

# DevPipeline Mobile SDK Guide

**Document ID:** PRD-DP-035 **Last Updated:** 2024-02-15 **Owner:** DevPipeline Product Team **Classification:** Public

---

## Overview

DevPipeline Mobile SDK enables you to build, test, and deploy mobile applications for iOS and Android. This guide covers configuration, testing, and deployment workflows for mobile development.

---

## Supported Platforms

### iOS

Framework	Supported
Swift	5.0+
Objective-C	Full support
SwiftUI	Full support
UIKit	Full support
Xcode	14.0+

### Android

Framework	Supported
Kotlin	1.6+
Java	11+
Jetpack Compose	Full support
Android Views	Full support
Gradle	7.0+

### Cross-Platform

Framework	Supported
React Native	0.70+
Flutter	3.0+
Xamarin	Full support
Ionic/Capacitor	Full support

---

## Setup

### iOS Configuration

```
# pipeline.yaml
jobs:
  build-ios:
    runs-on: macos-latest
    steps:
      - checkout
      - name: Install dependencies
        run: |
          cd ios
          pod install
      - name: Build
        run: |
          xcodebuild -workspace MyApp.xcworkspace \
          -scheme MyApp \
          -configuration Release \
          -destination 'generic/platform=iOS' \
          -archivePath build/MyApp.xcarchive \
          archive
      - name: Export IPA
        run: |
          xcodebuild -exportArchive \
          -archivePath build/MyApp.xcarchive \
          -exportOptionsPlist ExportOptions.plist \
          -exportPath build/
```

### Android Configuration

```
# pipeline.yaml
jobs:
  build-android:
    runs-on: ubuntu-latest
```

```

steps:
  - checkout
  - name: Setup Java
    uses: setup-java
    with:
      java-version: 17
  - name: Build
    run: |
      cd android
      ./gradlew assembleRelease
  - name: Sign APK
    run: |
      jarsigner -keystore $KEYSTORE_PATH \
      -storepass $KEYSTORE_PASSWORD \
      app/build/outputs/apk/release/app-release-unsigned.apk \
      $KEY_ALIAS

```

## React Native Configuration

```

# pipeline.yaml
jobs:
  build-rn:
    strategy:
      matrix:
        platform: [ios, android]
    steps:
      - checkout
      - name: Install dependencies
        run: npm install
      - name: Build iOS
        if: matrix.platform == 'ios'
        runs-on: macos-latest
        run: |
          cd ios && pod install
          npx react-native build-ios --mode Release
      - name: Build Android
        if: matrix.platform == 'android'
        run: |
          cd android
          ./gradlew assembleRelease

```

## Flutter Configuration

```

# pipeline.yaml
jobs:

```

```

build-flutter:
  strategy:
    matrix:
      platform: [ios, android]
  steps:
    - checkout
    - name: Setup Flutter
      uses: setup-flutter
      with:
        flutter-version: 3.16.0
    - name: Install dependencies
      run: flutter pub get
    - name: Build iOS
      if: matrix.platform == 'ios'
      runs-on: macos-latest
      run: flutter build ios --release
    - name: Build Android
      if: matrix.platform == 'android'
      run: flutter build apk --release

```

---

## Code Signing

### iOS Signing

#### Automatic Signing

```

jobs:
  build-ios:
    env:
      APPLE_ID: ${{ secrets.APPLE_ID }}
      APPLE_TEAM_ID: ${{ secrets.APPLE_TEAM_ID }}
    steps:
      - name: Install certificate
        run: |
          devpipeline mobile ios install-certificate \
            --certificate ${{ secrets.IOS_CERTIFICATE }} \
            --password ${{ secrets.CERT_PASSWORD }}
      - name: Install provisioning profile
        run: |
          devpipeline mobile ios install-profile \
            --profile ${{ secrets.PROVISIONING_PROFILE }}

```

## Match (Fastlane)

```
jobs:
  build-ios:
    steps:
      - name: Setup signing
        run: |
          bundle exec fastlane match appstore --readonly
env:
  MATCH_PASSWORD: ${{ secrets.MATCH_PASSWORD }}
  MATCH_GIT_URL: ${{ secrets.MATCH_GIT_URL }}
```

