Configure Bots to Perform Continuous Integrations

On This Page

Bots are processes that Xcode Server runs to perform integrations on the current version of a project in a source code repository. An integration is a single run of a bot. Integrations consist of building, analyzing, testing, and archiving the apps (or other software products) defined in your Xcode projects. With Xcode Server able to access the source code repositories of those projects, you can configure bots to perform continuous integrations on them.

On a development Mac, a *scheme* is the recipe for building your product. A scheme contains the targets, source files, and everything else that makes up your product. It also defines what operations are performed by a bot during an integration. To automate an integration, you share its scheme and create a bot to perform integrations using that scheme at the desired time. Bots can be configured to run:

- Every time a change is committed to the repository
- On a regular schedule, such as hourly, daily, or weekly
- When manually initiated

Bots also run integrations automatically whenever you install an updated version of Xcode. These integrations run immediately, prior to running any normally scheduled integrations. Compare these integrations with previous integrations to identify issues that may have been encountered as a result of the upgrade.

Share Build Schemes

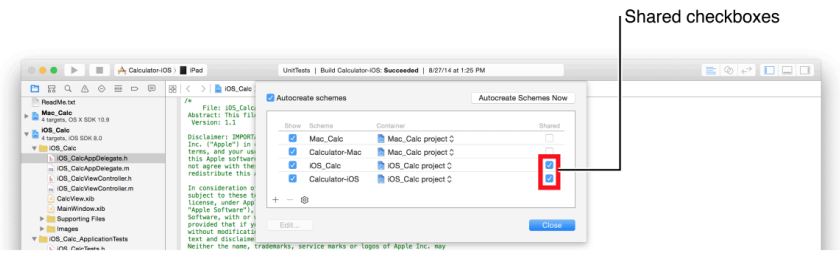
A scheme specifies which targets to build for a project, which build configuration to use, and which executable environment to use when the product is launched. When you create a new iOS or OS X project, Xcode creates a default scheme that includes settings to perform these actions:

- Analyze, which performs static code analysis
- Test, which runs any test cases that you implement
- Archive, which builds an archive of the product that the scheme built

For Xcode Server to perform these actions on a project, you must share the project's scheme. A *shared scheme* is one that you publish in a repository, along with the other shared project files.

To share an Xcode project's scheme

1. On your development Mac, check out and open the project that contains the scheme you want to share.
2. Choose Product > Scheme > Manage Schemes.
3. Select the Shared checkbox for the scheme to share, and click Close.



4. Choose Source Control > Commit.
5. Select the Shared Data folder.
6. Enter your commit message in the text field.
7. Select the "Push to remote" option (if your project is managed with Git).
8. Click the "Commit Files and Push" button.

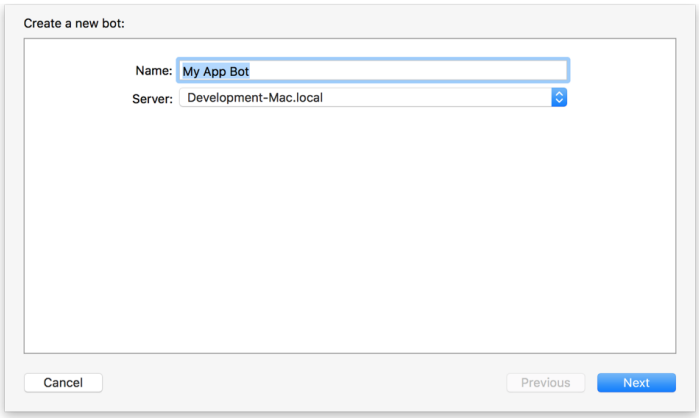
Schemes are an important part of the Xcode build system. For the purposes of using continuous integration, however, the important task is to set the chosen scheme to be shared and checked into the repository for the bot to use. For more information about managing schemes, see Xcode Help.

