

Continuous Integration with Xcode 6

Session 415

Brent Shank

Software Engineer, Xcode

Introduction

Why continuous integration?

What is Xcode Server?

What's new in Xcode Server?

Demos

Why Continuous Integration?

Enhances collaboration

Improves software quality

Catches problems quickly and automatically

Broadens test coverage

Gathers build and test history for your project

Easily distributes builds to your team

Terminology

Terminology



Scheme

Recipe for building
your project

Terminology



Scheme

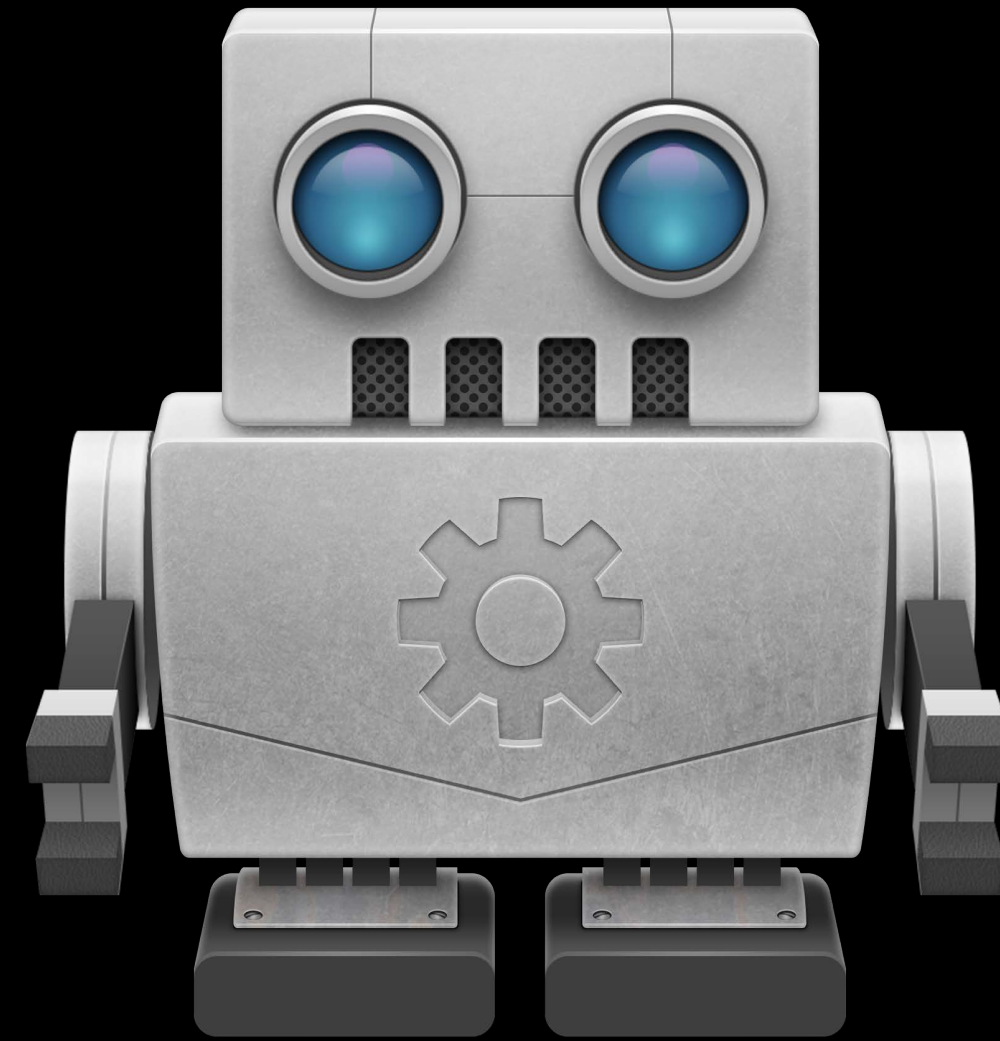
Recipe for building
your project

Terminology



Scheme

Recipe for building
your project



Bot

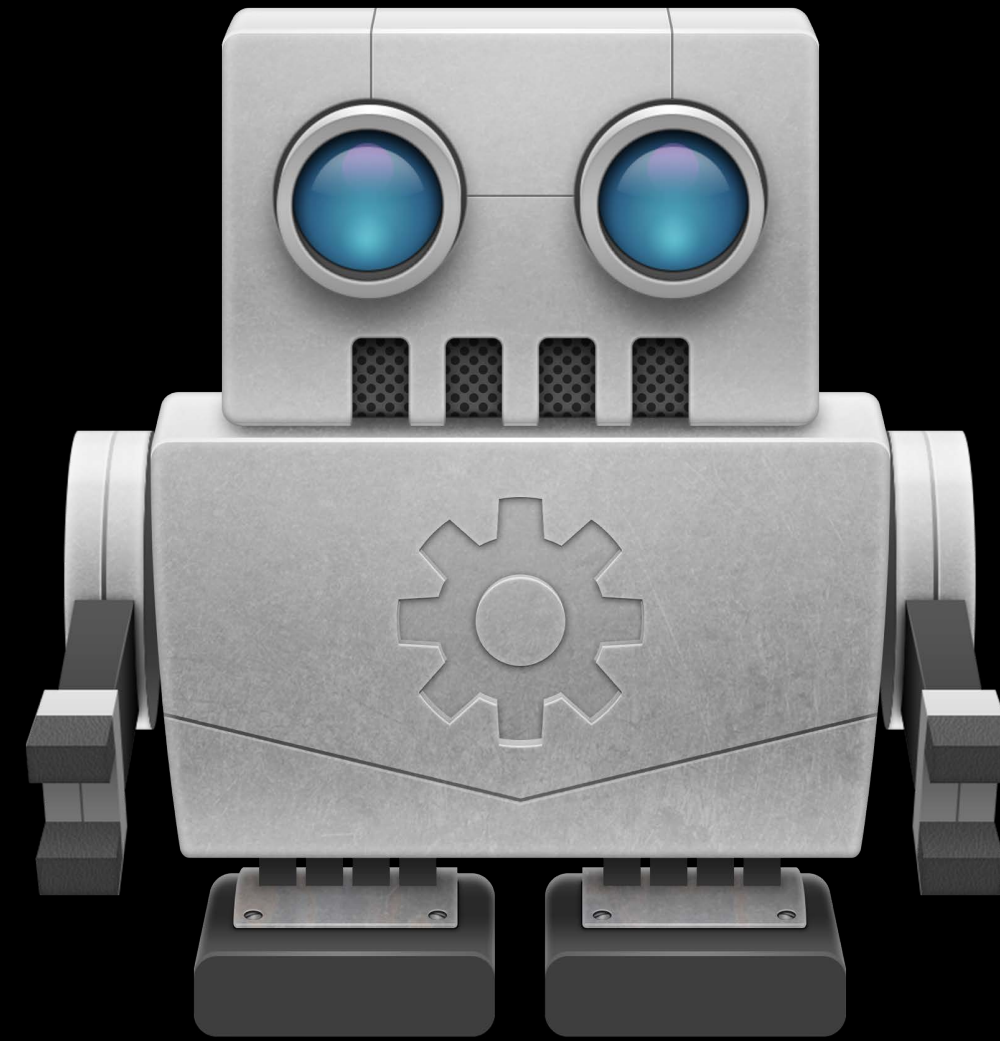
Analyze, build,
test, and archive
on a schedule