## Android Signing

```
jobs:
  build-android:
    steps:
      - name: Decode keystore
        run: |
          echo ${{ secrets.KEYSTORE_BASE64 }} | base64 -d > keystore.jks
      - name: Build signed APK
        run: |
          ./gradlew assembleRelease \
            -Pandroid.injected.signing.store.file=keystore.jks \
            -Pandroid.injected.signing.store.password=${{ secrets.KEYSTORE_PASSWORD }} \
            -Pandroid.injected.signing.key.alias=${{ secrets.KEY_ALIAS }} \
            -Pandroid.injected.signing.key.password=${{ secrets.KEY_PASSWORD }}
```

---

## Testing

### iOS Testing

```
jobs:
  test-ios:
    runs-on: macos-latest
    steps:
      - checkout
      - name: Run unit tests
        run: |
          xcodebuild test \
            -workspace MyApp.xcworkspace \
            -scheme MyApp \
```

```

        -destination 'platform=iOS Simulator,name=iPhone 15'
- name: Run UI tests
  run: |
    xcodebuild test \
      -workspace MyApp.xcworkspace \
      -scheme MyAppUITests \
      -destination 'platform=iOS Simulator,name=iPhone 15'

```

## Android Testing

```

jobs:
  test-android:
    steps:
      - name: Run unit tests
        run: ./gradlew test
      - name: Run instrumented tests
        uses: android-emulator
        with:
          api-level: 33
          script: ./gradlew connectedAndroidTest

```

## Device Testing

### Firebase Test Lab

```

jobs:
  device-tests:
    steps:
      - name: Build test APK
        run: ./gradlew assembleDebug assembleAndroidTest
      - name: Run on Firebase Test Lab
        run: |
          gcloud firebase test android run \
            --type instrumentation \
            --app app/build/outputs/apk/debug/app-debug.apk \
            --test app/build/outputs/apk/androidTest/debug/app-debug-androidTest.apk \
            --device model=Pixel6,version=33

```

### BrowserStack

```

jobs:
  device-tests:
    steps:

```

```

- name: Upload to BrowserStack
  run: |
    curl -u "$BROWSERSTACK_USER:$BROWSERSTACK_KEY" \
      -X POST https://api-cloud.browserstack.com/app-automate/upload \
      -F "file=@app-release.apk"
- name: Run tests
  run: npx browserstack-runner

```

---

## Deployment

### App Store (iOS)

```

jobs:
  deploy-ios:
    needs: [build-ios, test-ios]
    steps:
      - name: Upload to App Store Connect
        run: |
          xcrun altool --upload-app \
            --type ios \
            --file build/MyApp.ipa \
            --username ${{ secrets.APPLE_ID }} \
            --password ${{ secrets.APP_SPECIFIC_PASSWORD }}

```

### Using Fastlane

```

jobs:
  deploy-ios:
    steps:
      - name: Deploy to TestFlight
        run: bundle exec fastlane ios beta
      - name: Deploy to App Store
        run: bundle exec fastlane ios release

```

### Google Play (Android)

```

jobs:
  deploy-android:
    needs: [build-android, test-android]
    steps:
      - name: Upload to Play Store

```

```

uses: upload-google-play
with:
  service-account-json: ${{ secrets.GOOGLE_PLAY_KEY }}
  package-name: com.novatech.myapp
  release-file: app/build/outputs/bundle/release/app-release.aab
  track: internal # internal, alpha, beta, production

```

## Using Fastlane

```

jobs:
  deploy-android:
    steps:
      - name: Deploy to internal track
        run: bundle exec fastlane android internal
      - name: Promote to production
        run: bundle exec fastlane android production

```

## Over-the-Air (OTA) Updates

### CodePush (React Native)

```

jobs:
  ota-update:
    steps:
      - name: Deploy CodePush update
        run: |
          appcenter codepush release-react \
            -a MyOrg/MyApp-iOS \
            -d Production \
            --description "Bug fixes"

```

---

## Version Management

### Automatic Versioning

```

jobs:
  build:
    steps:
      - name: Set version
        run: |
          VERSION=$(cat package.json | jq -r .version)

