Transitioning from Intel to Apple Silicon

A developer's handbook

Universal binaries

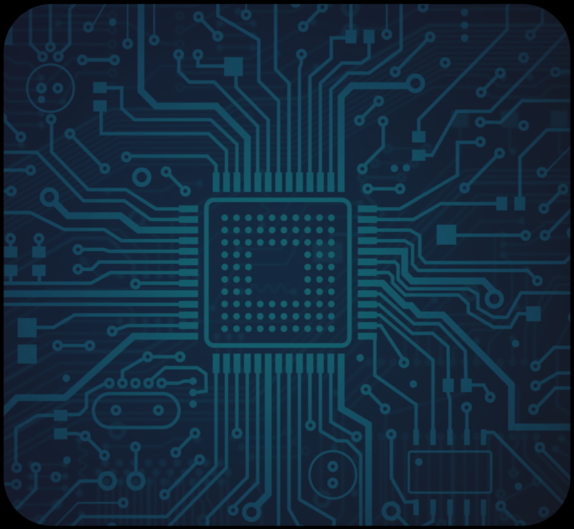
What runs natively on M1?

CI/CD tools

70%

more tips

MacStadium



Introduction

At its Worldwide Developers Conference (WWDC) in June 2020, Apple announced a major shift in their Mac hardware architecture – namely, that a new ARM-based Apple silicon processor is replacing the Intel chip that macOS has been running on for nearly the past two decades.

By November, Apple had announced its first Apple silicon chip for the Mac, the M1, and introduced a new MacBook Pro, MacBook Air, and Mac mini powered by the M1 chip. By moving it off of Intel, Apple is aligning the Mac with its other products like the iPhone, iPad, and Apple Watch, all of which are based on Apple's "system on a chip (SoC)" processors.

Most importantly, as part of these announcements, Apple set a two-year timeline for migrating their fleet from Intel to Apple silicon, calling the transition "the biggest leap ever for the Mac."

Rosetta 2 enables an Apple silicon-based Mac to run applications built for an Intel Mac, and the latest versions of Xcode include Apple silicon native compilers, editors, and debugging tools. However, as most developers know (or are finding out), a transition of this scale is not always as smooth sailing as one may hope.

We are excited about Apple silicon, and MacStadium was the first cloud provider to offer M1 minis in our data centers. Having worked with Mac development teams for over a decade, we understand the challenges that migrating from Intel to Apple silicon could present, and we want to help make the transition a little smoother.

In this eBook, we'll look at what this change in architecture means from both users' and developers' perspectives. We'll look at a variety of different paths to developing and delivering software on Apple silicon that teams may take as they transition to this new architecture. Finally, we'll help navigate your transition by identifying core factors related to your codebase, its dependencies, and the specific Apple platform you are targeting that may provide guidance to which path to take on your journey from Intel to Apple silicon.

Contents

"With its powerful features and industry-leading performance, Apple silicon will make the Mac stronger and more capable than ever."

Tim Cook, Apple CEO

In a state of transition

When Apple first announced the shift to ARM-based Apple silicon architecture in June of 2020, they unveiled a two-year transition plan to ease the shift for Apple's developer community. Now that roughly half of that window of time has elapsed, the landscape is looking quite a bit different for teams looking to make the transition than it did a year ago.

Most notably, the macOS and iOS-focused developer community has been diligently converting macOS-based development tools and their various dependencies to run natively on Apple silicon. This means that your path to conversion is likely becoming incrementally shorter, as one of the most common blockers for teams making the transition is the fact that they often have to wait for upstream dependencies to be converted first because all included code – both that which your team writes and any dependencies included in the project – must be compiled for Apple silicon.

Why the shift from Intel to ARM-based architecture?

ARM architecture is not entirely new. In fact, iOS has been running on ARM-based architecture since its inception in 2007. It has traditionally been very attractive for mobile devices because of its excellent efficiency in terms of energy consumption and processing speed. In that sense, Apple's M1 chip design represents the uniting of two different worlds in an effort to bring this same energy efficiency and processing speed to Mac. And, because of this coming together of worlds in a hardware sense, we are also seeing software features follow suit, such as iOS applications now being able to run natively on Mac hardware.

What from the Intel world runs on Apple silicon?