Terminology



Scheme

Recipe for building
your project



Bot

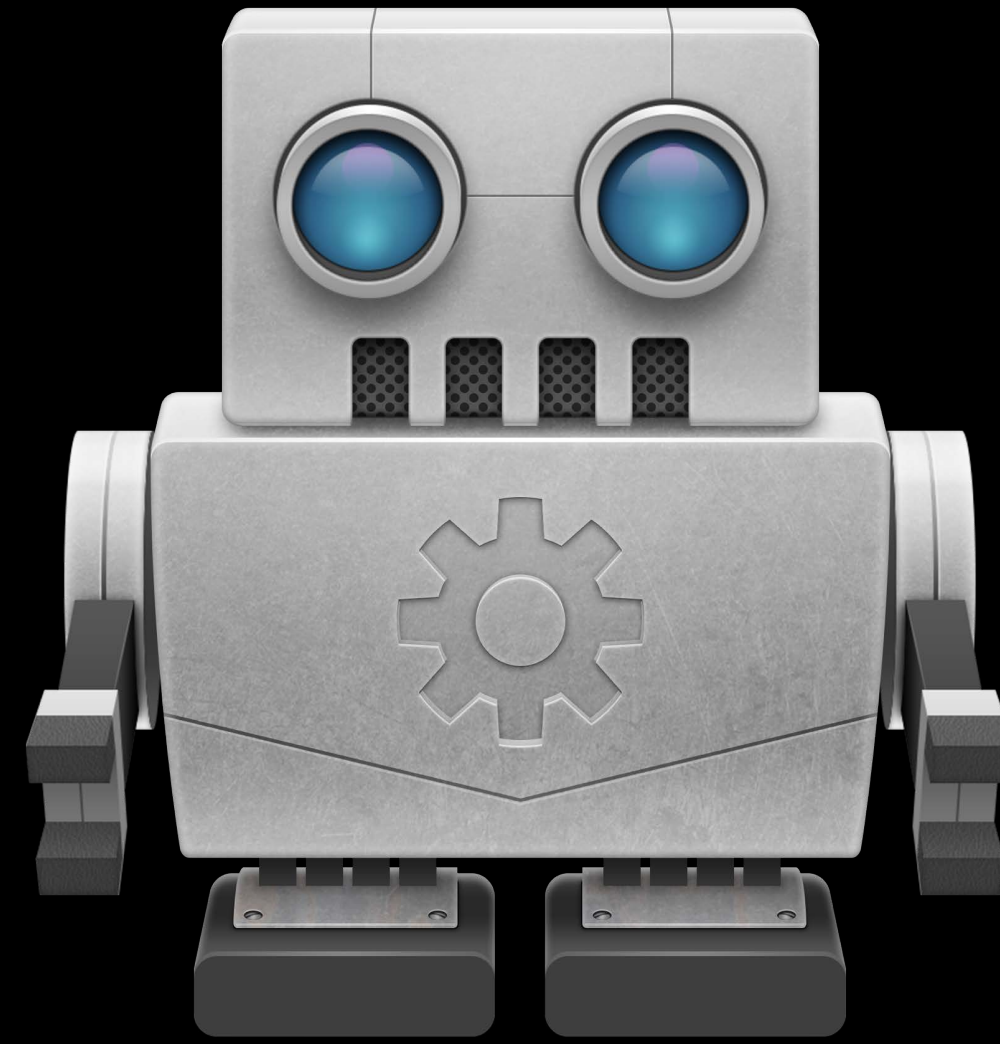
Analyze, build,
test, and archive
on a schedule



