Xcode Cloud

Continuous integration & delivery

Xcode Cloud

This is a CI/CD built into Xcode and designed specifically for Apple developers. It combines Xcode, TestFlight and App Store Connect.

- Automatically test apps on Apple devices in Simulator.
- Automatically submitting an app to TestFlight.
- Automatically send the app for review before publishing to the App Store.
- Access to Apple's cloud infrastructure.
- Potential bug notifications.

This feature is available starting with Xcode 13, and is currently in beta testing.

Requirements

To work with Xcode Cloud you need to meet some requirements.

Developer account requirements:
• You must be registered with the Apple Developer Program.
• An Apple ID must be added to Xcode.
• The app must have been created in App Store Connect or you must have permission to create it.

Project and workspace requirements:
• The project must contain an Xcode project or workspace file.
• A shared schema must be used.
• Archiving action for the schema must be enabled.
• A new build system must be used.
• Dependencies and libraries must be available for Xcode cloud.
• Automatic code signing must be used.

Version control system requirements:
Xcode Cloud requires the code to be in a Git repository. In addition, you will need a specific permission or role to connect Xcode Cloud to your repository. It supports the following source control providers:
• Bitbucket Cloud and Bitbucket Server – requires the administrator role to connect.
• GitHub and GitHub Enterprise – requires the organization owner or administrator role (if the organization is not used).
• GitLab and self-managed GitLab – maintainer role required.

Connection

Xcode ● ● Report navigation ● ● Cloud ● ● Select Product

↓

First Workflow

First workflow

When setting up Xcode Cloud, the first workflow includes:

Build for each change or pull request associated with the default branch.

Using the latest version of macOS and Xcode for a temporary environment.

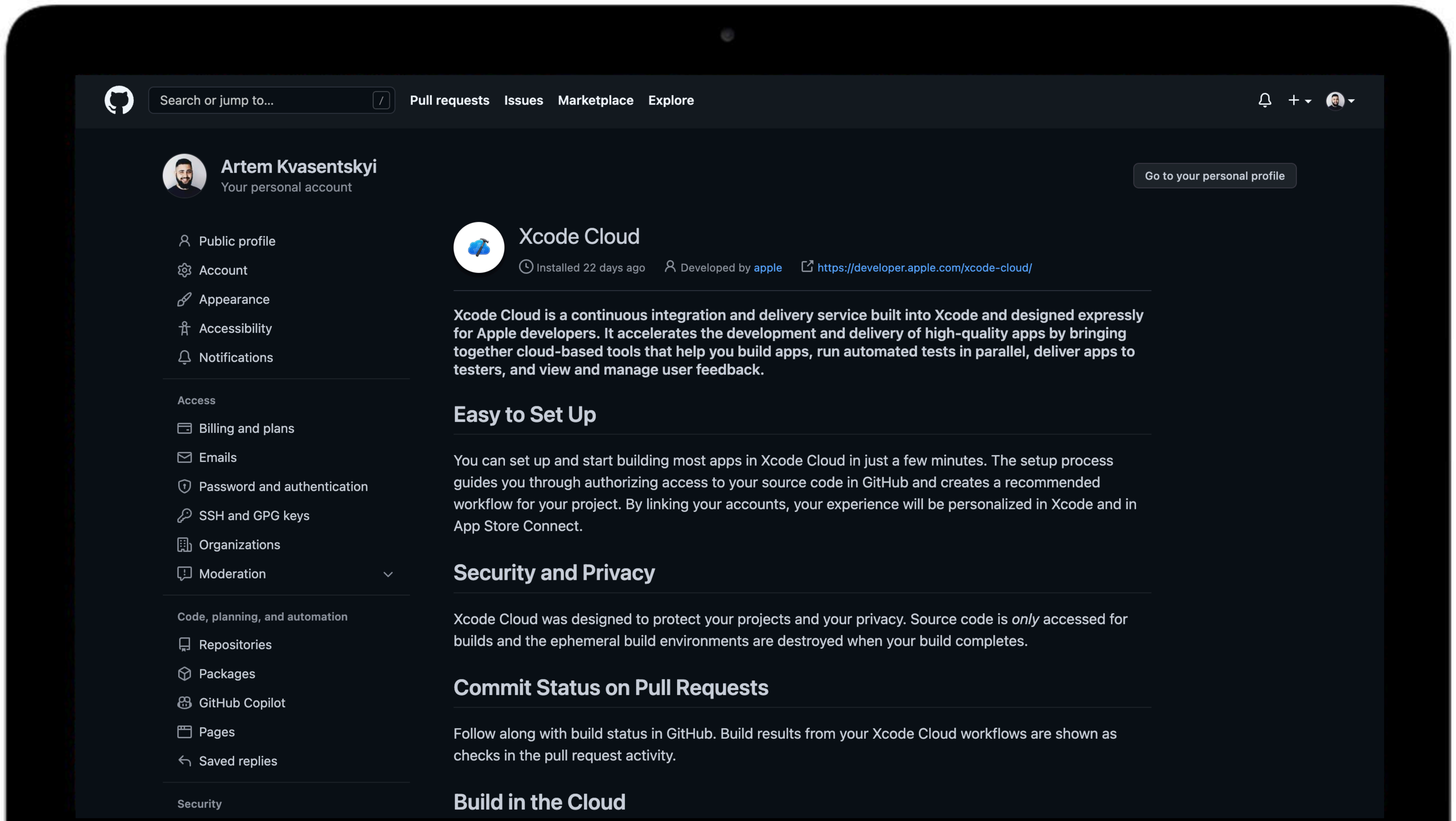
Using the archiving action.

Sending an email with information about the build upon completion.

You can edit this workflow if you need.

Setting up a repository

Xcode Cloud requires access to a Git repository with the project. It uses this access to automatically create and test code when changes are made. You will need to go through the authorisation process on your SCM provider's website.



Xcode Cloud

Installed 22 days ago · Developed by apple · https://developer.apple.com/xcode-cloud/

Xcode Cloud is a continuous integration and delivery service built into Xcode and designed expressly for Apple developers. It accelerates the development and delivery of high-quality apps by bringing together cloud-based tools that help you build apps, run automated tests in parallel, deliver apps to testers, and view and manage user feedback.

Easy to Set Up

You can set up and start building most apps in Xcode Cloud in just a few minutes. The setup process guides you through authorizing access to your source code in GitHub and creates a recommended workflow for your project. By linking your accounts, your experience will be personalized in Xcode and in App Store Connect.

Security and Privacy

Xcode Cloud was designed to protect your projects and your privacy. Source code is *only* accessed for builds and the ephemeral build environments are destroyed when your build completes.

Commit Status on Pull Requests

Follow along with build status in GitHub. Build results from your Xcode Cloud workflows are shown as checks in the pull request activity.

Build in the Cloud

Workflow

**Workflow is the configuration of the steps you want to perform in Xcode Cloud.
Workflow includes the following settings:**

