

Hotfix Process

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

2026-10-14

Table des matières

1	Hotfix Process	1
1.1	What Qualifies as a Hotfix	1
1.2	Hotfix Decision Flow	1
1.3	Roles and Responsibilities	2
1.4	Hotfix Process Steps	2
1.4.a	1. Issue Identification and Triage	2
1.4.b	2. Branch Management	2
1.4.c	3. Fix Implementation	2
1.4.d	4. Testing Process	2
1.4.e	5. Code Review	2
1.4.f	6. Version Update	2
1.4.g	7. Release Building	3
1.4.h	8. Expedited App Store Submission	3
1.4.i	9. Hotfix Merging	3
1.4.j	10. Post-Deployment Activities	3
1.5	Hotfix Communication Templates	3
1.5.a	Issue Acknowledgement	3
1.5.b	Hotfix Release	3
1.6	Post-Mortem Process	4
1.7	Hotfix Checklist	4
1.8	Continuous Improvement	5

1 Hotfix Process

This document outlines the process for implementing and deploying hotfixes to address critical issues in production releases of the Fitness Tracker application.

1.1 What Qualifies as a Hotfix

A hotfix is warranted when an issue meets one or more of these criteria:

- **Severity Level 1:** Complete app crash or non-functioning core feature
- **Severity Level 2:** Major feature works incorrectly, affecting many users
- **Security Vulnerability:** Issue that could compromise user data or security
- **Compliance Issue:** Problem that violates platform policies or regulations

1.2 Hotfix Decision Flow

```
flowchart TD
    A[Issue Reported] --> B{Severity Assessment}
    B -->|Critical| C[Initiate Hotfix]
    B -->|High| D{User Impact Analysis}
    D -->|Medium/Low| E[Schedule for Next Release]
    D -->|Widespread| C
    D -->|Limited| F{Workaround Available?}
    F -->|No| C
    F -->|Yes| G{Business Impact?}
    G -->|High| C
    G -->|Low| E
```

1.3 Roles and Responsibilities

Role	Responsibilities
Release Manager	Overall coordination, communication to stakeholders
Tech Lead	Technical assessment, solution approach approval
Developer	Issue analysis, fix implementation, testing
QA Engineer	Verification of fix, regression testing
Product Owner	Priority confirmation, approval of scope
Support Team	User communication, work-around documentation

1.4 Hotfix Process Steps

1.4.a 1. Issue Identification and Triage

- Issue reported and documented in tracking system
- Initial severity assessment by Tech Lead
- Decision to proceed as hotfix (using criteria above)
- Creation of hotfix ticket with highest priority

1.4.b 2. Branch Management

```
# Start from the tagged production release
git checkout v1.2.0

# Create a hotfix branch
git checkout -b hotfix/1.2.1

# Implement the fix
# ...

# Commit with detailed message referencing issue
```

```
git commit -m "Fix crash when creating custom exercises (fixes #456)"
```

1.4.c 3. Fix Implementation

- Developer implements minimal changes needed to address the issue
- No new features or non-critical fixes included
- Code follows standard practices with no shortcuts
- Unit tests added/updated to cover the issue

1.4.d 4. Testing Process

1.4.d.a 4.1 Specific Issue Testing

- Verify fix resolves the specific issue
- Test all edge cases related to the issue

1.4.d.b 4.2 Targeted Regression Testing

- Test related functionality that could be affected
- Focus on critical paths and related components

1.4.d.c 4.3 Compatibility Testing

- Verify fix on minimum supported OS versions
- Test on most common device configurations

1.4.e 5. Code Review

- Tech Lead performs expedited but thorough code review
- Emphasis on:
 - Fix correctness
 - Minimal scope of changes
 - Potential side effects
 - Test coverage

1.4.f 6. Version Update

```
// Android: build.gradle
android {
    defaultConfig {
        versionCode = [INCREMENTED]
```

```

        versionName = "1.2.1" //
Increment patch version
    }
}

```

1.4.g 7. Release Building

- Create signed builds following Deployment Guide
- Tag the release in git repository:

```

git tag -a v1.2.1 -m "Hotfix v1.2.1"
git push origin v1.2.1

```

1.4.h 8. Expedited App Store Submission

1.4.h.a 8.1 Google Play Store

- Request expedited review in Google Play Console
- Note that this is a critical bugfix in submission comments
- Select «Urgent update» option if available