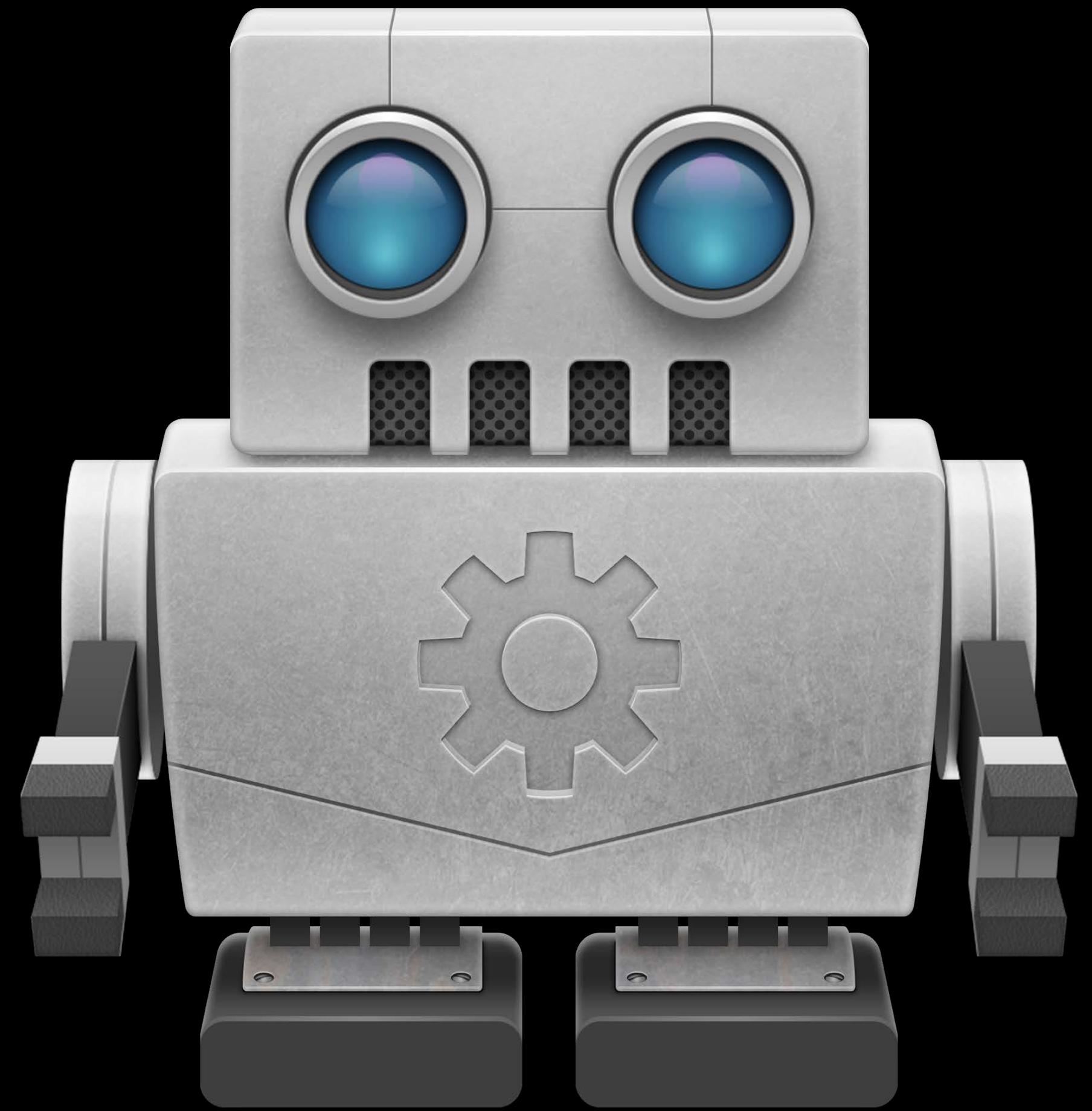
Integration

A single run
of a bot

Terminology



Terminology



What Is a Bot?