General

Environment

Start Condition

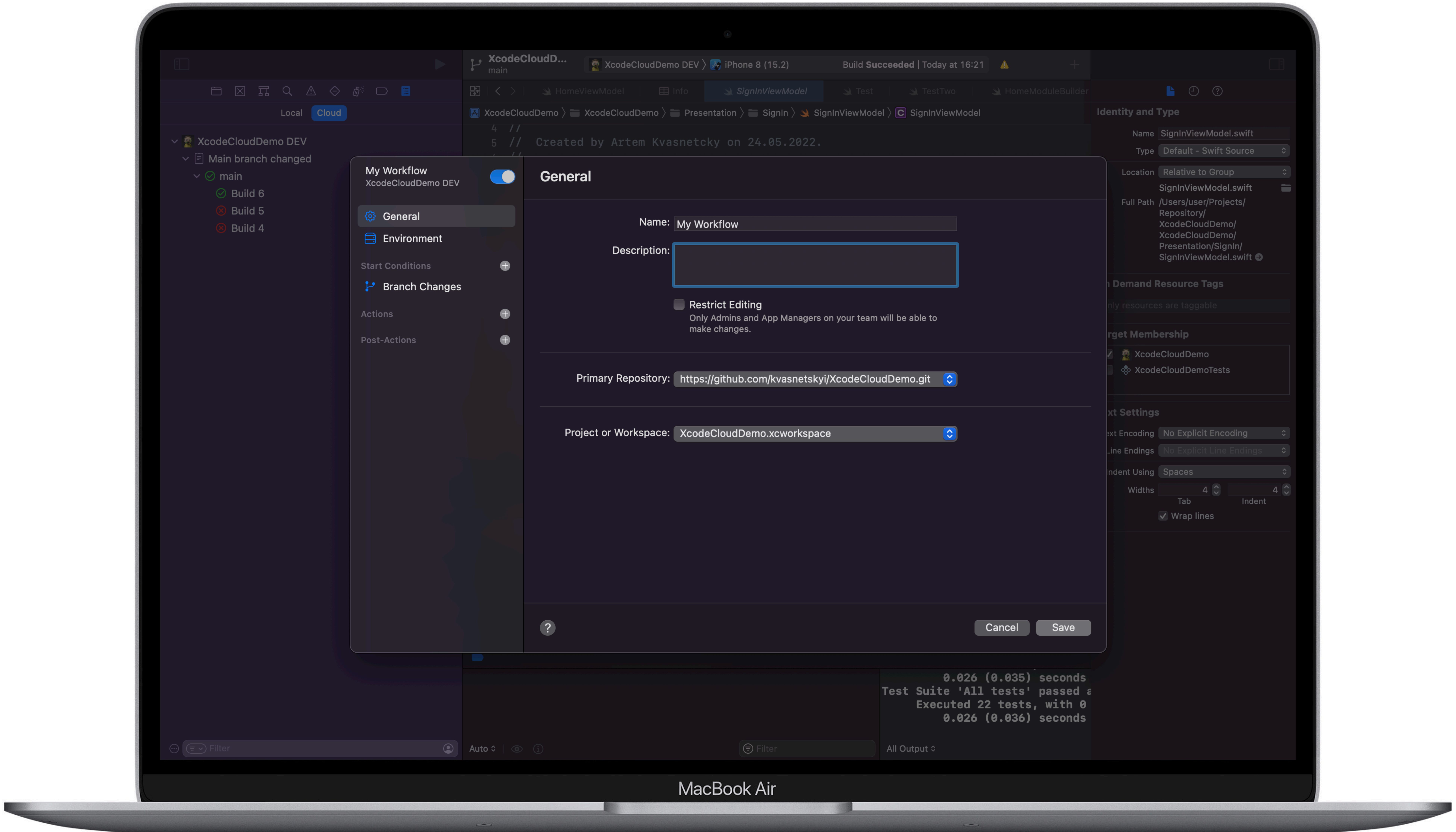
Actions

Post Actions

Custom build scripts

You can add new workflows or edit, duplicate, delete and suspend existing ones from Xcode or App Store Connect.

Workflow/General



General

Name: My Workflow

Description:

Restrict Editing
Only Admins and App Managers on your team will be able to make changes.

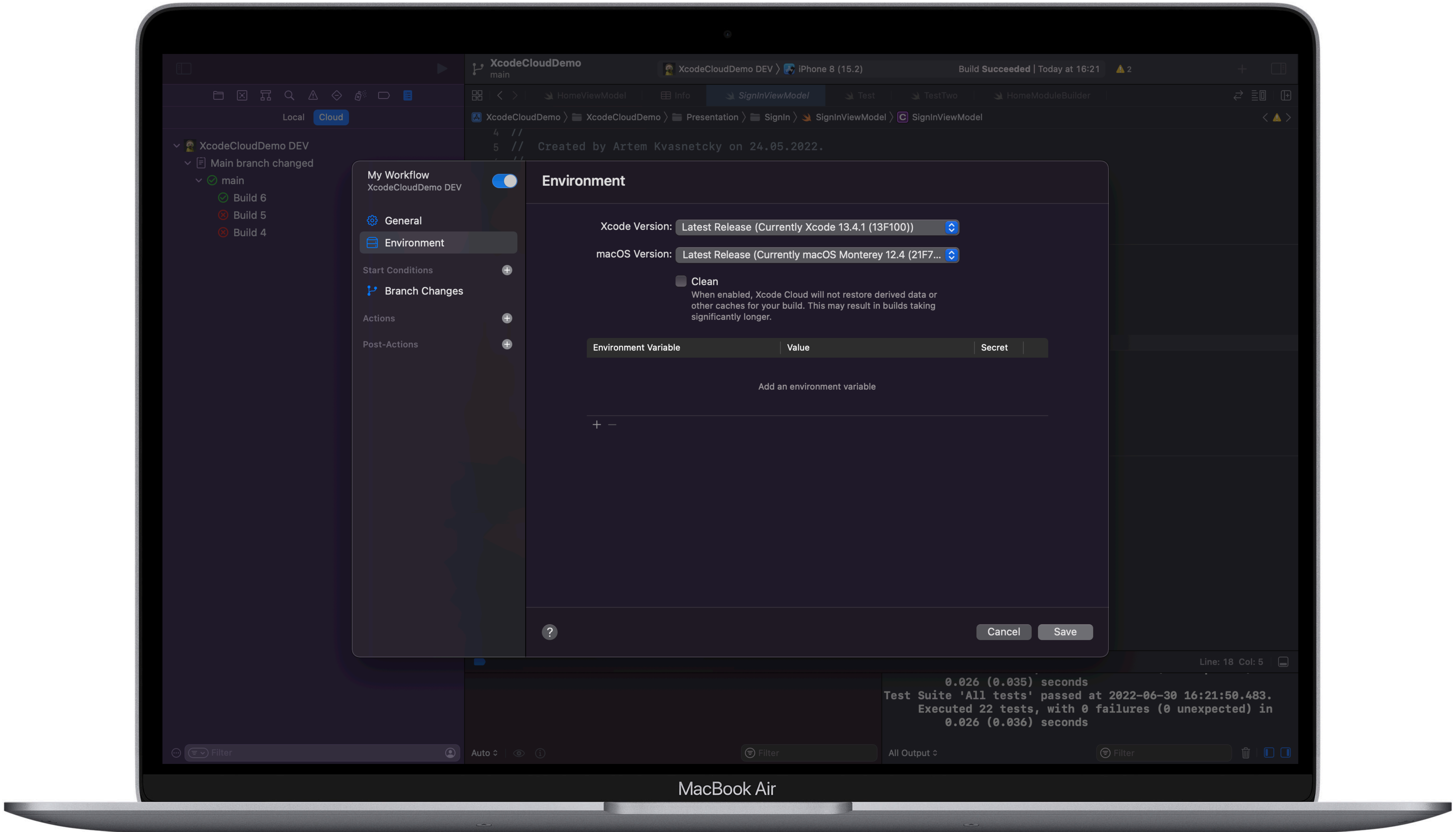
Primary Repository: https://github.com/kvasnetskyi/XcodeCloudDemo.git

Project or Workspace: XcodeCloudDemo.xcworkspace

```
// Created by Artem Kvasnetcky on 24.05.2022.
```

```
0.026 (0.035) seconds
Test Suite 'All tests' passed a
Executed 22 tests, with 0
0.026 (0.036) seconds
```

Workflow/Environment



```
//  Created by Artem Kvasnetcky on 24.05.2022.
```

Environment

Xcode Version: Latest Release (Currently Xcode 13.4.1 (13F100))

macOS Version: Latest Release (Currently macOS Monterey 12.4 (21F7...

☐ Clean
When enabled, Xcode Cloud will not restore derived data or other caches for your build. This may result in builds taking significantly longer.

Environment Variable	Value	Secret

Add an environment variable

```
    0.026 (0.035) seconds
Test Suite 'All tests' passed at 2022-06-30 16:21:50.483.
     Executed 22 tests, with 0 failures (0 unexpected) in
     0.026 (0.036) seconds
```

Workflow/Start Condition

Determine when Xcode Cloud starts a workflow.

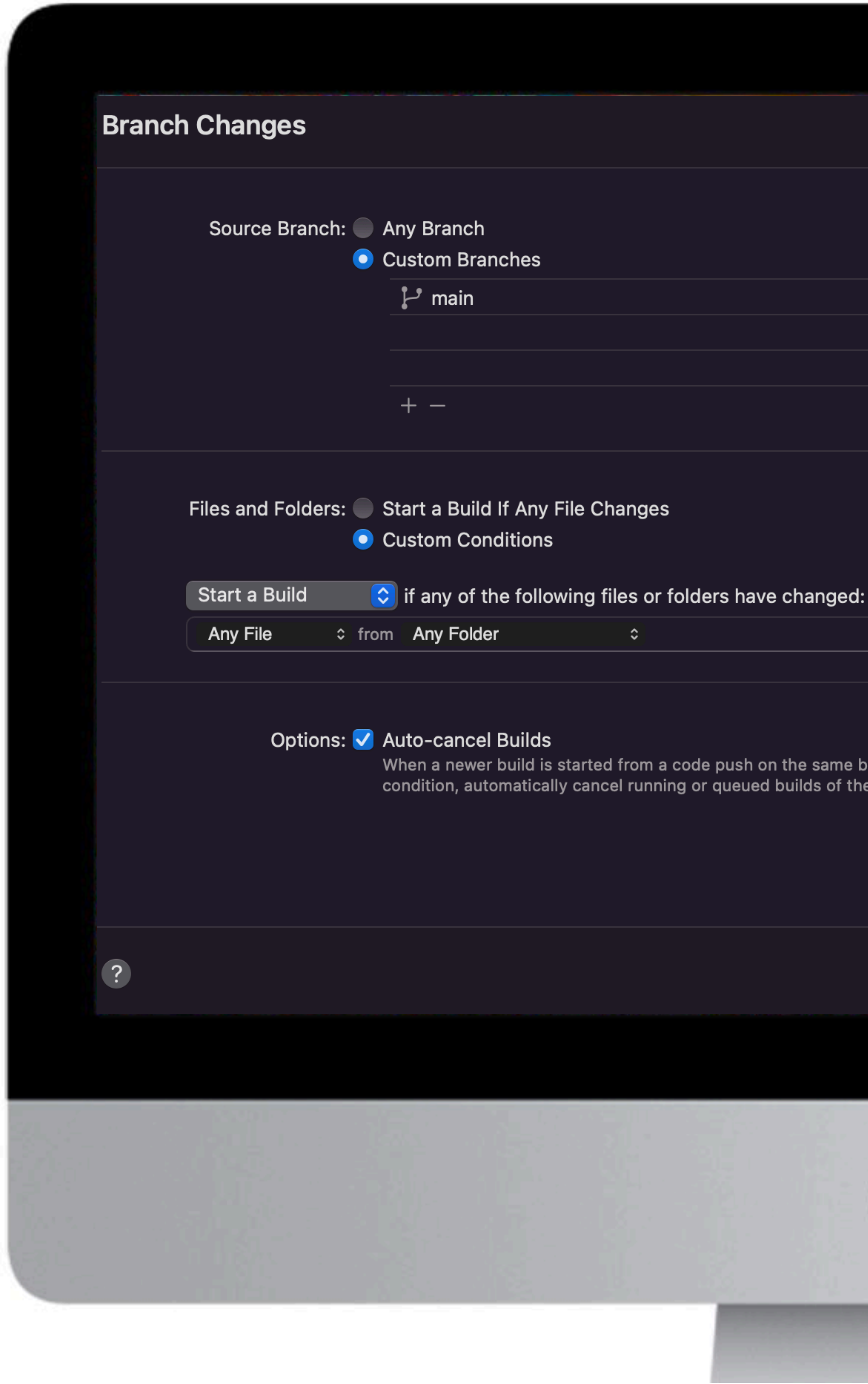
Branch Changes
Any, a specific, or several specific branches have been changed.