1.4.h.b 8.2 Apple App Store

- Request expedited review in App Store Connect
- Provide clear explanation of the critical issue
- Reference any user impact in the request

1.4.i 9. Hotfix Merging

After successful deployment:

```

# Merge hotfix into main
git checkout main
git merge --no-ff hotfix/1.2.1 -m
"Merge hotfix 1.2.1"

# Also merge into development branch
git checkout develop
git merge --no-ff hotfix/1.2.1 -m
"Merge hotfix 1.2.1"

```

1.4.j 10. Post-Deployment Activities

- Monitor crash reporting tools for issue resolution
- Update users via appropriate channels
- Document issue and resolution in knowledge base
- Schedule post-mortem if necessary

1.5 Hotfix Communication Templates

1.5.a Issue Acknowledgement

Subject: Important Notice: Known Issue in Fitness Tracker v1.2.0

Dear Fitness Tracker Users,

We've identified an issue in version 1.2.0 that may cause [brief description of issue].

Impact: [Describe who is affected and under what circumstances]

Workaround: [If available, provide temporary solution]

Status: Our development team is currently working on a fix that will be released as version 1.2.1 within [timeframe].

We apologize for any inconvenience and appreciate your patience.

The Fitness Tracker Team

1.5.b Hotfix Release

Subject: Fitness Tracker v1.2.1 Released - Important Fix

Dear Fitness Tracker Users,

We've just released version 1.2.1 which addresses [issue description].

What was fixed: [Specific details about the fix]

Action needed: Please update to the latest version as soon as possible.

The update is rolling out now and should be available to all users within 24 hours.

Thank you for your patience and understanding.

The Fitness Tracker Team

1.6 Post-Mortem Process

After each hotfix, we conduct a brief post-mortem to:

1. Understand root cause of the issue
2. Identify how it escaped detection before release
3. Improve processes to prevent similar issues
4. Document lessons learned

The post-mortem follows this template:

Hotfix Post-Mortem: v1.2.1

Issue Summary

- Description: [Brief description of the issue]
- Affected versions: [List of affected versions]
- User impact: [Scope and severity of impact]
- Time to resolution: [Hours/days from discovery to fix deployment]

Root Cause Analysis

- Technical cause: [What specifically caused the issue]
- Contributing factors: [Other factors that played a role]

Detection Analysis

- How was it detected: [User reports, monitoring, etc.]
- Why it wasn't caught earlier: [Gaps in testing, etc.]

Resolution

- Fix implemented: [Technical description of the solution]
- Verification method: [How the fix was verified]

Prevention Measures

- Process improvements: [Changes to prevent similar issues]
- Testing improvements: [Additional test cases or approaches]
- Monitoring improvements: [New alerts or metrics]

Timeline

- [Date/time] Issue reported
- [Date/time] Issue confirmed
- [Date/time] Hotfix development started
- [Date/time] Hotfix deployed

Action Items

- [] [Action item 1] - Owner: [Name], Due: [Date]
- [] [Action item 2] - Owner: [Name], Due: [Date]

1.7 Hotfix Checklist

Issue Qualification

- [] Issue severity confirmed as

```
critical
- [ ] User impact assessed
- [ ] Hotfix approval obtained from
Tech Lead and Product Owner

## Development
- [ ] Hotfix branch created from
production tag
- [ ] Issue reproduced in isolated
environment
- [ ] Fix implemented with minimal
scope
- [ ] Unit tests added/modified
- [ ] Manual testing completed

## Review & Testing
- [ ] Code review completed
- [ ] Fix verified to resolve the
specific issue
- [ ] Regression testing completed
- [ ] Performance impact assessed

## Release Preparation
- [ ] Version numbers incremented
appropriately
- [ ] Release notes prepared
- [ ] Signed builds created
- [ ] Git tag created

## Deployment
- [ ] Expedited review requested for
app stores
- [ ] Builds submitted to stores
- [ ] External communications
prepared

## Post-Deployment
- [ ] Hotfix merged to main branch
- [ ] Hotfix merged to development
branch
- [ ] Issue monitoring in place
- [ ] Post-mortem scheduled
```

1.8 Continuous Improvement

This hotfix process is reviewed after each hotfix deployment to identify opportunities for improvement in:

1. Development practices
2. Testing coverage
3. Deployment procedures
4. Communication flows
5. Monitoring capabilities

By treating each hotfix as a learning opportunity, we aim to reduce the frequency and impact of critical issues in production.