users by email

Manage testersClosed test using testers lists ^

Choose how to run your testing program. [Learn more](#)

Choose a testing method

Closed test using testers lists ▾

Users

CREATE LIST

After you create a list, you can reuse the list for Closed Testing with any of your published apps.

Active	List name	Number of users	
<input type="checkbox"/>	prosté seznam	3 testers	EDIT

Feedback Channel ?

Email address or URL

Opt-in URL

An opt-in link will be available here when you publish your app.

DEACTIVATE TRACK

SAVED

iOS - TestFlight


- Platform for test app versions distribution
- We can invite people by their Apple IDs or create public link
- Every tester needs to download Testflight app, they can start testing after accepting the invite
- For external testers, the build must be approved by Apple

Xcode: Upload build to TestFlight

- Open Organizer (Window -> Organizer)
 - Opens automatically after archive
- Select built to upload
- Distribute App



Inviting tester (iOS)

Testers (8) 			<input type="text" value="Search"/>	
Email	Name	Status ^		
marek[REDACTED]@gmail.com	Marek [REDACTED]	Installed 1.0 (8) January 18, 2019		
an[REDACTED]@icloud.com	Anna [REDACTED]	Installed 1.0 (5) January 9, 2019		
jan[REDACTED]@gmail.com	Jan [REDACTED]	Installed 1.0 (7) January 14, 2019		

Public Link

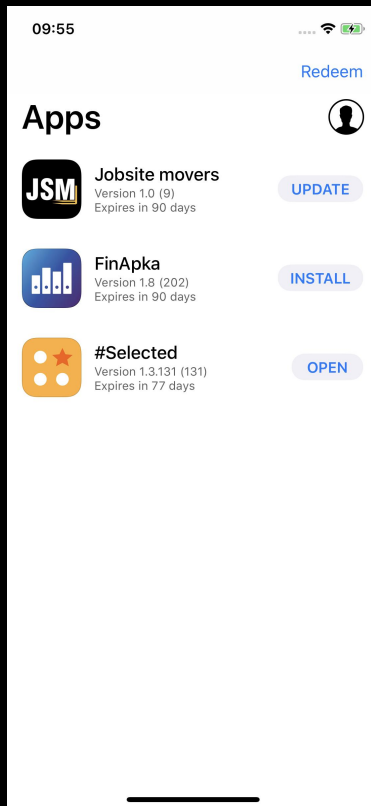
<https://testflight.apple.com/join/pBOAaPIR> [Disable Link](#) | [Copy Link](#)

Tester Count

0 [Set limit](#)

Installing app from TestFlight (iOS)

TestFlight app =>



Deploying to production

Deploying to production - Android

- Very similar to deploying Alpha or Beta versions
- App description and screenshots need to be filled in, as well as links to Terms and Conditions
- We can either promote a Beta release to a production release or upload a new production build

Release rollout percentage

- We can make the release rollout gradually (i.e. not for all users at once)
- If there is a production issue, it won't affect all users

Release rollout percentage

Specify the percentage of your user base that you want to rollout this release to.

Installs on active devices	Rollout percentage	Installs targeted by rollout
16	100%	16

Rollout countries ▼

Deploying to production - iOS

- Each build must be reviewed by Apple (takes about 2 days)
- You must provide credentials if login is required
- We can also release a version previously uploaded to Testflight

App Review Information

Sign-In Information ?

Provide a user name and password so we can sign in to your app. We'll need this to complete your app review.