Defines what to build

- Project and SCM information

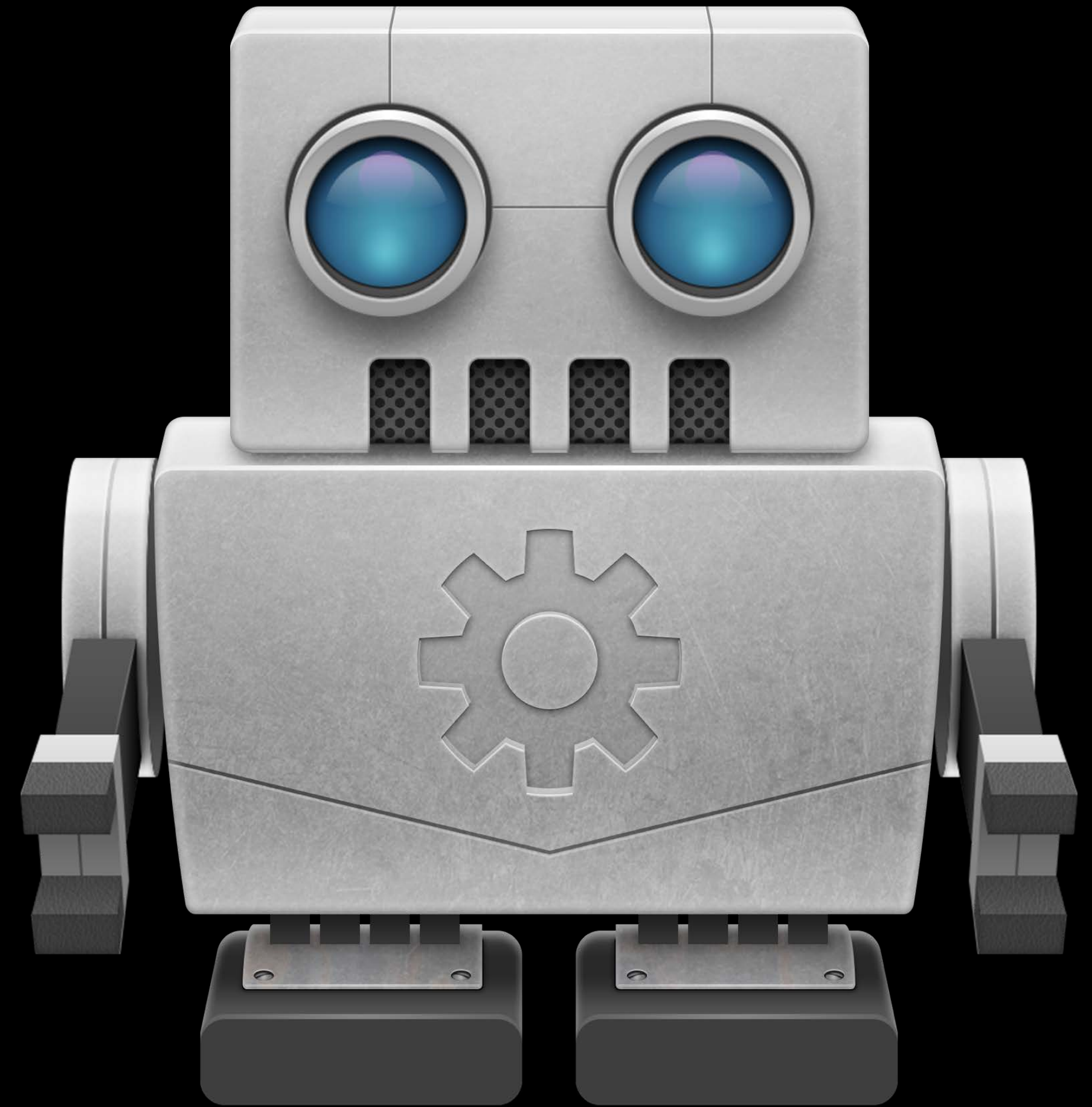
When to build it

- Periodic, on-commit, or manual

How to build it

- Shared scheme
- Static analysis
- Testing and devices
- Archives

Notifications



Xcode Server Features

Easy setup with Xcode and OS X Server

Integrates with the Apple Developer program

Builds iOS and Mac projects

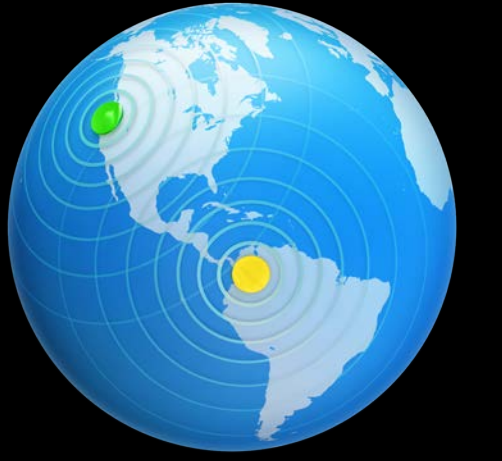
Runs tests on multiple devices and simulators

Produces IPA and PKG files

Seamless Xcode integration

Beautiful UI

OS X Server



Application that hosts Xcode Server

Simple setup

Hosted repositories

Devices

- Add and remove devices from your team
- Run tests on multiple iOS devices
- Automatic device provisioning for testing

Xcode Server for Xcode 6

What's new



A shiny new foundation

Support for Swift

Performance testing

Issue tracking

Support for pre- and post-integration triggers

Xcode Server

Improvements and performance

New foundation

Huge performance improvements

Super reliable

Great scalability



Xcode Server

Improvements and performance

New foundation

Huge performance improvements

Super reliable

Great scalability



Xcode Server

Triggers

Support for pre- and post-integration triggers

Trigger conditions

Shell script or email

Customized email reports

Environment variables

- Bot ID
- Bot Name
- Integration Number
- Integration Result
- And more!

Xcode Server

Issue tracking

Detects new issues (build errors, warnings, static analysis, test failures)

