

Deployment Guide

Emmanuel BRUNO
Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

2026-10-14

Table des matières

1	Deployment Guide	1
1.1	Deployment Overview	1
1.2	Prerequisites	1
1.3	Deployment Process	1
1.3.a	1. Build Preparation	1
1.3.b	2. Testing & Validation	2
1.3.c	3. Release Candidate Creation .	2
1.3.d	4. Store Submission	3
1.3.e	5. Staged Rollout	3
1.3.f	6. Post-Deployment Monitoring	3
1.3.g	6.4 Analytics Review	3
1.4	Rollback Procedure	3
1.4.a	1. Assessment	3
1.4.b	2. Decision	3
1.4.c	3. Rollback Execution	3
1.4.d	4. Communication	3
1.5	Deployment Checklist	4
1.6	Appendix: Continuous Deployment .	4

1 Deployment Guide

This document outlines the process for deploying the Fitness Tracker application to production environments.

1.1 Deployment Overview

Our deployment process consists of these key phases:

1. Build Preparation
2. Testing & Validation

3. Release Candidate Creation
4. Store Submission
5. Staged Rollout
6. Post-Deployment Monitoring

1.2 Prerequisites

Before starting the deployment process, ensure:

- All features for the release are complete and tested
- Release branch has been created (`release/vX.Y.Z`)
- Release notes are finalized and approved
- Legal review completed (if applicable)
- Marketing materials prepared (if applicable)

1.3 Deployment Process

1.3.a 1. Build Preparation

1.3.a.a 1.1 Version Update

```
// Android: build.gradle.kts
android {
    defaultConfig {
        versionCode =
[NEW_VERSION_CODE]
        versionName =
"[NEW_VERSION_NAME]"
    }
}
```

1.3.a.b 1.2 Environment Configuration

Ensure the build is configured to use production environments:

```
// BuildConfig.kt
object BuildConfig {
    const val API_BASE_URL = "https://
api.fitnesstracker.com/v1/"
    const val ANALYTICS_ENABLED = true
}
```

1.3.a.c 1.3 Feature Flag Verification

Review all feature flags to ensure they're appropriately set for production:

```
// FeatureFlags.kt
object FeatureFlags {
    const val ENABLE_SOCIAL_SHARING
= true
                                const val
ENABLE_ADVANCED_ANALYTICS = true
                                const val ENABLE_BETA_FEATURES
= false
}
```

1.3.b 2. Testing & Validation

1.3.b.a 2.1 Final QA Testing

Run through the complete Release Testing Checklist which includes:

- Regression testing
- Performance testing
- Security validation
- Localization verification
- Accessibility compliance

1.3.b.b 2.2 Clean Build Verification

Generate a clean build and verify:

```
# Android
./gradlew clean assembleRelease
bundleRelease
```

```
# iOS
xcodebuild clean archive -workspace
FitnessTracker.xcworkspace \
    -scheme FitnessTracker -
configuration Release \
    -archivePath build/
FitnessTracker.xcarchive
```

1.3.b.c 2.3 Signing and Validation

Ensure builds are properly signed:

- Android: Verify APK/AAB is signed with release keystore
- iOS: Verify app is signed with distribution certificate

1.3.c 3. Release Candidate Creation

1.3.c.a 3.1 Create Release Build

Generate the final release build(s):

```
# Android
./gradlew bundleRelease

# iOS
xcodebuild
-exportArchive -archivePath build/
FitnessTracker.xcarchive \
    -exportOptionsPlist
exportOptions.plist \
    -exportPath build/
FitnessTracker.ipa
```

1.3.c.b 3.2 Tag Release in Git

```
git tag -a v1.2.0 -m "Release v1.2.0"
git push origin v1.2.0
```

1.3.c.c 3.3 Archive Artifacts

Upload build artifacts to secure storage with appropriate versioning.

1.3.d 4. Store Submission

1.3.d.a 4.1 Google Play Store

1. Log in to Google Play Console
2. Navigate to «Production» track
3. Upload the signed App Bundle (.aab)
4. Complete store listing updates
5. Submit for review

1.3.d.b 4.2 Apple App Store

1. Log in to App Store Connect
2. Navigate to «My Apps» > «Fitness Tracker»
3. Create a new version with the appropriate version number
4. Upload build via Xcode or Transporter
5. Complete metadata and submit for review

1.3.e 5. Staged Rollout

1.3.e.a 5.1 Android Phased Release

Configure staged rollout in Google Play Console: - Initial rollout: 10% of users - Monitor for 24-48 hours - If stable, increase to 50% - After 24-48 more hours, increase to 100%

1.3.e.b 5.2 iOS Phased Release

Enable phased release in App Store Connect: - AppStore automatically manages the phased rollout over 7 days - Monitor for issues throughout the phased release

1.3.f 6. Post-Deployment Monitoring

1.3.f.a 6.1 Crash Monitoring

Monitor crash reporting tools for: - New crash types - Increase in crash rates - ANR (Application Not Responding) events - Exceptions related to new features

1.3.f.b 6.2 Performance Monitoring

Track key performance indicators: - App startup time - Screen render time - Network request success rates - Database operation speeds

1.3.f.c 6.3 User Feedback Monitoring

Monitor user feedback from: - Store reviews - Support tickets - Social media mentions - In-app feedback

1.3.g 6.4 Analytics Review

Review analytics data for: - Feature adoption rates - User engagement metrics - Conversion rates - Session duration changes

1.4 Rollback Procedure

If critical issues are discovered, follow this rollback procedure:

1.4.a 1. Assessment

Gather information to understand: - Issue severity - User impact percentage - Whether a fix can be deployed quickly

1.4.b 2. Decision

Decide between: - Immediate rollback - Hotfix deployment - Continuing with patches in next release

1.4.c 3. Rollback Execution

If rollback is chosen:

1.4.c.a 3.1 Android Rollback

1. In Google Play Console, halt the staged rollout
2. Revert to the previous version
3. Create a user notification about the issue

1.4.c.b 3.2 iOS Rollback

1. In App Store Connect, remove the current version from sale
2. Submit an expedited review for the previous version
3. Create a user notification about the issue

1.4.d 4. Communication

1. Notify internal stakeholders
2. Update support team with talking points
3. Publish incident status (if high-impact)
4. Prepare post-mortem analysis

1.5 Deployment Checklist

Build Preparation

- [] Version numbers updated
- [] Production API endpoints configured
- [] Feature flags verified
- [] Localization files complete
- [] Proguard/R8 rules verified

Testing & Validation

- [] Regression test suite passed
- [] Performance benchmarks met
- [] Security scan completed
- [] Accessibility compliance verified
- [] Clean builds generated successfully
- [] Release builds signed with production keys

Release Candidate

- [] Final builds approved by QA
- [] Git tag created for release
- [] Build artifacts archived
- [] Release notes finalized

Store Submission

- [] App store listings updated
- [] Screenshots updated (if needed)
- [] Privacy policy updated (if needed)
- [] Release notes added to store listing
- [] Builds uploaded to both stores
- [] Submissions completed

Staged Rollout

- [] Initial percentage defined
- [] Monitoring dashboard prepared
- [] Support team notified
- [] Rollout schedule communicated

Post-Deployment

- [] Crash monitoring active
- [] Performance monitoring active
- [] User feedback channels monitored
- [] Analytics review scheduled

1.6 Appendix: Continuous Deployment

For future implementation, we're exploring automated deployment using: - GitHub Actions for CI/CD - Fastlane for build and deployment automation - Automated testing gates for deployment approvals