GitLab CI/CD Pipeline for Android Application

BUILD, TEST, AND DEPLOY USING DOCKER, GRADLE & FASTLANE

Overview

- CI/CD pipeline with 7 stages
- Uses Docker, Gradle, and Fastlane
- Includes build, test, and release tracks
- Emphasizes versioning, caching, and secure deployment

Pipeline Stages

- 1. Environment – Set up Docker container
- 2. Build – Compile the Android app
- 3. Test – Run unit/instrumentation tests
- 4. Internal – Deploy to internal testers
- 5. Alpha – Promote to alpha channel
- 6. Beta – Promote to beta channel
- 7. Production – Final release to users

Docker Container Management

- .updateContainerJob template:
- - Authenticates with registry
- - Pulls cached image
- - Builds and pushes updated image
- updateContainer: Triggers only if Dockerfile changes
- ensureContainer: Ensures container availability (skipped if Dockerfile changed)

Build Jobs

- .build_job template:
- - Makes gradlew executable
- - Sets versionCode (pipeline ID) & SHA
- - Stores APKs as artifacts
- buildDebug: Uses Fastlane for debug builds
- buildRelease: Prepares release APK

Testing

- testDebug:
- - Runs unit tests on debug build
- - Validates code quality early

Deployment and Promotion

- .promote_job template:
- - Requires manual approval
- - Uses Google Play API
- - Cleans credentials after use
- publishInternal – Internal track
- promoteAlpha – Internal → Alpha
- promoteBeta – Alpha → Beta
- promoteProduction – Beta → Production (master only)

Key Features

- Docker caching for speed
- Artifact passing between stages
- Manual promotion approvals
- Secure, staged releases
- Version control with pipeline ID & Git SHA
- Production deployment restricted to master branch

Summary

- Robust, scalable CI/CD for Android
- Efficient Docker usage
- Controlled and secure release process
- Extensible for future stages (e.g., UI tests)

Understanding Gemfile in Android CI/CD with Fastlane

WHY RUBY AND FASTLANE MATTER IN ANDROID AUTOMATION

What is a Gemfile?

- A file used to declare Ruby gem dependencies for a project
- Commonly used with the Bundler tool
- Ensures consistent gem versions across environments
- Key for managing Fastlane in CI environments

What is a Ruby Gem?

- A packaged Ruby library or command-line tool
- Examples: fastlane, cocoapods, jekyll
- Distributed via https://rubygems.org
- Installed using `gem install` or `bundle install`

Why Use a Gemfile in Android Projects?

- Android uses Gradle, but Fastlane is a Ruby tool
- Fastlane automates builds, versioning, and deployment
- Fastlane must be installed using RubyGems
- Gemfile ensures Fastlane is consistently installed in CI/CD

Gemfile Example

- source "https://rubygems.org"
- gem "fastlane"

- This declares the source and the dependency on Fastlane

Using Gemfile in GitLab CI

- CI job example:
- - bundle install # installs Fastlane from Gemfile
- - bundle exec fastlane buildRelease
- Ensures CI runners use the same Fastlane version

Summary

- Fastlane is a Ruby tool used in Android CI/CD for deployment
- A Gemfile is required to install Fastlane consistently
- Bundler + Gemfile is the standard way to manage Ruby dependencies
- Important for team consistency and CI reliability

What is Fastlane?

AUTOMATE YOUR MOBILE APP DEVELOPMENT PIPELINE

Key Features of Fastlane

- Build Automation – Automate Android/iOS builds
- Code Signing – Handle signing certificates and profiles
- Test Automation – Run unit and UI tests
- Deployment – Publish apps to Google Play and App Store
- Versioning – Manage app versions and changelogs
- Screenshots – Capture and upload localized screenshots

Common Fastlane Commands (Lanes)

- fastlane beta – Deploy to beta testers
- fastlane release – Push app to the store
- fastlane screenshots – Capture and upload screenshots
- fastlane test – Run tests on your app

Why Use Fastlane?

- Saves time with automation
- Minimizes manual release errors
- Supports both Android and iOS
- Easy integration with CI/CD tools like GitLab CI, GitHub Actions, Jenkins

Fastlane Example (Android Fastfile)

```
default_platform(:android)
platform :android do
  lane :beta do
    gradle(task: 'assembleRelease')
    upload_to_play_store(track: 'beta')
  end
end
```