```

```

BUILD=$(echo $GITHUB_RUN_NUMBER)

# iOS
/usr/libexec/PlistBuddy -c "Set :CFBundleShortVersionString $VERSION" ios/MyApp/Info.plist
/usr/libexec/PlistBuddy -c "Set :CFBundleVersion $BUILD" ios/MyApp/Info.plist

# Android
sed -i "s/versionName \".*\"/versionName \"$VERSION\"/" android/app/build.gradle
sed -i "s/versionCode .*/versionCode $BUILD/" android/app/build.gradle

```

## Semantic Versioning

```

# Trigger on tags
on:
  push:
    tags:
      - 'v*'

jobs:
  release:
    steps:
      - name: Get version from tag
        run: echo "VERSION=${GITHUB_REF#refs/tags/v}" >> $GITHUB_ENV
      - name: Build and deploy
        run: ./scripts/release.sh $VERSION

```

---

## Caching

### iOS Caching

```

jobs:
  build-ios:
    steps:
      - name: Cache CocoaPods
        uses: cache
        with:
          key: pods-${{ hashFiles('ios/Podfile.lock') }}
          path: ios/Pods
      - name: Cache derived data
        uses: cache
        with:
          key: deriveddata-${{ hashFiles('ios/*.xcworkspace') }}
          path: ~/Library/Developer/Xcode/DerivedData

```

## Android Caching

```
jobs:
  build-android:
    steps:
      - name: Cache Gradle
        uses: cache
      with:
        key: gradle-${{ hashFiles('**/*.gradle*', 'gradle-wrapper.properties') }}
        path: |
          ~/.gradle/caches
          ~/.gradle/wrapper
```

---

## Notifications

### Slack Notifications

```
jobs:
  notify:
    needs: [deploy-ios, deploy-android]
    steps:
      - name: Notify Slack
        run: |
          curl -X POST ${{ secrets.SLACK_WEBHOOK }} \
            -H 'Content-type: application/json' \
            -d '{
              "text": "Mobile app v${{ env.VERSION }} deployed!",
              "blocks": [
                {
                  "type": "section",
                  "text": {
                    "type": "mrkdwn",
                    "text": "*Mobile App Released* :rocket:\nVersion: ${{ env.VERSION }}\n"
                  }
                }
              ]
            }'
```

---

## Best Practices

### Build Optimization

1. **Cache dependencies** - CocoaPods, Gradle, npm
2. **Parallel builds** - Run iOS and Android in parallel
3. **Incremental builds** - Use build caching
4. **Resource optimization** - Right-size runners

### Security

1. **Secure credentials** - Use secrets management
2. **Rotate signing keys** - Regular rotation
3. **Review permissions** - Minimal app permissions
4. **Security scanning** - Include in pipeline

### Quality

1. **Automated testing** - Unit, integration, E2E
  2. **Device testing** - Test on real devices
  3. **Performance testing** - Monitor app performance
  4. **Accessibility testing** - Include a11y checks
- 

## Troubleshooting

### iOS Build Failures

```
# Clean build
xcodebuild clean

# Clear derived data
rm -rf ~/Library/Developer/Xcode/DerivedData

# Reset pods
cd ios && pod deintegrate && pod install
```

### Android Build Failures

```
# Clean build
./gradlew clean
```

```
# Clear gradle cache  
rm -rf ~/.gradle/caches  
  
# Invalidate caches  
./gradlew --stop
```

## **Signing Issues**

- Verify certificate not expired
  - Check provisioning profile includes device
  - Ensure bundle ID matches
  - Verify keystore password
- 

## **API Reference**

### **Mobile Commands**

Command	Description
<code>devpipeline mobile ios build</code>	Build iOS app
<code>devpipeline mobile android build</code>	Build Android app
<code>devpipeline mobile ios test</code>	Run iOS tests
<code>devpipeline mobile android test</code>	Run Android tests
<code>devpipeline mobile deploy</code>	Deploy to stores

---

*Related Documents: Getting Started (PRD-DP-001), Pipeline Configuration (PRD-DP-005), Parallel Builds (PRD-DP-030)*