Tag Changes
A Git tag was created or changed.

Pull Request Changes
A PR has been created or changed.

On a Schedule for a Branch
A pre-set time has elapsed.

For all conditions except "On a Schedule" you can select **"Monitor or Ignore Specific Files and Folders"**, which can help you ignore, or alternatively, pay attention to changes if they affect:
• Any file in a specific folder.
• A specific file in any or a specific folder.
• A file with a specified extension in any folder or specific folder.



Branch Changes

Source Branch: Any Branch [✓ Custom Branches]
main

Files and Folders: Start a Build If Any File Changes [✓ Custom Conditions]

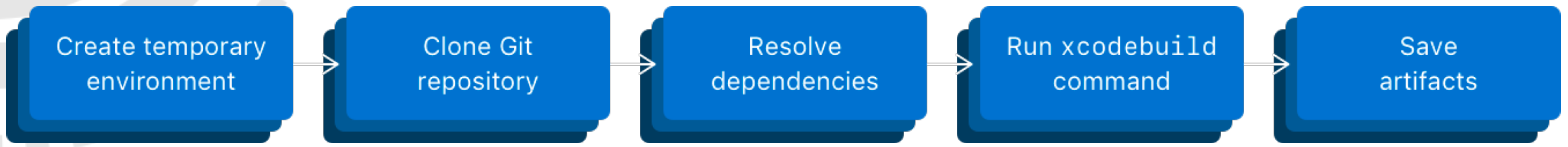
Start a Build if any of the following files or folders have changed:
Any File from Any Folder

Options: [✓ Auto-cancel Builds]
When a newer build is started from a code push on the same b
condition, automatically cancel running or queued builds of the

Workflow/Actions

These are the actions that will be performed when conditions are called from Start Condition.

When Xcode Cloud runs an actions it:



You can choose from the following available actions:

Build

Analyse

Test

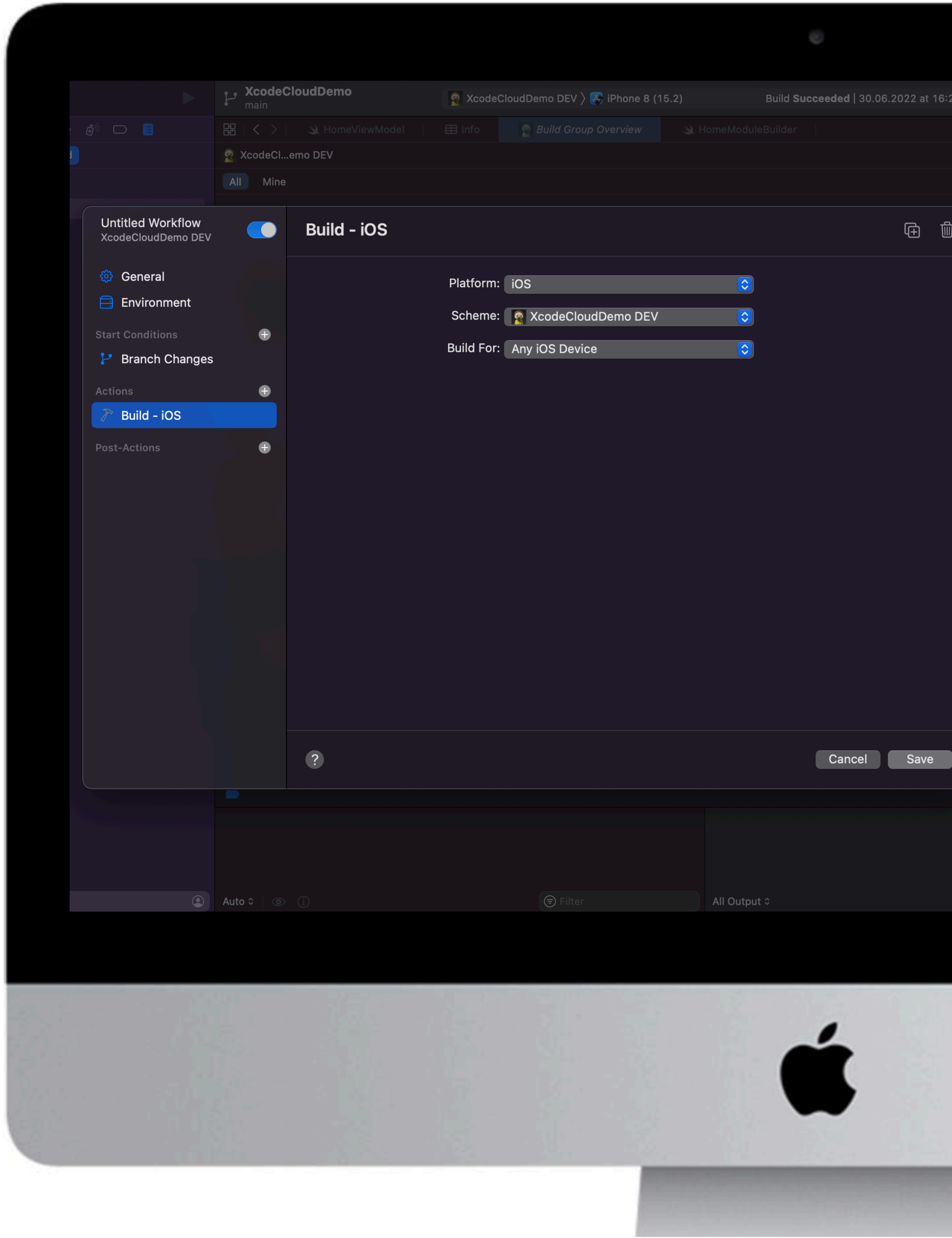
Archive

Workflow/Actions/Build

When Xcode Cloud performs the build action, it accesses the source code and runs the **xcodebuild build** command to create the build product.

Once complete, Xcode Cloud makes the following artefacts available:

- build product,
- build logs,
- result bundle.



