DevPipeline Mobile Build Support - Project Update

Project: Mobile SDK and Build Support **Update #:** 5 **Date:** July 28, 2024
Project Lead: Emma Thompson **Status:** On Track

Executive Summary

The DevPipeline mobile build support project is progressing well. We've completed iOS build support and are finalizing Android. Beta launch scheduled for August 15, 2024.

Project Overview

Objective

Enable mobile app development teams to build, test, and deploy iOS and Android applications using DevPipeline.

Goals

1. Native iOS build support (Xcode, Swift, Objective-C)
2. Native Android build support (Gradle, Kotlin, Java)
3. Cross-platform framework support (React Native, Flutter)
4. App signing and distribution integration
5. Device testing integration (Firebase Test Lab, BrowserStack)

Timeline

Phase	Description	Status	Date
Phase 1	iOS Build Support	Complete	Jun 30
Phase 2	Android Build Support	In Progress	Aug 5
Phase 3	Cross-platform	In Progress	Aug 10

Phase	Description	Status	Date
Phase 4	Testing Integrations	Planned	Aug 15
Phase 5	Beta Launch	Planned	Aug 15
Phase 6	GA Release	Planned	Sep 15

Phase 1 Complete: iOS Build Support

Delivered Features

macOS Runners: - Dedicated macOS runners with Xcode 15.x - Multiple Xcode version support (14.3, 15.0, 15.1) - Automatic dependency management (CocoaPods, SPM)

Build Capabilities: - Full Xcode build support - Unit test and UI test execution - Code signing integration - Provisioning profile management - IPA generation

Distribution: - TestFlight upload integration - App Store Connect API support - Direct IPA artifact storage

Example Configuration

```yaml
ios-build:
  runs-on: devpipeline-macos
  steps:
    - uses: checkout
    - uses: setup-xcode
      with:
        version: '15.1'
    - run: |
        pod install
        xcodebuild -workspace App.xcworkspace \
          -scheme App \
          -configuration Release \
          -archivePath build/App.xcarchive \
          archive
    - uses: ios-signing
      with:
        certificate: ${{ secrets.IOS_CERTIFICATE }}
        provisioning_profile: ${{ secrets.PROVISIONING_PROFILE }}
    - uses: upload-testflight
```

```
      with:
        api_key: ${{ secrets.APP_STORE_CONNECT_KEY }}
```

Phase 2 Progress: Android Build Support

Completed (Week of Jul 22)

- ☒ Android SDK installation on Linux runners
- ☒ Gradle build support
- ☒ Unit test execution
- ☒ APK/AAB generation
- ☒ Keystore management

In Progress

- □ Instrumented test support (75%)
- □ Emulator integration (60%)
- □ Google Play Console upload (40%)

Android Example Configuration

```
android-build:
  runs-on: devpipeline-large
  steps:
    - uses: checkout
    - uses: setup-java
      with:
        version: '17'
    - uses: setup-android
      with:
        sdk-version: '34'
    - run: ./gradlew assembleRelease
    - uses: android-signing
      with:
        keystore: ${{ secrets.ANDROID_KEYSTORE }}
        key_alias: ${{ secrets.KEY_ALIAS }}
        key_password: ${{ secrets.KEY_PASSWORD }}
    - uses: upload-artifact
      with:
        path: app/build/outputs/apk/release/app-release.apk
```

Phase 3 Progress: Cross-Platform Support

React Native

☒ Basic build support
☒ Metro bundler integration
☐ Detox testing integration (in progress)

Flutter

☒ Flutter SDK setup action
☒ iOS and Android builds
☐ Flutter integration tests (planned)

Technical Decisions

macOS Runner Architecture

Decision	Choice	Rationale
Runner type	Dedicated VMs	iOS requires macOS hardware
Virtualization	AWS Mac instances	Cost-effective, scalable
Xcode versions	Last 3 major	Balance support and maintenance

Android Runner Architecture

Decision	Choice	Rationale
Runner type	Linux containers	Cost-effective
Emulator	Hardware acceleration	Required for speed
SDK management	Automatic updates	Security compliance

Risks and Mitigations

Risk: macOS Runner Capacity

Issue: Limited AWS Mac instance availability in some regions

Mitigation: - Pre-provisioned runner pool - Queue management for peak times - Option for self-hosted macOS runners

Status: Monitoring, capacity sufficient for beta

Risk Resolved: Code Signing Complexity

Issue: iOS code signing is notoriously complex

Resolution: - Automatic profile matching - Clear error messages - Step-by-step documentation - Migration guide from other CI systems

---

Customer Feedback (Alpha)

5 customers participated in alpha testing:

Positive: - “Finally, a CI that handles iOS signing properly” - “Build times competitive with other solutions” - “Integration with existing pipelines seamless”

Improvement requests: - More Xcode version options - Faster macOS runner startup - Better caching for CocoaPods

---

Metrics

Build Performance

Platform	Avg Build Time	Cache Hit Rate
iOS	8.5 min	72%
Android	6.2 min	78%
React Native	12.3 min	65%

Reliability

Metric	Target	Current
Build success rate	>95%	94.2%
Runner availability	>99%	99.8%

Metric	Target	Current
Artifact upload success	>99%	99.5%

Resource Status

Team

Role	Allocated	Status
Engineers	4	On track
QA	1	On track
Tech Writer	0.5	Documentation in progress

Budget

Category	Budgeted	Spent	Status
Infrastructure	$30,000	$22,000	Under
Device Testing	$10,000	$8,000	Under
Total	$40,000	$30,000	On track

Next Steps

This Week (Jul 29 - Aug 2)

1. Complete Android instrumented tests
2. Finish emulator integration
3. Begin Google Play upload feature

Next Week (Aug 5 - Aug 9)

1. Complete Android GA features
2. Flutter integration tests
3. Documentation review

Beta Launch (Aug 15)

1. Public beta announcement
2. Onboard beta customers
3. Feedback collection process

Dependencies

Dependency	Status	Impact
Marketing launch plan	Ready	None
Docs complete	In progress	None (on track)
Support training	Scheduled	Aug 12
Pricing approved	Approved	None

Questions for Stakeholders

1. **Pricing confirmation:** Confirm macOS runner pricing (currently $0.08/min)
2. **Beta scope:** How many beta customers to target? (Proposing 25)
3. **GA timeline:** Confirm Sep 15 GA is still target

Next Update: August 5, 2024