Create Bots to Perform Integrations

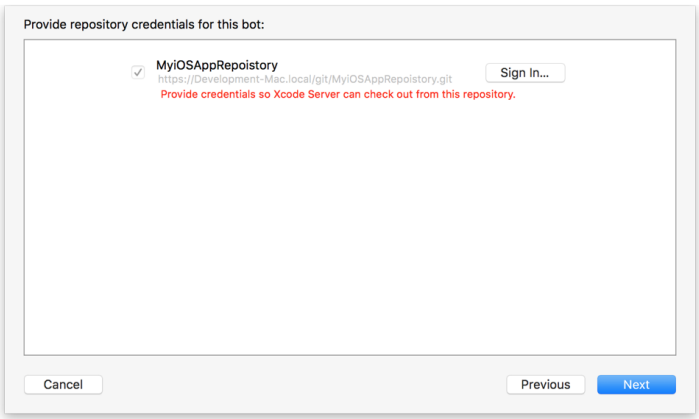
After sharing a scheme, create a bot to perform integrations using the scheme.

To create a bot

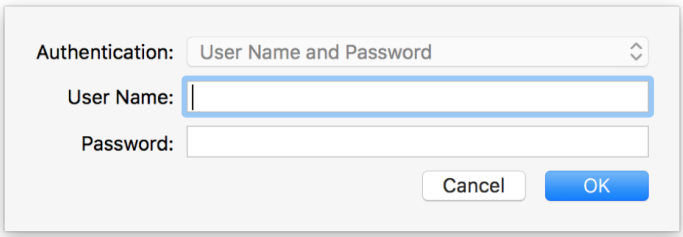
1. On your development Mac, open the Xcode project containing the scheme that defines the actions to automate
2. Choose Product > Create Bot, specify a name for the bot, choose a server, and click Next.



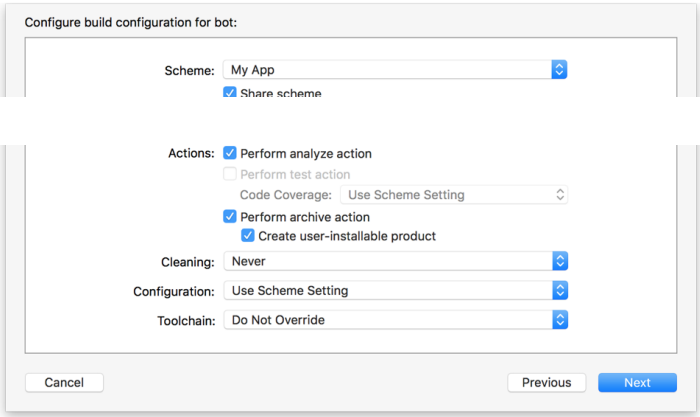
3. Select a repository and click Sign In to provide credentials for your repository. You need to do this even if you've added repository credentials in your Xcode preferences because each bot stores its own set of credentials in a secure keychain on the server.



Enter your authentication credentials when prompted, then click OK.



4. Configure the desired attributes for the bot, and click Next.



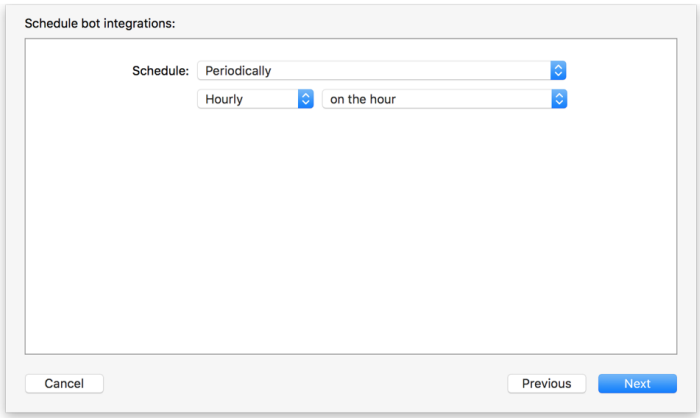
On This Page

- Choose a scheme and indicate whether to share the scheme through the project's source code repository.
- Specify bot actions by selecting the appropriate checkboxes. You can enable static analysis, testing, and product archiving.
- Choose whether to clean products before building. When performing a clean integration, the bot won't reuse the previous build. Use the Cleaning pop-up menu to specify the frequency with which to clean: before each integration, once a day, once a week, or never.
- If you have alternative toolchains installed in `/Library/Developer/Toolchains/`, such as toolchains downloaded from Swift.org, select a toolchain to use when running integrations. To use the default tools, set the Toolchain pop-up menu to Do Not Override.