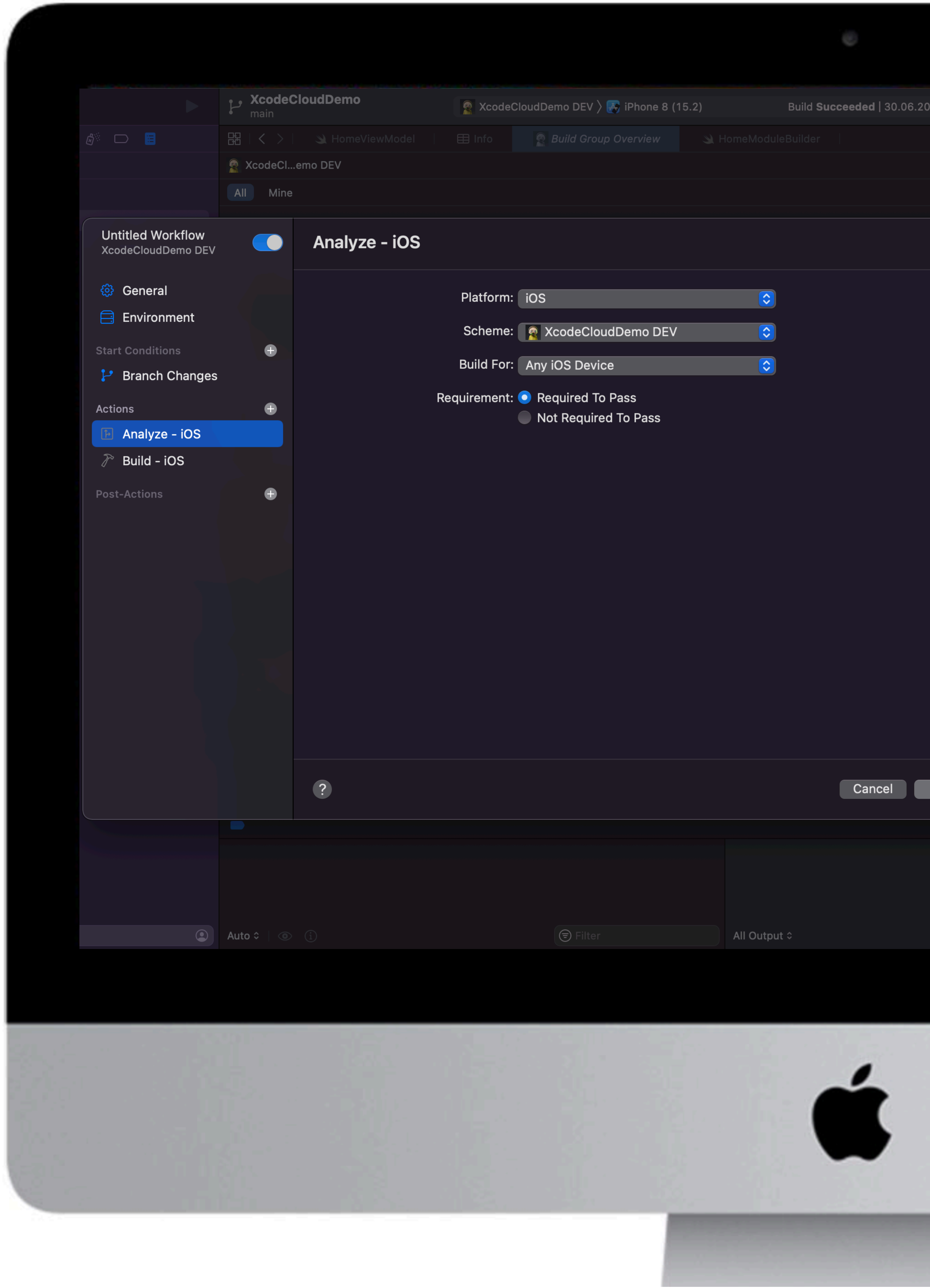
Build - iOS

Platform: iOS
Scheme: XcodeCloudDemo DEV
Build For: Any iOS Device

Workflow/Actions/Analyse

Analysis can help look for memory leaks or other problems. This step is quite time-consuming, so it is not recommended to run it regularly.

When Xcode Cloud performs the analysis action, it accesses your source code and runs the **xcodebuild analyze** command.



Untitled Workflow
XcodeCloudDemo DEV

General
Environment
Start Conditions
Branch Changes
Actions
Analyze - iOS
Build - iOS
Post-Actions

Analyze - iOS

Platform: iOS
Scheme: XcodeCloudDemo DEV
Build For: Any iOS Device
Requirement: [✓ Required To Pass] Not Required To Pass

Workflow/Actions/Test

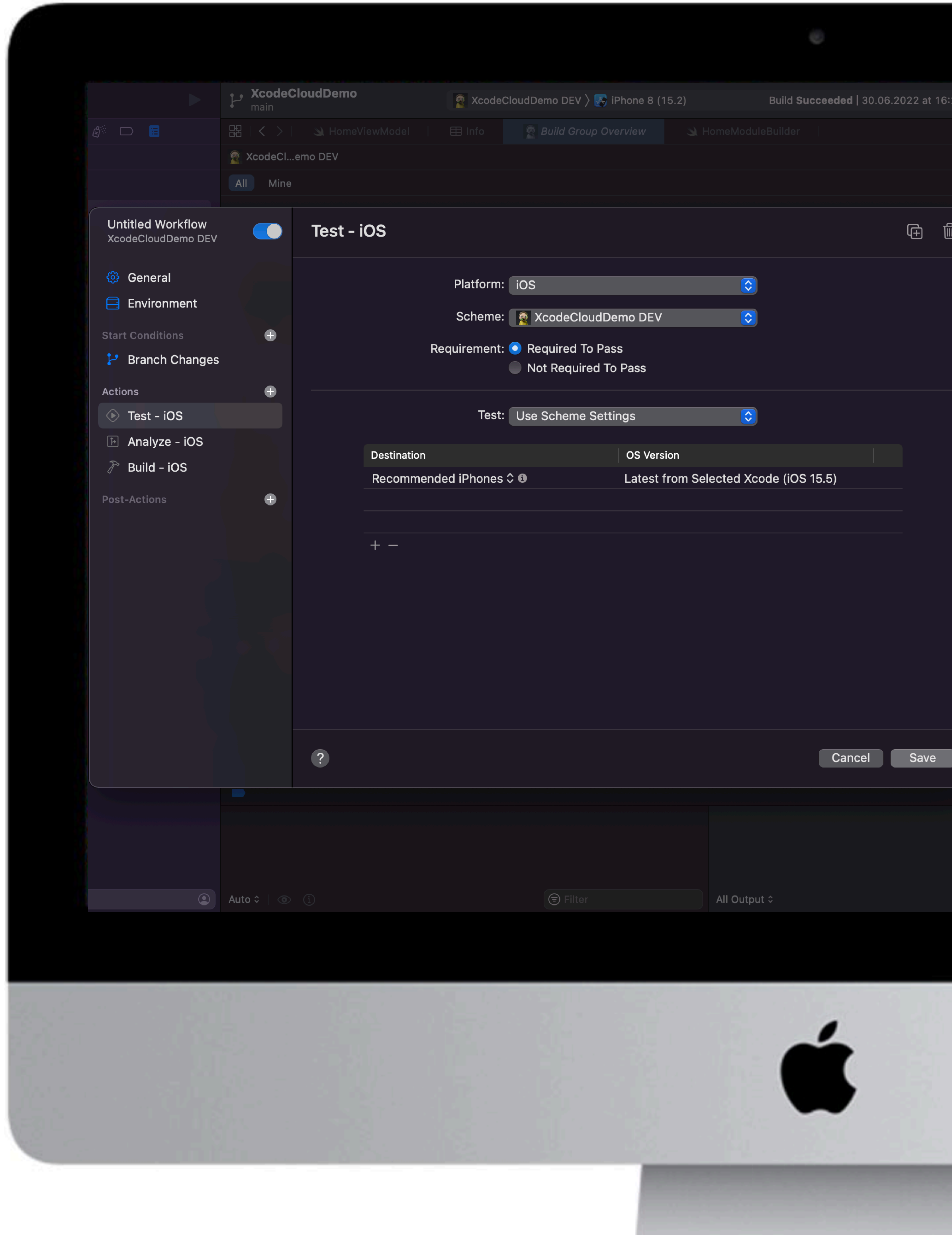
The test action is performed in two separate steps:

1. **xcodebuild build-for-testing command**

In the first step, Xcode Cloud accesses the source code and runs the xcodebuild build-for-testing command.

2. **xcodebuild test-without-building**

In the second step, Xcode Cloud uses the build created in the first step to run your tests with the xcodebuild test-without-building command.



Test - iOS

Platform: iOS

Scheme: XcodeCloudDemo DEV

Requirement: [✓ Required To Pass] Not Required To Pass

Test: Use Scheme Settings

Destination	OS Version
Recommended iPhones	Latest from Selected Xcode (iOS 15.5)

Workflow/Actions/Archive

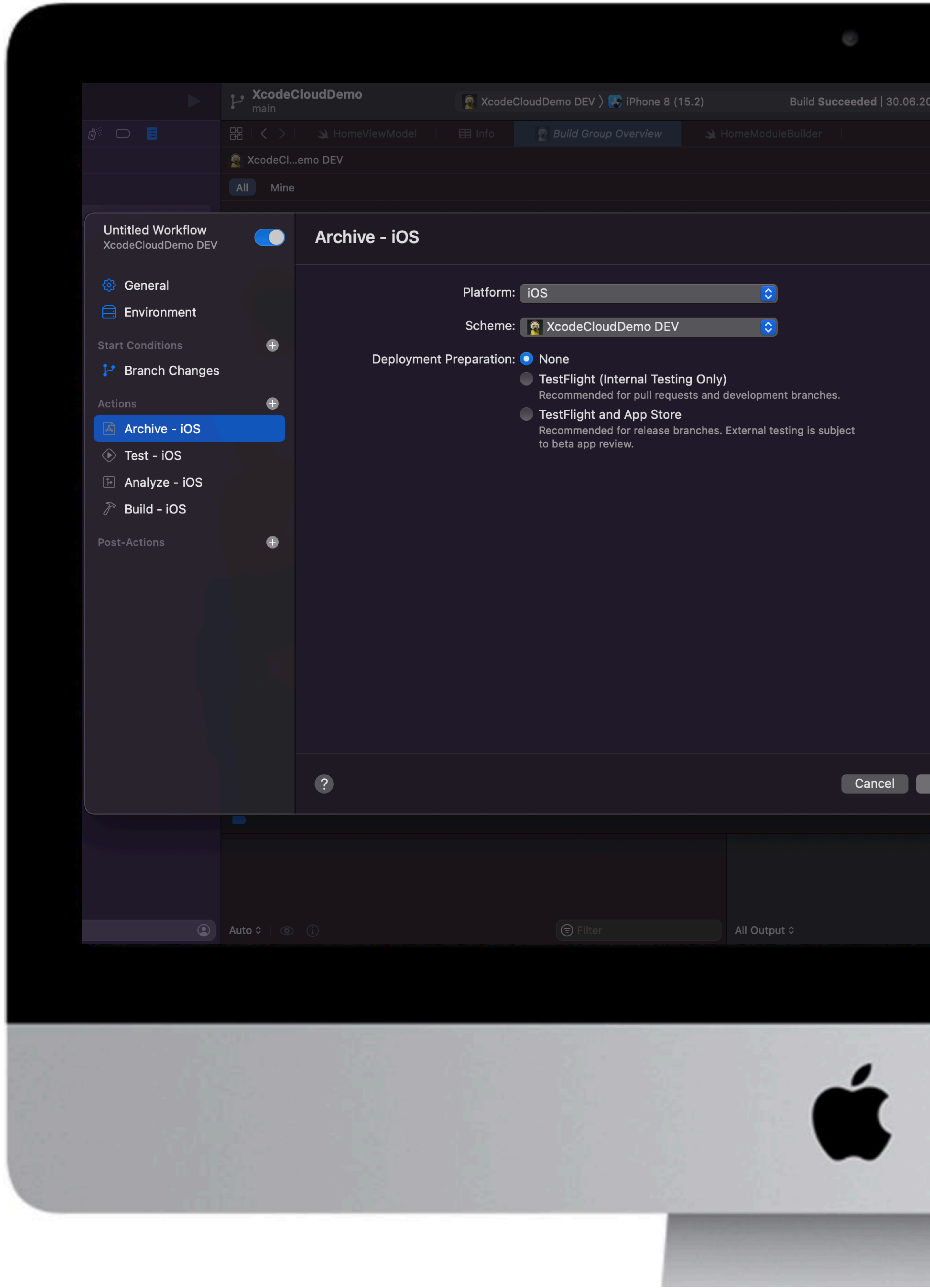
When Xcode Cloud performs the archive action, it accesses your code and runs the **xcodebuild archive** command.