Interestingly, most macOS software built for Intel – especially software that Apple produces directly – will "just work" on Apple silicon with the help of emulation via Rosetta 2. Rosetta 2 is essentially an interpreter that reads binary intended to be read by Intel machines and translates it to the variety of instructions Apple silicon is anticipating. This generally allows the software to run quite well. In fact, the enormous increase in processing power that the Apple silicon provides often allows emulated applications to run faster on Apple silicon than they did on Intel-based machines.

However, because emulation requires an additional layer of processing in order to execute this translation of instructions, it incurs a penalty in terms of memory usage and top-end processing speed. This is particularly true in the case of virtualized macOS, as is commonly used in modern CI/CD pipelines for iOS and macOS.

What tools are Apple silicon ready?

For a detailed breakdown of what currently runs natively on Apple silicon, MacStadium is a proud sponsor of isapplesiliconready.com, an actively maintained list of macOS products and tools that have already been converted, as well as those slated for conversion.

Core Apple developer resources

The following table highlights the massive amount of work that the macOS dev community has put into facilitating this transition in architectures. As you can see, core Apple developer resources are unlikely to be a blocker for your team as you make this transition.

Product	Runs natively on Apple silicon	Runs in Rosetta 2 on Apple silicon
macOS 11+	✓	✓
Xcode 12+	✓	✓
Homebrew	✓	✓
CocoaPods	✓	✓
Fastlane	✓	✓

CI/CD tools

In addition to the core resources above, you'll also want to consider whether or not your CI runner has been converted to run natively on Apple silicon.

Product	Runs natively on Apple silicon	Runs in Rosetta 2 on Apple silicon
Jenkins	✓	✓
GitHub Actions	✗ (.NET 6.0 requirement released)	✓
GitLab Runner	✓	✓
Buildkite	✓	✓
Azure DevOps	✗ (.NET 6.0 requirement released)	✓
Bamboo	✗	✓

as of December 13, 2021

What is the motivation for running builds and tests on Apple silicon?

There are essentially two different perspectives about migrating software development to Apple silicon. And each of these camps has a different set of motivating factors for both when and why they will ultimately execute the change.

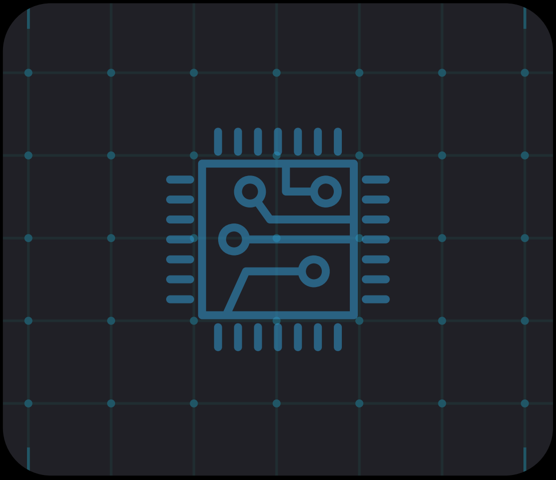
Mac app builders

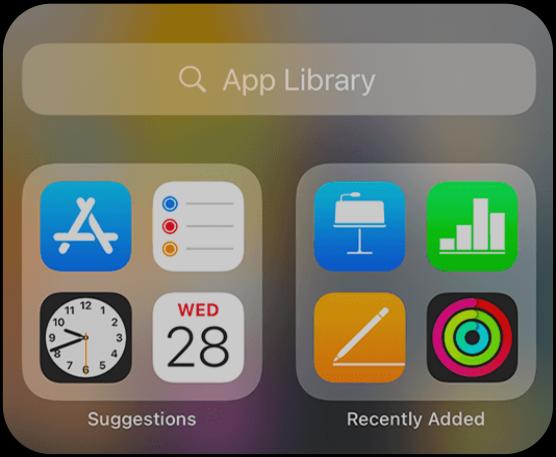
The first group is the most obvious. Teams that are building applications specifically for macOS will need to build their code on Apple silicon in order for their end-users to be able to run that same code on their local M1 Mac. Ultimately, as Apple migrates all of its Macs to Apple silicon and away from Intel, apps will need to be built for Apple silicon. And even during the two-year transition, arguably macOS apps will perform better if they do not rely on Rosetta 2.

iOS app builders

Conversely, teams that are targeting other platforms, such as iOS, will have more peripheral motivations for making the change. In the near term, these include the opportunity to benefit from Apple silicon's massive increase in processing speed and power. This will allow builds and tests to run much faster, which in turn means that development teams will lose less time waiting for the execution of CI/CD workflows.

