Setting Up a Project for Xcode Cloud

A step-by-step guide to connecting your app and running your first build

Xcode Cloud is Apple's continuous integration and delivery (CI/CD) service built into Xcode and App Store Connect. It automatically builds, tests, and distributes your app in the cloud whenever you push changes, so you can catch issues early and ship to TestFlight and the App Store faster. This guide walks through connecting an existing Xcode project to Xcode Cloud and creating your first workflow.

Before You Start

- A Mac with a current version of **Xcode** installed.
- Membership in the **Apple Developer Program** (Xcode Cloud is included; usage is metered by compute hours).
- The **Account Holder** or **Admin** role in App Store Connect (needed to grant access the first time).
- Your project in a supported **source control** system (GitHub, GitLab, Bitbucket, or similar) and committed to a remote repository.
- You are signed in to Xcode with your Apple ID under *Settings* > *Accounts*.

Step-by-Step Setup

Step 1 — Open your project and start onboarding

Open your app project in Xcode. Go to the **Report navigator** (the speech-bubble icon in the left sidebar) and select the **Cloud** tab, then click **Get Started**. Alternatively, use the menu **Product > Xcode Cloud > Create Workflow**.

Step 2 — Select the app or framework to connect

Xcode shows the apps and frameworks in your workspace. Choose the product you want Xcode Cloud to build, then continue. Xcode prepares a default workflow based on your project.

Step 3 — Review the suggested workflow

Xcode proposes a starter workflow. A workflow defines *when* a build runs and *what* it does. You can accept the suggestion now and refine it later, or click **Edit Workflow** to customize it before the first run.

Step 4 — Configure the workflow

In the workflow editor you set the key sections:

- **General** — name and describe the workflow.
- **Environment** — pick the Xcode version and macOS version to build on.
- **Start Conditions** — what triggers a run (e.g. a push to a branch, a pull request, or a tag).
- **Actions** — what the build does: **Build**, **Analyze**, **Test**, or **Archive** (add them with the **+** button).
- **Post-Actions** — what happens after a successful build, such as distributing to **TestFlight** or notifying via email/Slack.

Step 5 — Grant access to your source control

Click **Grant Access**. Xcode opens App Store Connect in your browser so you can authorize Xcode Cloud to read your repository on your SCM provider (GitHub, GitLab, Bitbucket, etc.). Install/authorize the Xcode Cloud app for the specific repository when prompted.

Step 6 — Confirm the app in App Store Connect

Xcode Cloud checks whether your app already exists in App Store Connect using the project's **Bundle Identifier**. If it doesn't exist, the assistant creates it for you and asks you to provide an **SKU** (a unique internal identifier you choose) and confirm the app name.

Step 7 — Choose a branch and start the first build

Pick the Git branch Xcode Cloud should build (commonly *main*). Click **Start Build** (or **Complete**). Xcode Cloud provisions a clean macOS environment, clones your repo, and runs the workflow. You can watch progress live in the Cloud tab of the Report navigator.

Step 8 — Review results and iterate

When the build finishes you'll see logs, test results, and any artifacts. Fix issues if it failed, then push a new commit to trigger another run. Edit the workflow any time to add tests, change triggers, or set up distribution.

Tip: keep your first workflow simple

Start with a single Build (or Build + Test) action triggered on your main branch. Once you confirm builds succeed end to end, layer in automated tests and TestFlight distribution. This makes failures much easier to diagnose early on.

Common gotchas

- Make sure the scheme you want to build is marked **Shared** (Product > Scheme > Manage Schemes).
- Code signing is handled by Xcode Cloud automatically, but your bundle ID and team must be set correctly.
- Only the Account Holder/Admin can grant SCM access the first time — line that up before you start.
- Xcode Cloud usage is measured in compute hours; check your plan's included allowance.

Connecting Xcode Cloud to GitHub

Xcode Cloud works directly with GitHub (and GitHub Enterprise Cloud). The connection is made through the **Xcode Cloud GitHub App**, which you authorize once per repository or organization. Here's how to set it up and what to watch for.

1. Push your project to GitHub

Make sure your Xcode project is committed and pushed to a GitHub repository, and that the scheme you want to build is marked **Shared** (Product > Scheme > Manage Schemes > check *Shared*). Commit the .xcscheme file so Xcode Cloud can find it.

2. Start the grant-access step

During workflow setup, when you reach **Grant Access**, Xcode opens App Store Connect in your browser and redirects you to GitHub to install the Xcode Cloud GitHub App.

3. Choose the repository scope

On GitHub, pick whether to grant access to **All repositories** or **Only select repositories**. Selecting just the one repo you're building is the safer choice. You must be an **owner or admin** of the GitHub org/repo to install the app.

4. Authorize and return to Xcode

Approve the installation. GitHub redirects you back to App Store Connect / Xcode, which confirms the repository is now linked. Xcode Cloud can now read your code and report build status back to GitHub.

5. Pick a branch and build

Choose the branch (e.g. *main*) and start your first build, exactly as in the general setup. From now on, pushes and pull requests on the configured branches can trigger Xcode Cloud automatically.

What you get back in GitHub

Once connected, Xcode Cloud posts **build status checks** on your commits and pull requests — so you can see pass/fail directly in the PR and even require a green Xcode Cloud check before merging (via GitHub branch protection rules).

GitHub-specific gotchas

• You need **org owner / repo admin** rights to install the GitHub App the first time.
• For **private** repos, the GitHub App grants Xcode Cloud read access — no personal access tokens needed.
• If your org uses **SSO**, make sure the Xcode Cloud app is authorized for the SAML org.
• To change which repos are accessible later: GitHub > *Settings* > *Applications* > *Installed GitHub Apps* > *Xcode Cloud* > *Configure*.
• Pull requests from **forks** don't trigger builds by default, for security.

After Your First Build

- Add a **Test** action so every push runs your unit/UI tests automatically.

- Add a **TestFlight** post-action to deliver builds to internal testers on each successful archive.
- Create separate workflows for different purposes (e.g. PR validation vs. release builds).
- Use **custom build scripts** (ci_scripts/ci_post_clone.sh, etc.) for dependencies or extra steps.

Sources: Apple Developer — Connect your project to Xcode Cloud & Create practical workflows (WWDC23); createwithswift.com; rootstrap.com; browserstack.com. Interface labels may vary slightly by Xcode version.