When archiving, you will need to select the destination of your archive. Possible options:

- **None**
  Use this option if you are not setting up a workflow to distribute the application.

- **TestFlight (Internal Testing Only)**
  The exported application archive is suitable for distribution to internal testers and developers using TestFlight.

- **TestFlight and App Store**
  The exported application archive is suitable for distribution to external testers using TestFlight and for release to the App Store.



Untitled Workflow
XcodeCloudDemo DEV

General
Environment

Start Conditions
Branch Changes

Actions
Archive - iOS
Test - iOS
Analyze - iOS
Build - iOS

Post-Actions

Archive - iOS

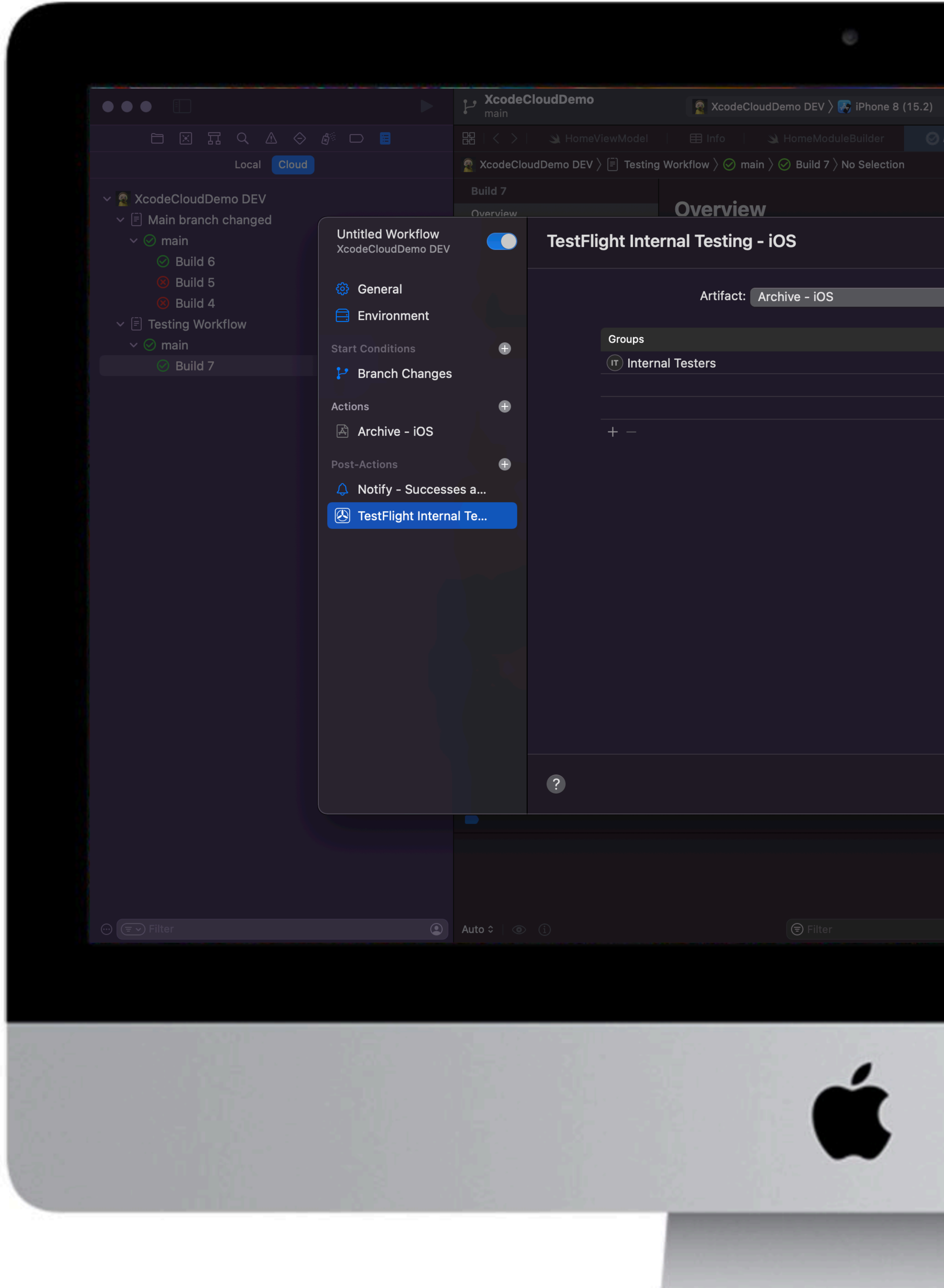
Platform: iOS
Scheme: XcodeCloudDemo DEV
Deployment Preparation: [✓ None]
TestFlight (Internal Testing Only)
Recommended for pull requests and development branches.
TestFlight and App Store
Recommended for release branches. External testing is subject to beta app review.

Workflow/Post-actions

Actions that take place after building.

Setting up custom notifications
Xcode Cloud can send notifications to email or Slack when a build succeeds or fails.

Publish to TestFlight
Xcode Cloud can distribute a new version of the application in TestFlight for both internal and external testers.



Untitled Workflow
XcodeCloudDemo DEV

General
Environment
Start Conditions
Branch Changes
Actions
Archive - iOS
Post-Actions
Notify - Successes a...
TestFlight Internal Te...

TestFlight Internal Testing - iOS

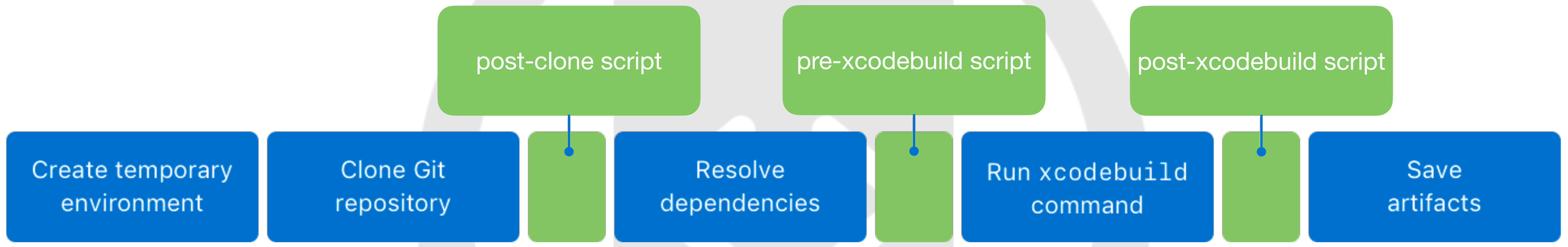
Artifact: Archive - iOS

Groups
Internal Testers

Custom build scripts

These are your custom shell scripts with which you can extend the functionality of Xcode Cloud.

Xcode Cloud recognises three different types of scripts:



Important
- You cannot gain administrator rights using **sudo**.
- Files you create with scripts are not available to other scripts – Xcode Cloud deletes all files created by scripts.

Custom build scripts

To create the scripts you need:

1. Script folder

Create a folder called **ci_scripts** in the project root.

2. Shell Script

Create Shell Script using Xcode template without adding it to the target.

3. Choose a script type

Name the script depending on its type:
- ci_post_clone.sh,
- ci_pre_xcodebuild.sh,
- ci_post_xcodebuild.sh.

