Applications of Continuous Integration in Mobile Development: Post-print

Authors: Cai Guohua, Wei Lili, Han Xiaobo

Date: 2023-10-08T00:00:00+00:00

Abstract

This paper elaborates on how to address the characteristics and requirements of rapid iterative development and multi-channel continuous release for mobile clients by designing a process-oriented solution using the Jenkins continuous integration tool platform. This solution automates a large number of repetitive and cumbersome manual operations, covering code inspection, building, testing, hardening, signing, and deployment. By leveraging automated processes, it assists development teams in increasing release frequency, rapidly identifying and correcting issues, thereby improving work efficiency and shortening version release cycles.

Full Text

Preamble

Title: Application of Continuous Integration in Mobile Development

Abstract: This paper elaborates on how to address the characteristics and demands of rapid iterative development and multi-channel continuous release for mobile clients. By leveraging the Jenkins continuous integration tool platform, we designed a pipeline solution encompassing code inspection, building, testing, hardening, signing, and deployment. This automates numerous repetitive and tedious manual operations, helping development teams increase release frequency, quickly identify and correct issues, thereby improving work efficiency and shortening version release cycles.

Keywords: continuous integration; mobile development; Jenkins; automation

With the mobile internet thriving, mobile applications as the gateway to the internet have become fiercely contested battlegrounds for internet companies. This imposes higher requirements on the iterative development, testing, release,

and other processes of mobile applications. Beyond meeting basic functional requirements, they must also keep pace with market changes in terms of security, reliability, concurrent channel distribution, and information timeliness. China Search' s mobile application development adopts an agile development model. During multi-version iterative development, frequent activities such as code integration, compilation, testing, packaging, and release deployment are required. How to liberate developers from frequent manual labor, reduce errors caused by manual operations, accelerate iteration speed, and improve release quality has become an urgent problem to solve. Through research, selection, and practice, China Search has developed an automated continuous integration and release solution for mobile applications based on the Jenkins tool platform. This solution standardizes and optimizes numerous stages, integrates various heterogeneous tools and environments, and achieves automation from code compilation to release.

1.1 What is Continuous Integration

Continuous integration is a software development practice where development teams perform frequent system builds based on certain changes, such as continuous inspection, continuous compilation, continuous validation, continuous deployment, continuous infrastructure, and continuous reporting.

1.2 Why Continuous Integration is Needed

Through continuous integration, we can:

(1) **Liberate manpower and reduce manual operation error rates.** Continuous integration automates repetitive manual operations such as compilation, review, packaging, and deployment through tools.

(2) **Identify problems earlier.** By continuously integrating code, changes can be obtained earlier, testing can begin sooner, and problems can be discovered earlier, reducing the cost of problem resolution.

(3) **Enhance project visibility.** Continuous integration makes the software development process more transparent, and the continuous integration system can provide real-time build status and quality feedback.

(4) **Deliver results faster.** Continuous integration shortens the time for compilation, integration, testing, release, and deployment, enabling quicker delivery.

(5) **Achieve higher product quality.** Code inspection can be integrated into tools to check each code submission through continuous integration.

[Figure 1: see original paper] Typical Continuous Integration Scenario

1.3 Components of a Continuous Integration Build System

A complete continuous integration build system should include:

(1) **Code configuration management tools and appropriate version control strategies** to ensure code maintainability and traceability. This solution uses Subversion as the version repository.

(2) **A well-defined and orchestrated build integration process design**, including compilation, testing, distribution, and deployment.

(3) **A continuous integration service platform** to automate the execution of repetitive work and continuously implement the pre-defined build process. This solution uses Jenkins as the continuous integration service platform.

1.4 Selection of Continuous Integration Tool Platform

Jenkins, as an open-source software, is a Java-based continuous integration tool for monitoring continuous repetitive work. It has powerful built-in functions capable of completing common build tasks. Jenkins has a rich plugin library that can be customized according to business needs for building, testing, deployment, or other custom logic tasks. It can also perform static inspections and has been widely recognized and used in the industry.

2.2 Integrated System Overall Architecture

For China Search' s mobile application development model and actual requirements, the continuous integration platform integrates the Jira process management platform with the Jenkins automation platform. The Jira platform is used for project-related roles to submit project information and requirements and to manage workflow approval processes. The Jenkins tool platform adopts a master-slave structure that separates control logic from build actions. The Jenkins master node is mainly used for decomposing integration stages, configuring logic for each process step, and task distribution. Jenkins slave nodes are primarily responsible for executing distributed tasks and generating intermediate outputs. Given that the Android mobile client build environment involves numerous tool dependencies and complex configurations, the Android build slave node environment is containerized for easier replication, migration, and maintenance.

Multiple stakeholder roles are involved in the project release process:

- **Product managers**, responsible for submitting application information, code versions, application versions, channel distribution information, and descriptions of new features.
- **Testers**, responsible for functional and integration testing of generated applications.

- **Project leaders**, making final decisions on whether the product should go live and the timing of release.
- **Quality managers**, tracking the entire process, monitoring stage transitions, and overseeing standardized security supervision.