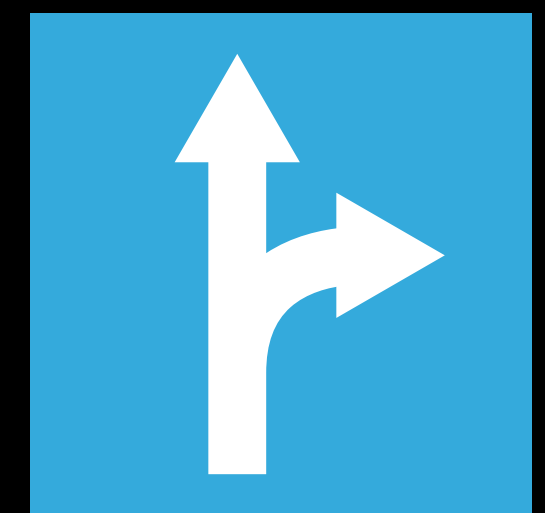
Attributes issues to committers

Tracks unresolved issues

Identifies resolved issues

Assistant editor integration

Surfaced in email notifications



Demo

Server setup, bot configuration, triggers

Matt Patenaude (a.k.a. Short Matt)

Xcode Server

Swift support



Builds Swift projects

Builds mixed Swift/Objective-C projects

Complete testing support

Performance testing

Assistant editor for issues

Complete feature parity with Objective-C projects

Xcode Server

Performance testing

Runs performance tests on multiple devices

Supports iOS devices and Mac

Leverages baselines that you configure in Xcode

Xcode Server

Performance testing—Baselines

The screenshot displays the Xcode Server interface for a project named 'iOSBotProject'. The main view shows the results for 'Integration 4', which completed 2 minutes ago. The tests are categorized as 'Passed' and 'Performance'. The tests listed are:

- testMeasureExample1: 0.11 s, Status: Passed
- testXML: 4.31 s, Status: +0.67% worse (±3%)
- testExample: 0.30 s, Status: -40.29% better
- testHoldMyHand: 0.30 s, Status: -40.29% better
- testFib: 0.00 s, Status: Passed

A performance result tooltip is shown for the 'testXML' test, displaying a bar chart and the following data:

- Performance Result
- Result: 0.672% worse (±3%)
- Average: 3.81s
- Baseline: 3.78s
- Max STDDEV: 10.00%
- Value: 3.981 (4.62%)

The bar chart shows the performance of the test across 10 runs, with the 5th run highlighted. The tooltip also includes an 'Edit' button.