> IMPORTANT
>
> If you use an alternative toolchain, you might encounter regressions because the alternative tools won't have been fully qualified by Apple for official release.

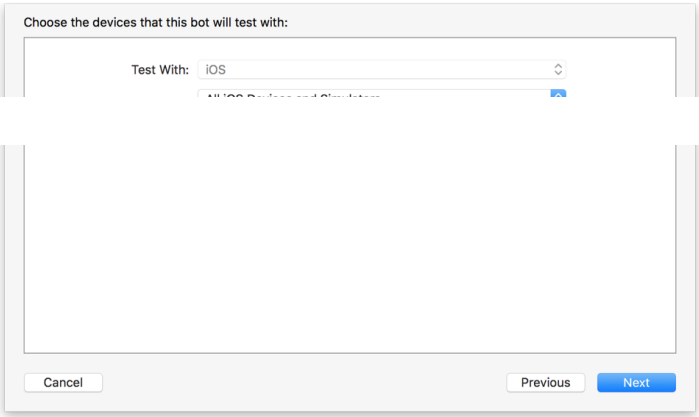
5. Specify an integration schedule, and click Next.

   You can schedule the bot to perform integrations periodically (hourly, daily, or weekly), on every commit, or manually.



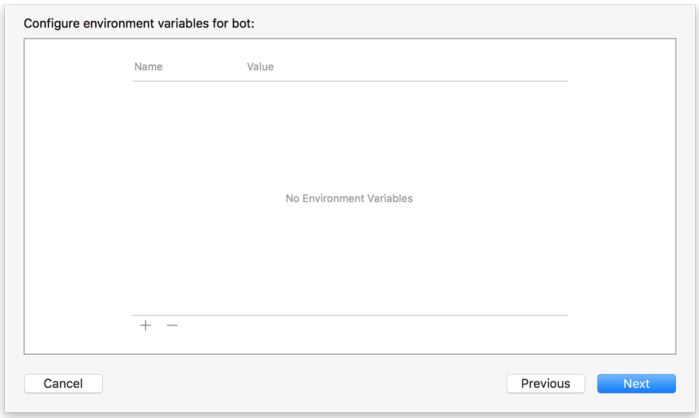
6. For an iOS app, choose what kinds of devices or simulators the bot will be tested on and click Next.

   Any devices you specify must be connected to the server for the test action to complete.



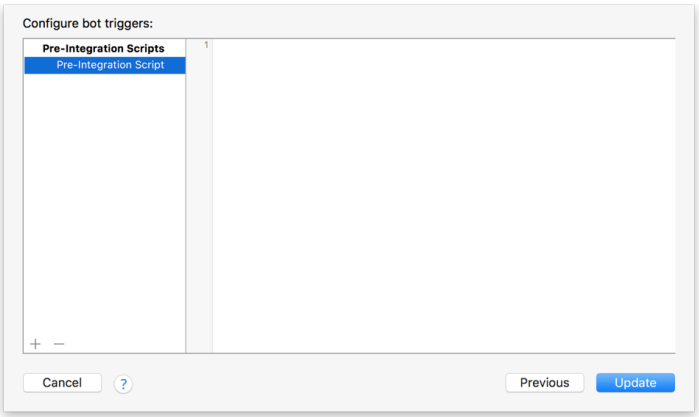
On This Page

7. Define any environment variables needed by Run Script build phases that execute as part of your integration, or for your pre-integration and post-integration triggers, then click Next.



> NOTE
>
> In addition to the custom environment variables you define here, you can use Xcode Server's built-in environment variables in Run Script build phases that execute as part of your integration, pre-integration triggers, and post-integration triggers. For a list of these variables, see Xcode Server Environment Variable Reference.