4. Make the script executable

From the terminal go to the ci_scripts folder, and make the script executable by running the command:
chmod +x ci_post_clone.sh *(or another script name)*

5. Add the script

Add the script to the file, including #!/bin/sh first line.

6. Commit the script

Commit the script in the repository.

Custom build scripts

Add resources to the CI scripts

Custom build scripts run in a temporary build environment where the source code may not be available. Therefore, all resources accessed by the scripts must be placed in the ci_scripts directory.

If you need to edit a specific file associated with your source code, you can create a symbolic link to the file in the ci_scripts directory.

! **The script files should always be directly in the ci_scripts folder.**

Access environment variables

Environment variables make the script as flexible as possible. You can use your own custom environment variables, for example, you can put an API key in there which will be used by the script to send logs to the server. Also, Xcode Cloud sends already prepared environment variables.

```
if [[ -n $CI_PULL_REQUEST_NUMBER ]];
then
    echo "This build started from a pull request"
fi
```

The list of prepared variables can be seen at: https://developer.apple.com/documentation/xcode/environment-variable-reference

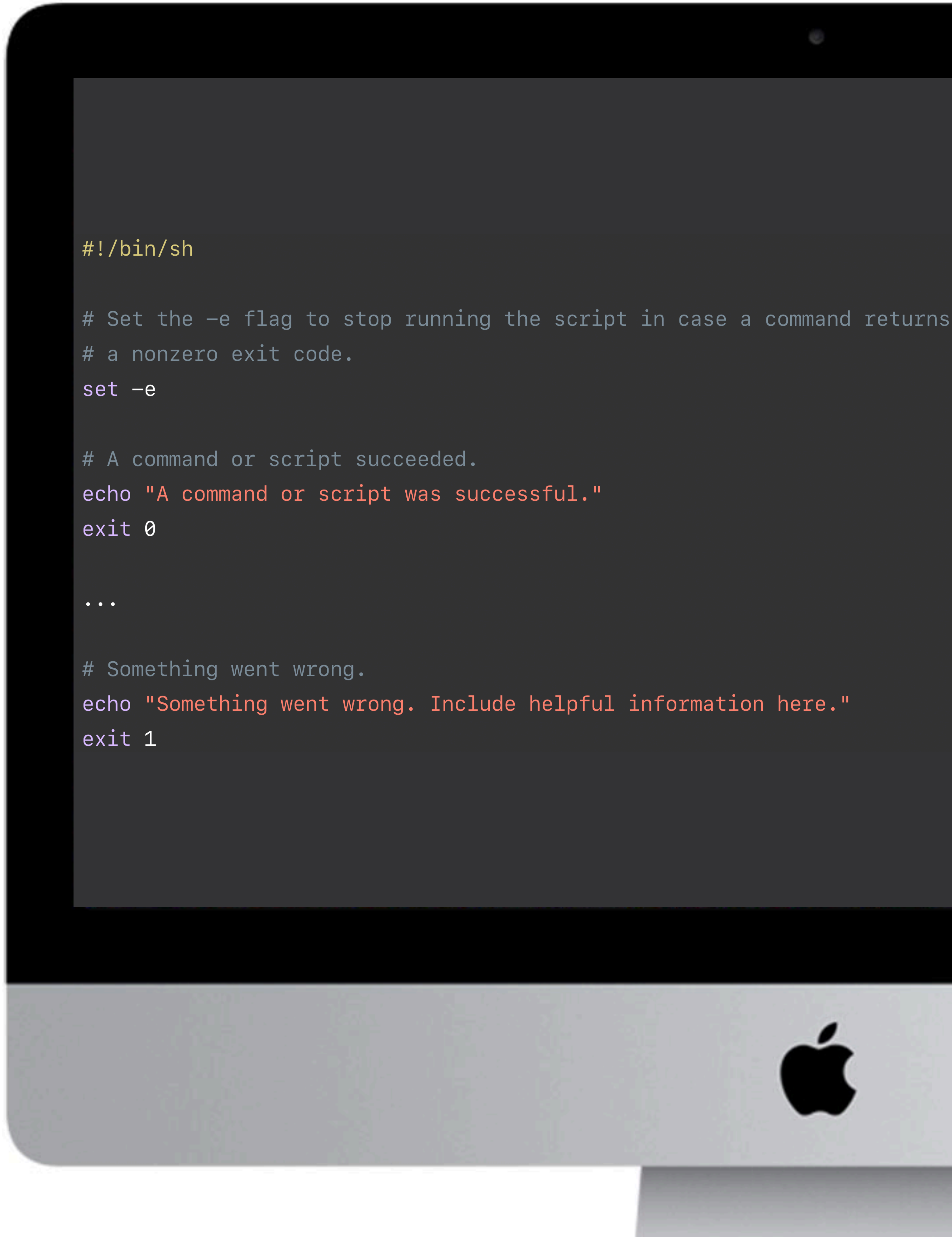
Custom build scripts

Debug information

The logs from your script appear in the build report's build logs, which can be useful for debugging. But it's worth remembering that **confidential information shouldn't be logged** unless it's a secret custom environment variable. In the case of secret custom environment variables Xcode Cloud replaces it with (**********) in the build logs.

Write resilient scripts

Custom build scripts can perform important tasks. You can write a script that returns a nonzero exit code if the script fails. This is how you tell Xcode Cloud that something has gone wrong and allow it to complete the build to let you know there is a problem.



```
#!/bin/sh

# Set the -e flag to stop running the script in case a command returns
# a nonzero exit code.
set -e

# A command or script succeeded.
echo "A command or script was successful."
exit 0

...

# Something went wrong.
echo "Something went wrong. Include helpful information here."
exit 1
```

Dependencies

Swift Packages + Xcode Cloud

Xcode Cloud supports public packages managed by Git out of the box. However, if the package is private, access to the private repository must be granted by Xcode Cloud.

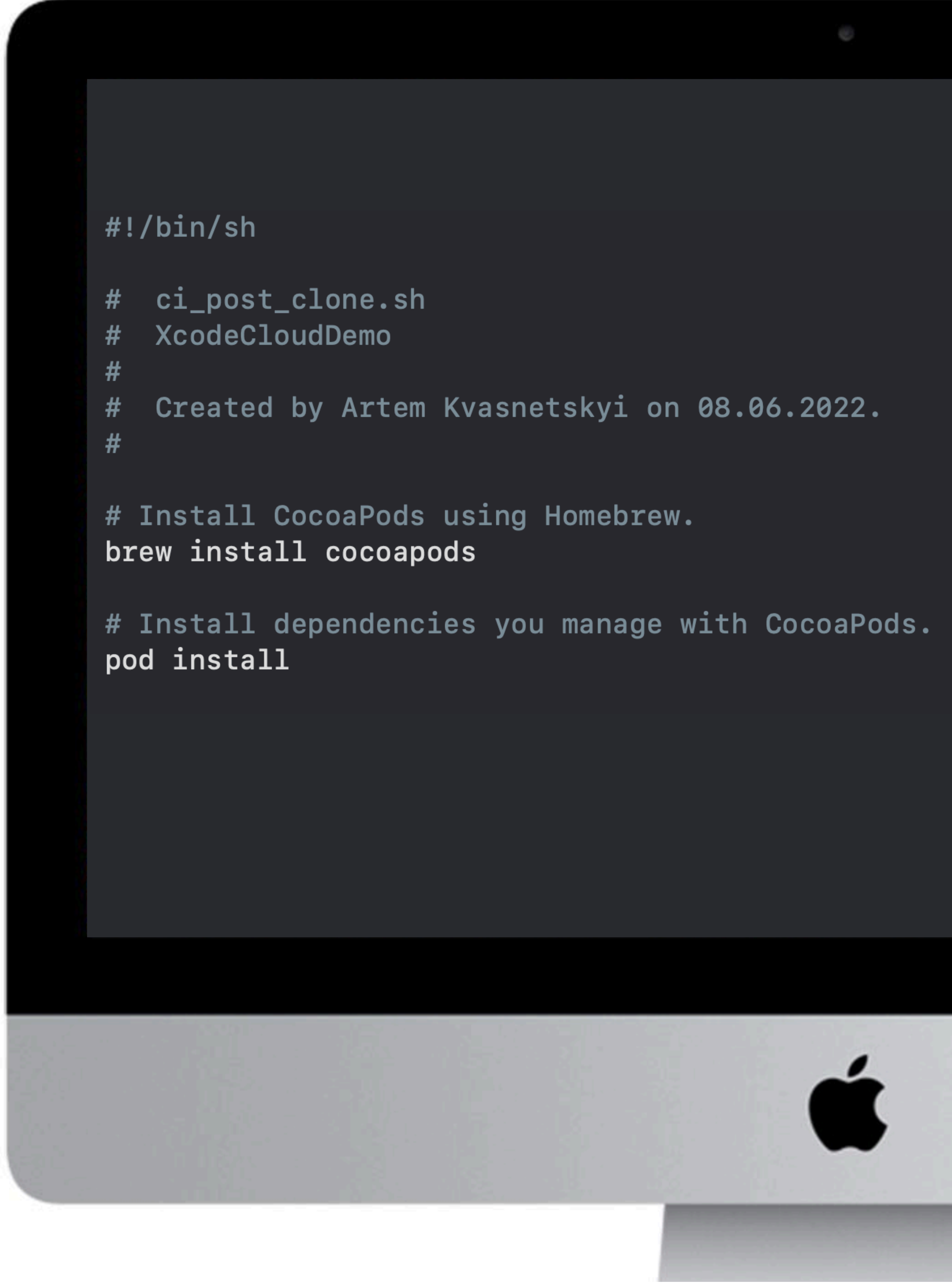
In order for Xcode Cloud to allow SPM dependencies your **Package.resolved file must be committed.**

! **You cannot connect Xcode Cloud to more than one account or instance of the same SCM provider.**

CocoaPods / Carthage + Xcode Cloud

The temporary environment does not include any third-party tools other than Homebrew. You can use it to install CocoaPods or Carthage.