However, these same iOS dev teams will soon have a more pressing motivation for building and testing their software on Apple silicon. Namely, as part of Apple's transition process, new Xcode releases will require Apple silicon. This means that in order to continue to produce new and exciting applications that take advantage of the latest available user enhancements and security features, teams will need to convert their development and CI/CD to target Apple silicon.





How will converting to Apple silicon impact development?

Regardless of the platform that you are targeting – macOS, iOS, or anything else – all upstream dependencies required by your Xcode project must first be built for Apple silicon. Until that has been done, the best you will be able to achieve is emulation with Rosetta 2.

Beyond that requirement, which platform you are targeting becomes important. There are different considerations when building apps for macOS itself versus building for any of the other platforms that require macOS for development but not directly for delivery of an end product (iOS, watchOS, etc.).

Impacts specific to macOS development

Developing for macOS, in particular, presents the unique challenge of delivering an end product for a user-base that is spread across two distinct architectures – Intel and Apple silicon. In the near term, the majority of Mac users will be on existing Intel machines, but as new M1 hardware is made available, a steadily increasing percentage of users will move to Apple silicon.

The core takeaway here is that macOS apps need to be built for both platforms because your end-users are currently working on both types of machines. The building of macOS apps for Intel-based and Apple silicon can be achieved in one of two ways: either target both Intel-based Macs and Apple silicon separately when compiling binaries or target something called a universal binary.

A universal binary runs natively on both Apple silicon and Intel-based Mac computers, because it contains executable code for both architectures.

Universal binaries for macOS

Universal binaries contain executable code for both Apple silicon and Intel-based architectures. This means that when a user goes to download your application from the internet if you provide a universal binary, he or she won’t need to select the correct version in order to successfully install the application. Instead, the user will simply download the supplied .dmg file and when it runs on the user’s machine, it will determine which architecture it is running on, and it will execute the correct binary accordingly.

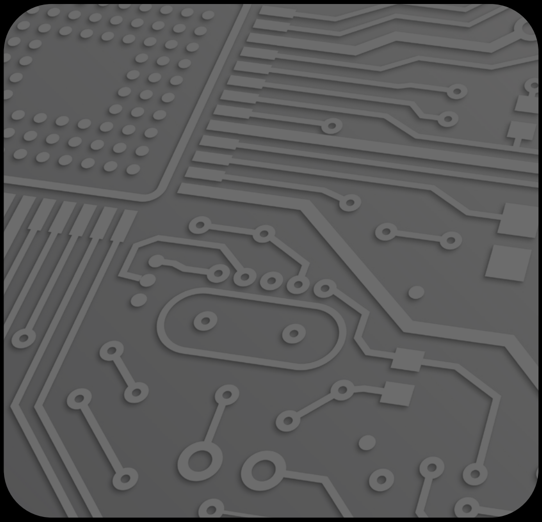
Impacts to development for other Apple platforms

Production builds for iOS – and other non-Mac Apple platforms – will result in the same binary because the physical hardware these products run on (and thus the binary they can interpret) is fundamentally unchanged by this update.

However, because most iOS development workflows involve Xcode simulators, non-production build artifacts will only run on the simulator associated with the architecture the build was executed on. That is, to use the Xcode simulator on an Apple silicon machine, the build must be executed on an Apple silicon machine as well, and vice versa.

Codebases that interact with low-level architecture features

If you're interacting with low-level APIs, like the Metal API for example, there will likely be specific changes that will need to be made to your codebase. Apple has specifically documented such required changes.



What implications will Apple silicon have for CI/CD pipelines?

In addition to any changes you'll need to make directly to your codebase, you'll also need to update your build infrastructure to include Apple silicon hardware. You may also want to consider selecting a CI runner that runs natively on Apple silicon if your current solution hasn't yet been converted.