[Figure 2: see original paper] Continuous Integration Process Framework

China Search' s mobile product applications adopt an agile development model and have established a standard workflow divided into the following stages: coding → building → integration → testing → delivery → release. By leveraging tool platforms and automation, continuous integration, continuous delivery, and continuous deployment are implemented at different development stages.

Continuous Integration: During development, developers frequently merge working copies into the development trunk. Each merge is an integration process. Through continuous integration, code errors can be detected promptly, deviation from the main development line can be avoided, and high-quality rapid product iteration can be achieved.

Continuous Delivery: Through continuous delivery, new versions can be frequently delivered to testing teams for review. Once testing is approved, the code can enter the release stage.

Continuous Deployment: Once testing is approved, applications can be automatically deployed to the online production environment through continuous deployment, enabling release anytime and anywhere.

[Figure 3: see original paper] Integrated System Overall Architecture

After product managers submit relevant product information through the Jira platform, the Jenkins system tool performs base64 encoding on each parameter to avoid parameter transmission and parsing issues. After obtaining parameter information, the Jenkins platform decodes and adapts parameters, passing them to the process orchestration module, which orchestrates the entire build and deployment process. This module also manages the transition of each decomposed stage on worker nodes and integrates final outputs. The customized notification module is responsible for timely notifications and alerts for various statuses and exceptions in the process. The role management module provides necessary security constraints for projects, granting corresponding platform usage permissions to different roles. This module also manages version repository specific paths and key configurations and is, in principle, the responsibility of quality managers.

The specific build tools used on slave nodes are as follows:

Build Tool	Version	Purpose
sshd	Follows system	Provides Jenkins runtime environment support and Android Java code execution
Build-tools	-	Official Android build assistant tools provided by Google
Openjdk	-	For Jenkins master-slave connection configuration
Gradle	-	Open-source automation build tool for parsing Android project configuration and integrating build steps

Orchestrated specific tasks are distributed to Jenkins worker nodes for concrete processing. Worker node functions mainly include: providing heterogeneous environment basic capabilities, completing client builds, third-party tool hardening, channel distribution application generation, application signing, compression packaging, and standardized production environment output.

2.3 Mobile Application Integration and Release Process Solution

According to the characteristics of China Search' s mobile client product development, automated build and release of mobile applications are divided into two scenarios: daily iterative development and product launch with multi-channel release. Two build process release solutions are designed for these application scenarios.

2.3.1 Daily Iteration Development Scenario

During daily iterative development, developers check out the latest code from the source code trunk to local. After coding and local testing, they check local copies back into the source code trunk according to configuration management strategies. The Jenkins build platform automatically receives notification of the check-in action and sequentially executes code specification checking, building, unit testing, packaging, and testing. Once an error is discovered during execution, subsequent build processes are terminated, and error information is notified to developers in real time. Relevant personnel can view the execution

process online in real time or continue other work. The system automatically sends execution results upon completion.

[Figure 4: see original paper] Automated Build Process Solution for Daily Iterative Development Scenario

2.3.2 Official Product Launch and Multi-channel Release Process

Based on code versions that have passed testing and review, product official launch and update release can be conducted according to plan or demand. Due to operational activity promotion needs, there is also a requirement for multi-channel release. Products released to the production environment for users require additional security considerations beyond daily builds to prevent mobile applications from being de-shelled, tampered with, or implanted with advertisements or other malicious operations, which would endanger user information security and affect user experience. Therefore, third-party hardening tools are introduced to perform hardening operations on installation packages. Hardened application packages have been obfuscated, encrypted, and compressed, possessing relatively high security. For numerous application markets and third-party promotion channels, channel marking is required when releasing applications. Applications installed by users in different markets and promotion channels can be measured through statistical interfaces, collected according to channel dimensions for promotion effectiveness, number of active users, user habits, etc., to facilitate further user behavior analysis and product optimization by product personnel. Application signing, as one of the Android application specifications, is mainly used for basic application information labeling, and the signature also provides a certain degree of tamper-proofing functionality. This solution utilizes key-store and signing keys to implement batch automated signing of application packages, further improving the standardized information of application client release packages. Hardened and signed application packages still need to undergo manual testing verification and approval processes before being released to the production environment for users.

[Figure 5: see original paper] Official Product Launch and Multi-channel Release Process Solution

Based on Jenkins platform' s powerful foundational capabilities and flexible extensibility, China Search has designed an automated build and deployment process for mobile application development projects, involving code building, channel distribution marking, third-party hardening, file standardization, and automated batch signing. The entire process demonstrates automation characteristics and organically integrates project-related roles and task responsibilities into various stages of the workflow, reducing communication costs among relevant personnel and enabling historical retention of operations from project building to final deployment for easy traceability. This both satisfies the needs of rapid integration for daily development projects and ensures stable and rapid release of the production environment.

(Author Affiliation: China Search Information Technology Co., Ltd.)

Note: Figure translations are in progress. See original paper for figures.