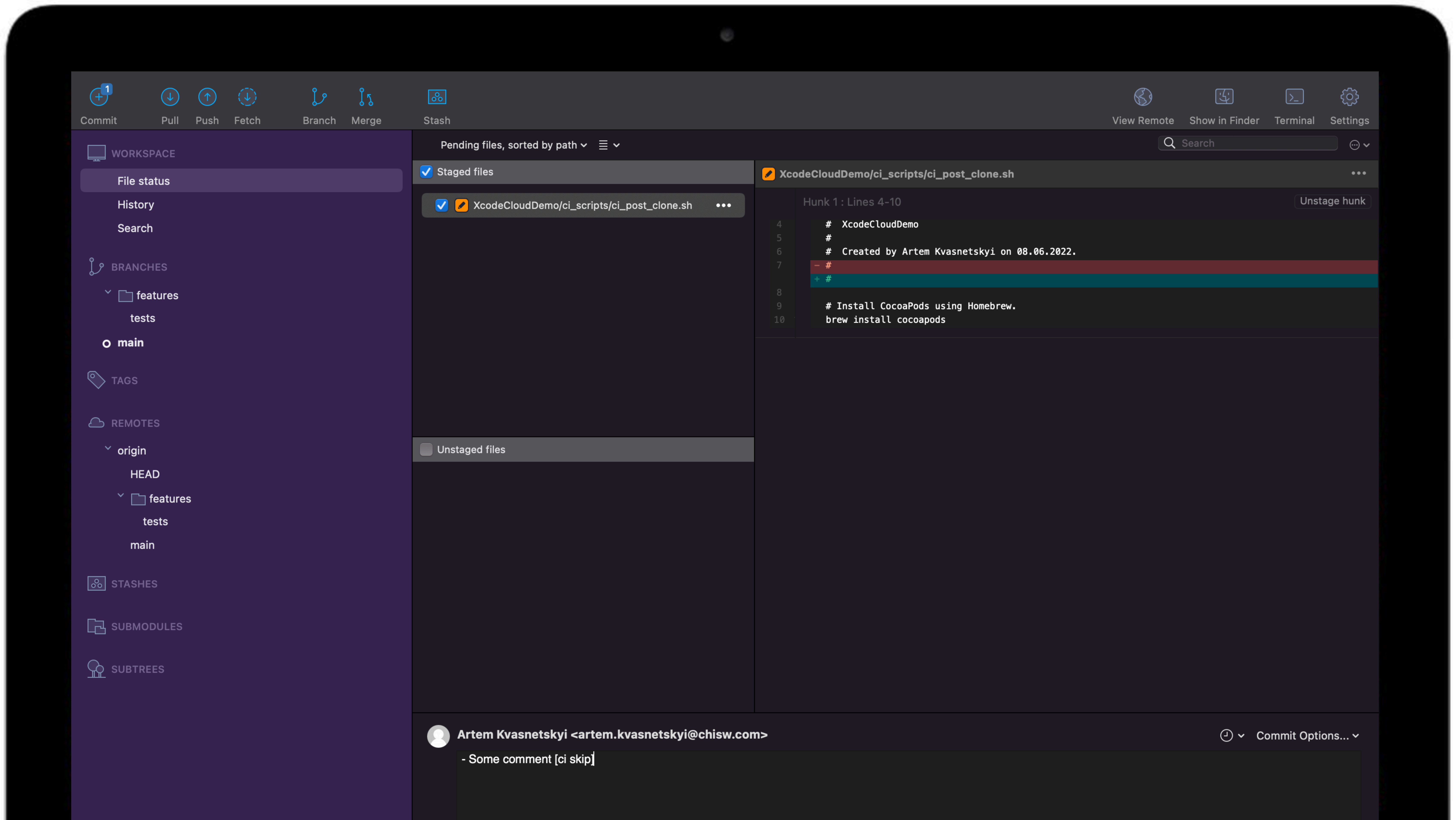
To use Cocoapods, your **Podfile and Podfile.lock must be committed.**



```
#!/bin/sh

#  ci_post_clone.sh
#  XcodeCloudDemo
#
#  Created by Artem Kvasnetskyi on 08.06.2022.
#

# Install CocoaPods using Homebrew.
brew install cocoapods

# Install dependencies you manage with CocoaPods.
pod install
```

Ignore Changes

Xcode Cloud knows how to ignore certain changes in Git. To avoid triggering a workflow related to branch changes, when writing a commit comment, write [ci skip] at the end.



XcodeCloudDemo/ci_scripts/ci_post_clone.sh

```
#  XcodeCloudDemo
#
#  Created by Artem Kvasnetskyi on 08.06.2022.
- #
+ #

# Install CocoaPods using Homebrew.
brew install cocoapods
```

Artem Kvasnetskyi <artem.kvasnetskyi@chisw.com>

- Some comment [ci skip]

Build Number

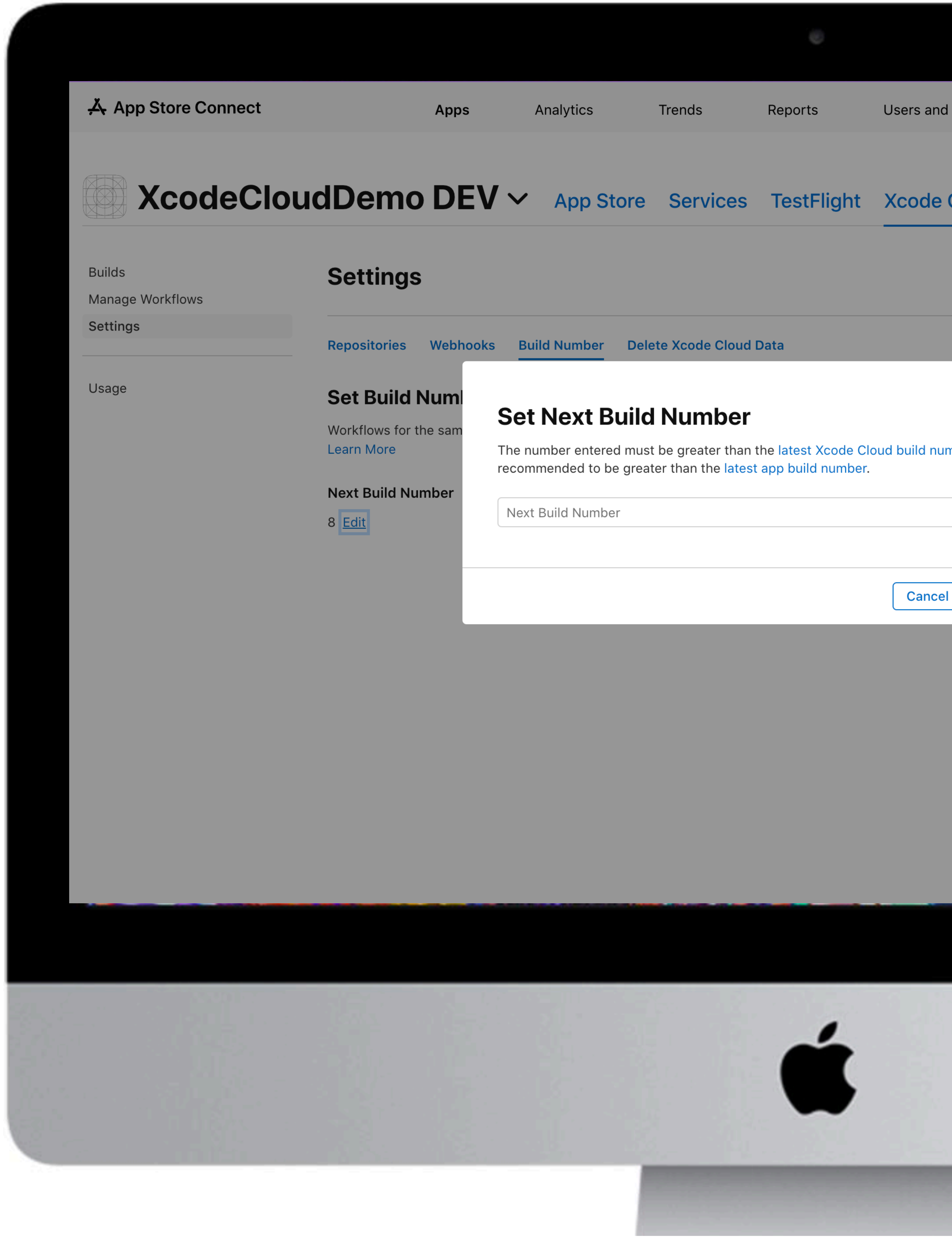
Xcode Cloud assigns a number to each build, starting with 1, and automatically increments it.