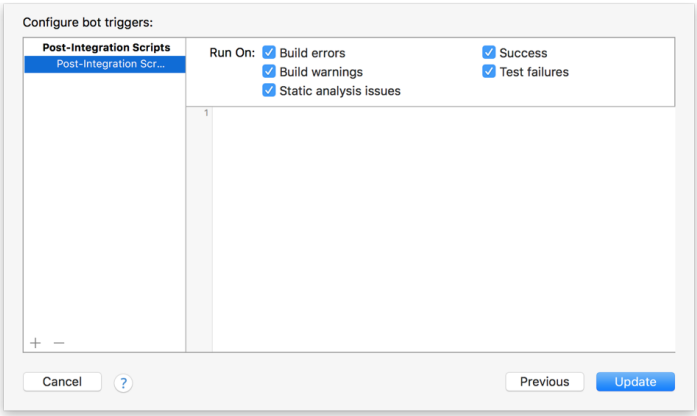
8. Bots can be configured to perform actions—known as triggers—before and after integration. A trigger can run custom shell scripts and send email reports.
    - Pre-integration triggers run before an integration. To create a pre-integration script, click the Add button (+) at the bottom of the script pane and choose Pre-Integration Script from the pop-up menu. Enter the desired script into the script pane. This script can reference any environment variables you defined in the previous configuration step. It can also reference Xcode's built-in environment variables.



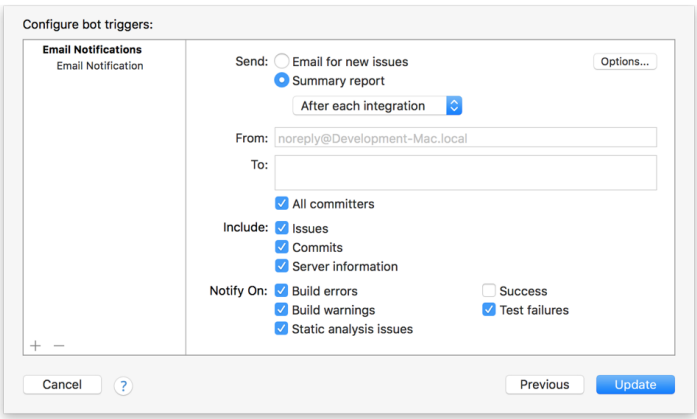
- Post-integration triggers run after an integration. To create a post-integration script, click the Add button (+) at the bottom of the script pane and choose Post-Integration Script from the pop-up menu. Enter the desired script into the script pane. This script can reference any environment variables you defined in the previous configuration step. It can also reference Xcode's built-in en

  on success, on test failures, on build errors, on build warnings, or on static analysis warnings. Select the appropriate condition checkboxes for your integration.