Xcode Server

Performance testing—Objective-C

```
- (void)testPerformanceExample {  
    // This is an example of a performance test case.  
    [self measureBlock:^(  
        // Put the code you want to measure the time of here.  
    )];  
}
```


Xcode Server

Performance testing—Swift

```
func testPerformanceExample() {  
    // This is an example of a performance test case.  
    self.measureBlock() {  
        // Put the code you want to measure the time of here.  
    }  
}
```


Demo

Swift support, performance testing

Matt Moriarity (a.k.a. Tall Matt)

Summary

New foundation

Tons of bug fixes and under the cover improvements

Triggers

Issue tracking

Swift support

Performance testing

Summary

The screenshot displays a mobile application interface for a CI/CD pipeline. The top navigation bar shows the project name 'iOSBotProject' and the build status 'Build iOSBotProject: Succeeded' with a timestamp of '5/27/14 at 12:11 PM'. Below the navigation bar, there are tabs for 'Nightly Bot' and 'Integrate (3)'. The main content area is divided into a left sidebar and a right main panel. The sidebar, titled 'By Group', lists several components: 'iOSBotProject' (5/27/14, 12:11 PM), 'Project' (5/27/14, 12:11 PM), 'Nightly Bot' (127.0.0.1), 'Integrate (3)' (Today, 10:08 AM, Running tests), 'Integrate (2)' (Today, 10:03 AM, Finished with...), 'Integrate (1)' (Today, 9:59 AM, Failed tests), and 'On-commit Analysis Bot' (127.0.0.1). The main panel shows 'Integration 3' with the status 'Integration in progress...'. A progress bar indicates 'Integrating (step 5 of 9)...' with a blue bar showing approximately 50% completion. Below the progress bar, the text 'Running tests' and 'Testing...' is visible. The bottom of the screen shows a standard mobile OS dock with icons for settings, a clock, a document, and a home button.

Summary

The screenshot shows a CI/CD dashboard for a project named 'iOSBotProject'. The main view is the 'Summary' page for 'Integration 2', which took 2 minutes. The summary displays four key metrics: 0 Errors (No Change), 8 Warnings (No Change), 0 Issues (No Change), and 7 Tests (-1). Below the metrics, there are sections for 'Unresolved Issues' (8 warnings) and 'Resolved Issues' (1 test failure).

Integration 2
2 minutes

Summary Tests Logs

Metric	Count	Status
ERRORS	0	No Change
WARNINGS	8	No Change
ISSUES	0	No Change
TESTS	7	-1

Unresolved Issues (8 warnings)

- Warning** Unknown warning option '-Wnon-modular-include-in-framework-module'
First introduced 1 integration ago.
- Warning** Unknown warning option '-Werror=non-modular-include-in-framework-module'
First introduced 1 integration ago.
- Warning** Sending 'MainViewController *const __strong' to parameter of incompatible type 'id<NSFileManagerDelegate>'
in brentdemoapp/iOSBotProject/iOSBotProject/MainViewController.m
First introduced 1 integration ago.
- Warning** 'dismissModalViewControllerAnimated:' is deprecated: first deprecated in iOS 6.0
in brentdemoapp/iOSBotProject/iOSBotProjectTests/SongsViewController.m
First introduced 1 integration ago.
- Warning** Unused variable 'kAutoreleasePoolPurgeFrequency'
in brentdemoapp/iOSBotProject/iOSBotProjectTests/CocoaXMLParser.m
First introduced 1 integration ago.