Adding Apple silicon build infrastructure efficiently

This transition to Apple silicon is proving to be an opportunity for many teams targeting iOS and macOS builds to rethink where their macOS compute resources live. In particular, many teams are looking to the cloud for the sake of simplifying and reducing the cost of buying and managing a bank of Apple silicon servers. A hybrid cloud environment is particularly appealing because these Apple silicon servers can run alongside existing Intel-based machines in the near term, ultimately replacing the existing Intel-based servers entirely.

Building on Intel and M1 with Orka 2.0

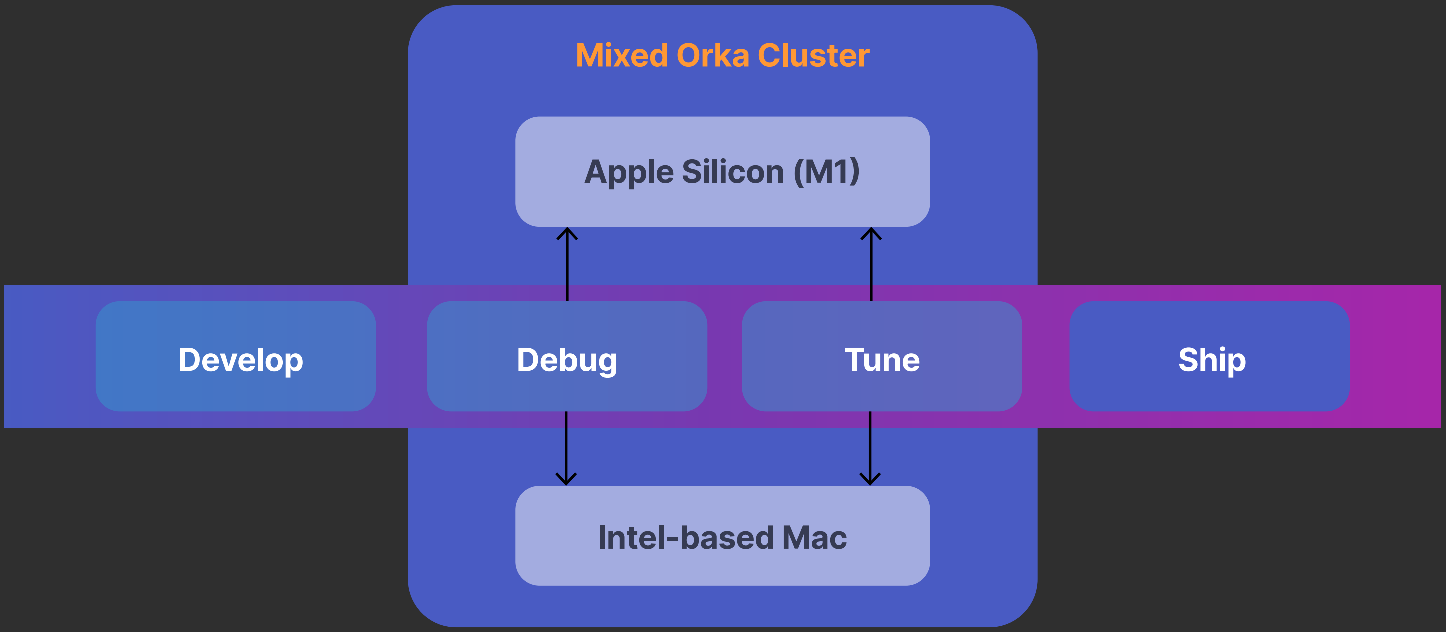
MacStadium's Orka offers a unique opportunity to run fully ephemeral CI/CD processes on mixed clusters consisting of both Intel-based and M1 Macs. This means that you can spin up single-use macOS VMs based on either architecture type from within your CI/CD pipeline. From the same Orka environment, you can run both Intel-based Mac and Apple silicon builds, helping developers transition from one architecture to the other.

And once dev teams have completed the transition to Apple silicon, converting an entire Orka cluster to M1 will be as simple as opening a ticket with MacStadium. For teams targeting macOS, a mixed cluster in Orka is an excellent option for delivering an installation-friendly universal binary, as shown below.

Orka is ideal for facilitating an extended transition period from Intel-based to Apple silicon servers. Because you can have a mixed Orka cluster that contains both Intel and Apple silicon-based nodes, you can maintain the integrity of your CI/CD processes for Intel while also beginning to run parallel builds targeting Apple silicon.