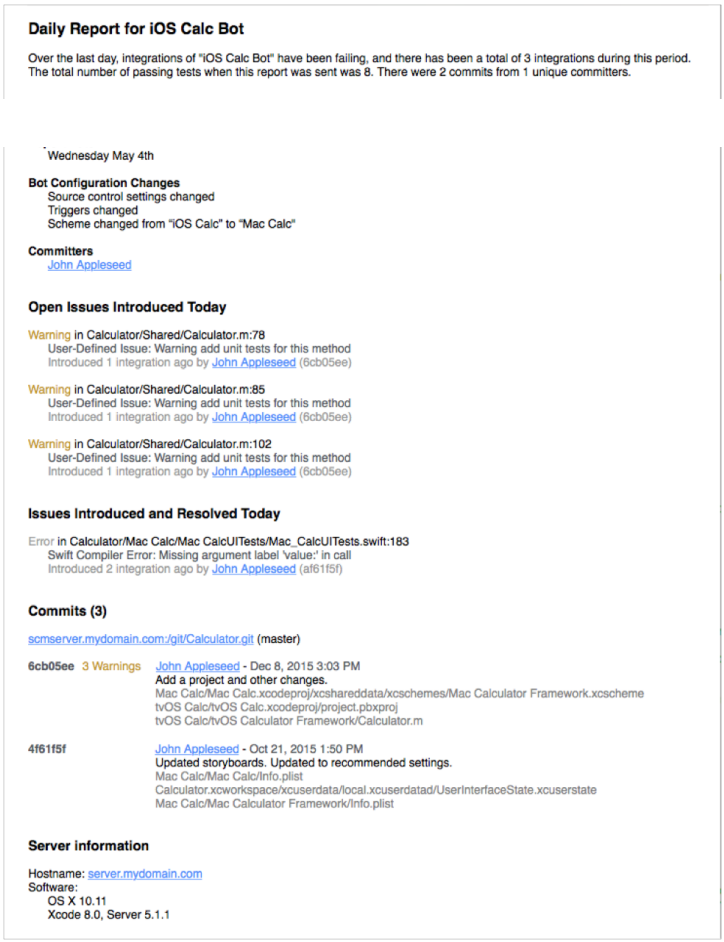
On This Page



- Email notification triggers run when new issues occur, after performing an integration, daily, or weekly. To create a post-integration script, click the Add button (+) at the bottom of the script pane and choose Email Notification from the pop-up menu. Specify whether you want notifications for new issues only, or regular summary reports. New issue reports go to the committers. For summary reports, you must specify the recipients. Select the information to include and the conditions under which reports are sent, such as on success, on test failures, on build errors, on build warnings, or on static analysis warnings.



The following screenshot shows an example of a daily summary report email notification.



Daily Report for iOS Calc Bot

Over the last day, integrations of "iOS Calc Bot" have been failing, and there has been a total of 3 integrations during this period. The total number of passing tests when this report was sent was 8. There were 2 commits from 1 unique committers.

Wednesday May 4th

Bot Configuration Changes
Source control settings changed
Triggers changed
Scheme changed from "iOS Calc" to "Mac Calc"

Committers
John Appleseed

Open Issues Introduced Today

Warning in Calculator/Shared/Calculator.m:78
User-Defined Issue: Warning add unit tests for this method
Introduced 1 integration ago by John Appleseed (6cb05ee)

Warning in Calculator/Shared/Calculator.m:85
User-Defined Issue: Warning add unit tests for this method
Introduced 1 integration ago by John Appleseed (6cb05ee)

Warning in Calculator/Shared/Calculator.m:102
User-Defined Issue: Warning add unit tests for this method
Introduced 1 integration ago by John Appleseed (6cb05ee)

Issues Introduced and Resolved Today

Error in Calculator/Mac Calc/Mac CalcUITests/Mac_CalcUITests.swift:183
Swift Compiler Error: Missing argument label 'value:' in call
Introduced 2 integration ago by John Appleseed (af61f5f)

Commits (3)

scmserver.mydomain.com:/git/Calculator.git (master)

6cb05ee 3 Warnings John Appleseed - Dec 8, 2015 3:03 PM
Add a project and other changes.
Mac Calc/Mac Calc.xcodeproj/xcshareddata/xcschemes/Mac Calculator Framework.xcscheme
tvOS Calc/tvOS Calc.xcodeproj/project.pbxproj
tvOS Calc/tvOS Calculator Framework/Calculator.m

4f61f5f John Appleseed - Oct 21, 2015 1:50 PM
Updated storyboards. Updated to recommended settings.
Mac Calc/Mac Calc/Info.plist
Calculator.xcworkspace/xcuserdata/local.xcuserdatad/UserInterfaceState.xcuserstate
Mac Calc/Mac Calculator Framework/Info.plist

Server information

Hostname: server.mydomain.com
Software:
OS X 10.11
Xcode 8.0, Server 5.1.1

9. Click Create to build the bot.

As explained in Manage and Monitor Bots from the Report Navigator, use the report navigator to manually start the bot, edit the bot, and delete it. Use a web browser to monitor the status of your bots, download integration assets, and install iOS products, as discussed in Monitor Bots from a Web Browser.

Follow Best Practices

To take advantage of continuous integration in your product development workflow, follow these guidelines:

- **Develop test suites and test cases.** After developing tests, include them in schemes for your bots to run. To help ensure that the changes you make aren't broken by you or others later, complement those changes with tests that determine whether a method or a set of methods used in a sequence functions as intended.
- **Perform static analysis.** Include static analysis in your integrations. Static analysis is a deep examination of your code, following code paths that your app may not follow during normal development. This process uncovers hard-to-find coding errors and also identifies areas in your code that don't follow recommended API usage, such as Foundation and AppKit idioms.
- **Conduct performance testing.** Configure Xcode to run performance tests for your apps on multiple devices. This will allow you to track your performance and identify potential problems up front, before distributing your product to users.
- **Ensure that your product builds and is packaged correctly.** Archive your product after making major changes, especially structural changes, such as adding or removing files. Let your bots archive for you automatically. The ability to build and archive your product is a main indicator of the correctness of your code changes.

For detailed information on the extensive testing capabilities in Xcode, refer to *Testing with Xcode*.