Show more (3)

Resolved Issues

- Test Failure** failed: Time average is 20% worse (max allowed: 10%).
in <unknown>

Summary

The screenshot shows the Xcode interface for a project named 'iOSBotProject' on an 'iPod touch' device. The main window displays the test results for 'Integrate (4)', which took 2 minutes to complete. The tests are categorized as 'Passed' and 'Performance'. The test results are as follows:

Test Name	Status	Time
testMeasureExample1	Passed	0.11 s
testXML	Passed	+0.67%
testExample	Passed	
testHoldMyHand	Passed	-40.29%
testFib	Passed	0.00 s
testSort	Passed	0.33 s
testMeasureExample	Passed	0.11 s

Summary

The screenshot shows the Xcode interface for a test run. The left sidebar lists the project and build configurations. The main area displays the test results for 'Integration 4' (2 minutes). The tests are grouped by status: Passed, Failed, and All. The 'testHoldMyHand' test is highlighted with a performance regression callout.

Test Name	Device	Status	Time	Change
testMeasureExample1	iPod touch (5th generation), iOS 7.1.1	Passed	0.11 s	
testXML	iPod touch (5th generation), iOS 7.1.1	Passed	4.31 s	+0.67%
testXML	iPod touch (5th generation), iOS 8.0	Passed	4.31 s	+0.67%
testExample		Passed		
testHoldMyHand	iPod touch (5th generation), iOS 7.1.1	Passed	0.30 s	-40.29%
testHoldMyHand	iPod touch (5th generation), iOS 8.0	Passed	0.30 s	-40.29%
testFib		Passed	0.00 s	
testSort		Passed	0.33 s	
testMeasureExample		Passed	0.11 s	

Performance Result
Result: 0.672% worse (±3%)
Average: 3.81s
Baseline: 3.78s
Max STDDEV: 10.00%

Value: 3.981 (4.62%)

Summary

The screenshot displays the Xcode IDE interface for an iOS project named 'iOSBotProject'. The left sidebar shows the project's file structure, including source files like AppDelegate.h and AppDelegate.m, storyboard files, and various parser and controller classes. The main editor area is split into two panes. The top pane shows a 'Configure bot triggers:' dialog box with a 'Run Script' trigger. The script contains the following code:

```
1  #!/bin/sh
2
3  cp "$XCS_INTEGRATION_PRODUCT_PATH" /Volumes/
  WebServer/Products/
4  curl -X POST http://intranet.company.com/
  announce_build?number=
  $XCS_INTEGRATION_NUMBER&status=
  $XCS_INTEGRATION_STATUS
```

Below the script, the 'Run Script' trigger is configured with three checked options: 'On success', 'On test failures', and 'On build errors'. The dialog box has 'Cancel', 'Previous', and 'Create' buttons.

The bottom pane shows the source code for AppDelegate.m, with the following methods visible:

```
25 - (void)applicationDidEnterBackground:(UIApplication *)application
26 {
27     // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information
28     // to restore your application to its current state in case it is terminated later.
29     // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user
30     // quits.
31 }
32
33 - (void)applicationWillEnterForeground:(UIApplication *)application
34 {
35     // Called as part of the transition from the background to the inactive state; here you can undo many of the changes made on
36     // entering the background.
37 }
38
39 - (void)applicationDidBecomeActive:(UIApplication *)application
40 {
41     // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously
42     // in the background, optionally refresh the user interface.
43 }
44
45 - (void)applicationWillTerminate:(UIApplication *)application
```

More Information

Dave DeLong

Developer Tools Evangelist

delong@apple.com

Documentation

Xcode Continuous Integration Guide

<http://developer.apple.com>

Apple Developer Forums

<http://devforums.apple.com>

Related Sessions

-
- Testing in Xcode 6

Marina

Thursday 9:00AM

Labs

-
- Continuous Integration

Tools Lab C

Thursday 2:00PM

 WWDC14