Transition from Intel to M1 on Orka

MacStadium solution engineers can help guide you through transitioning from Intel to M1. From bare metal to Orka virtualization, we have a solution for your dev team. Contact us now for more info or to get started. Contact us now for more info or to get started.



Maintaining or updating your CI runner

Depending on the CI runner you are currently utilizing in your pipeline, you may want to consider upgrading to one that runs natively on Apple silicon, or wait to transition to Apple silicon until your CI runner's scheduled update, as most tools have established timelines if they aren't already running natively.

This may be worth the effort or the wait because early efforts to emulate CI runners with Rosetta 2 have proven to be non-trivial affairs. In particular, when the CI runner is stood up in emulation, the processes that it kicks off, such as calling xcodebuild, for example, must then break out of emulation in order to successfully execute a native build, which has proven to be a challenge for many early adopters.

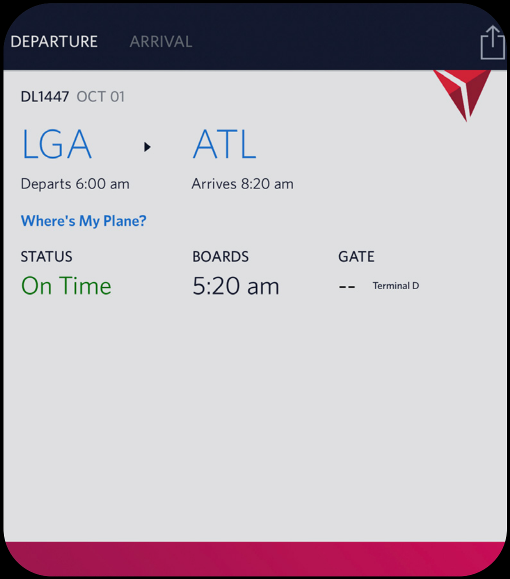
If your current CI tooling works natively on Apple silicon, then you can clearly move forward with the transition. But if that is not the case, and you don't want to go to the trouble of changing your CI stack, you may want to wait for a release of a natively-running Apple silicon version.

Case Study: Delta Air Lines

Delta's iOS build and CI/CD pipeline was built on a fleet of bare metal 2012 Intel-based Mac minis housed in MacStadium's data centers. These machines were still performing well; however, Apple's shift from Intel to ARM-based Apple silicon meant that as newer versions of macOS and Xcode were released, the older hardware would not be supported.

The Fly Delta app team was faced with a decision: upgrade their Intel-based machines to the latest 2018 version of the Mac mini, or switch to Apple silicon M1 minis. Knowing that Apple had established a two-year timeline for full migration from Intel to Apple silicon, the team decided to "just take the plunge and go forward with the M1 machines."

Read the full case study to learn how their transition to M1 resulted in builds that are twice as fast as the legacy 2012 minis.



Planning the transition to Apple silicon

Mac hardware architecture is undergoing a fundamental change that will require some degree of accommodation on the part of all developers in the Apple ecosphere. Teams building macOS apps will need to compile their binary on Apple silicon directly in order for end-users to be able to run their software natively on their own Apple silicon Macs. Teams building for iOS and other Apple platforms will need to make the transition as well, as new Xcode versions – and the feature and security upgrades that come with them – will only be available for Apple silicon.

Because this is such a fundamental and temporally-driven shift, it is the perfect opportunity to explore how and where to host your Mac build infrastructure, as your pool of available macOS compute resources will first need to be augmented, and then ultimately be replaced, with Apple silicon.

MacStadium is here to help. In addition to our virtualization platform, Orka, we offer bare metal Apple silicon servers in each of our data centers. Ready to discuss your migration plans from Intel-based servers to the new Apple silicon? Contact MacStadium's sales engineers to discuss your particular use case and to learn more about our Mac cloud solutions.

Ready to move to the cloud?

If you are running Mac build machines in-house, now may be the perfect time to move your build infrastructure to cloud-hosted Mac servers.

- No large capital expenditures to replace aging Intel machines with new M1 Macs
- Easy development transition with virtualization solutions like Orka that support both architectures
- Predictable monthly costs, fast and easy scalability, and 24/7 remote hands support take the stress out of managing Mac hardware

MacStadium

www.macstadium.com

www.macstadium.com/orka

sales@macstadium.com