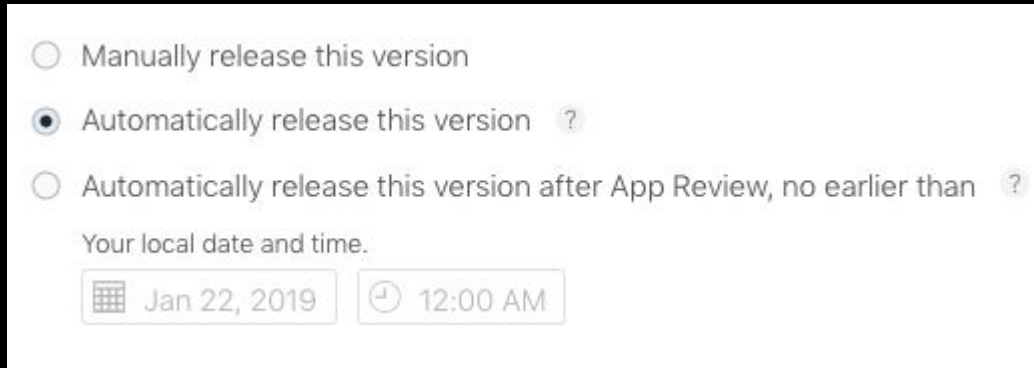
☒ Sign-in required

test@email.cz

ApplePassword123

Deploying to production - iOS

- If we choose manual release, we should be aware that the app takes some time to actually appear in the store, even after being approved by Apple



A screenshot of the Xcode release options dialog. It features three radio button options for releasing the app version. The first option is 'Manually release this version'. The second option, 'Automatically release this version', is selected with a filled radio button and includes a help icon. The third option is 'Automatically release this version after App Review, no earlier than', also with a help icon. Below these options is the text 'Your local date and time.' followed by two input fields: a date field showing 'Jan 22, 2019' and a time field showing '12:00 AM'.

☐ Manually release this version

☒ Automatically release this version ?

☐ Automatically release this version after App Review, no earlier than ?

Your local date and time.

Deployment - good practices

- Keep a checklist of things to go through before every release
- Test every build before putting it to production
- Keep track about changes made since the last update, add a changelog

Building with Expo

- Expo has a cloud service for building standalone apps developed in Expo environment
- We only need to add a some settings into our `app.json` file
- We can customize bundle id, icon, splashscreen and many other things, complete list can be found in [Expo docs](#)

Building with Expo

- We do not need to worry about things like android keystore, Expo can generate that for us (but lets us use our own if we want)
- Still, if we want to use some native library unsupported by Expo, we will have to do `expo eject` and build manually

How to build with Expo

- We need to do this once
 - Configure bundle id, app name etc. in app.json
 - `$ exp login`
- To perform each build
 - `$ exp start`
 - `$ exp build:android` or `exp build:ios`
- Then choose own keystore or generate new one (Android build) or login with AppleID (iOS build)
- You will then get URL to download APK/IPA package

Screen orientation

Screen orientation

- AndroidManifest.xml

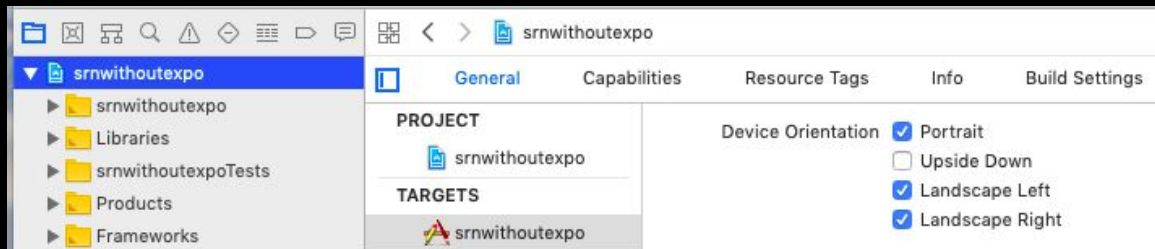
```
<activity  
    ...  
    android:name=".MainActivity"  
    android:label="@string/app_name"  
    android:screenOrientation="portrait">
```

Screen orientation

- AndroidManifest.xml

```
<activity  
    ...  
    android:name=".MainActivity"  
    android:label="@string/app_name"  
    android:screenOrientation="portrait">
```

- iOS project



Screen orientation

- <https://github.com/yamill/react-native-orientation>
 - External lib to change orientation programmatically
 - Also supports listeners to orientation changes

```
Orientation.lockToPortrait()
```

```
Orientation.addOrientationListener(this.handleOrientationChange)
```

```
// etc...
```

Questions?

Projects



Sources

- <https://facebook.github.io/react-native/docs/signed-apk-android>
- <https://docs.expo.io/versions/v32.0.0/workflow/configuration/>