You may therefore have a problem when developing applications for the Mac. You need to set up the build number so that it is constantly incrementing. App Store Connect is used to solve this situation.

To set up the next build number:

- Go to your app page on the App Store Connect.
- Click the Xcode Cloud tab and select Settings.
- Click the Build Number tab under Settings.
- Click the Edit button next to Next Build Number.
- Enter the new build number and save your changes.

! **For this you need the Admin or App Manager role.**



Set Build Numl

Workflows for the sam
Learn More

Next Build Number
8 Edit

Set Next Build Number

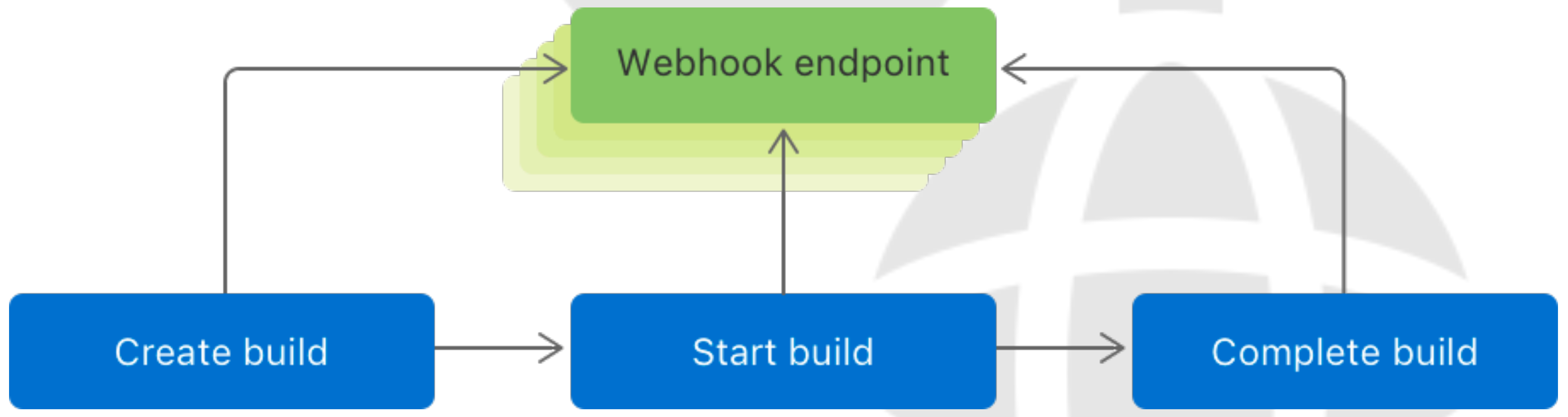
The number entered must be greater than the latest Xcode Cloud build num
recommended to be greater than the latest app build number.

Next Build Number

Configuring webhooks in Xcode Cloud

You can connect up to five custom services that can somehow react to Xcode Cloud events.

Xcode Cloud sends an HTTP request to a given endpoint every time it creates, starts and completes a build. In turn, the service must send an HTTP status code in response. If it returns a server error that can be repeated or Xcode Cloud does not receive a response within 30 seconds, it resends the request until it receives a successful response.



What the JSON request from Xcode Cloud looks like can be seen at:
https://developer.apple.com/documentation/xcode/configuring-webhooks-in-xcode-cloud

Configuring webhooks in Xcode Cloud

To create a webhook in Xcode Cloud you need:

App Store Connect

1. Go to Xcode Cloud in App Store Connect.

Settings > Webhook

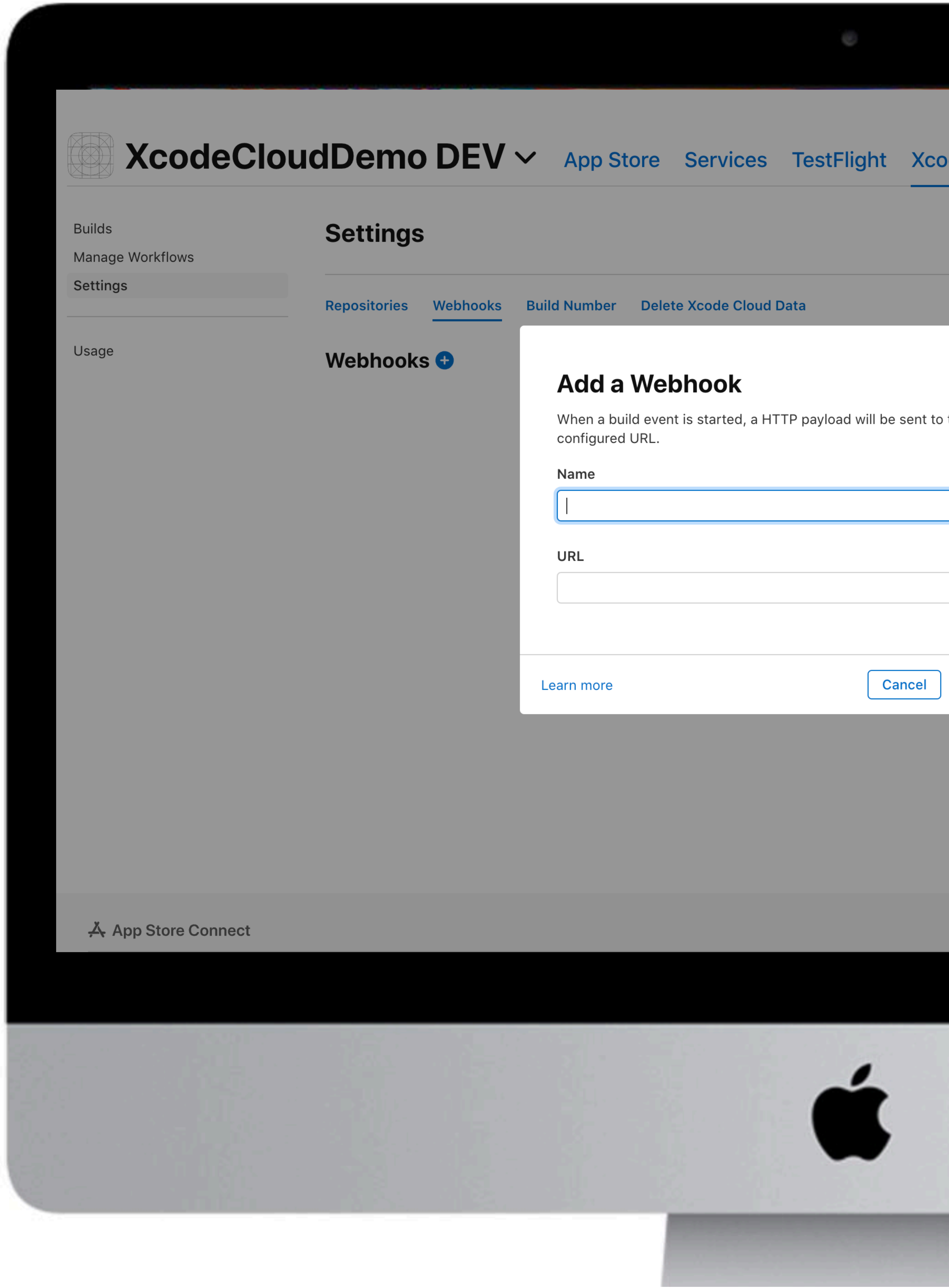
2. In the sidebar, select Settings > Webhooks > Add button.

Unique name

3. Enter a unique name for your webhook.

Service URL

4. Enter the URL of a service that can receive and handle HTTPS requests from Xcode Cloud.



When a build event is started, a HTTP payload will be sent to
configured URL.

Disconnecting the project from Xcode Cloud

Deleting data from Xcode Cloud:
In Xcode Navigator, go to Report, right-click on your app, and select Delete Xcode Cloud Data. After that, go to App Store Connect to the app you want, select Settings and click Delete Xcode Cloud Data.

These apps will no longer be available immediately and will be removed from the Apple system within 30 days.

Disconnect Xcode Cloud from Slack:
Open Slack workspace in the Slack app and select Settings & administration > Manage apps. Select "Apps" in the sidebar, then select the Xcode Cloud app. Click "Remove App".

Disconnecting Xcode Cloud from the repository:
• To disable Bitbucket Server, GitHub Enterprise, or self-managed GitLab:
Go to App Store Connect under Users and Access, select Xcode Cloud tab, hover over the SCM provider, click Remove.
• To disable Bitbucket, GitHub, or GitLab: Go to App Store Connect under Users and Access, select Xcode Cloud tab, select Integrations in the sidebar, click Unlink next to the SCM provider.

You then need to disable the Personal Access Tokens or Apps that allowed Xcode Cloud access to the repository. Disabling depends on the SCM provider. For more detailed instructions:
https://developer.apple.com/documentation/xcode/removing-your-project-from-xcode-cloud

Pricing

25 hours/month

$14.99/month

Free through December 2023

100 hours/month

$49.99/month

250 hours/month

$99.99/month

1000 hours/month

$399.99/month

Thanks for your attention