Behind the scenes: How we host Ars Technica, part 1

Enlarge / Take a peek inside the Ars vault with us!
Aurich Lawson | Getty Images

reader comments

22 with

A bit over three years ago, just before COVID hit, we ran a long piece on the tools and tricks that make Ars function without a physical office. Ars has spent decades perfecting how to get things done as a distributed remote workforce, and as it turns out, we were even more fortunate than we realized because that distributed nature made working through the pandemic more or less a non-event for us. While other companies were scrambling to get work-from-home arranged for their employees, we kept on trucking without needing to do anything different.

However, there *was* a significant change that Ars went through right around the time that article was published. January 2020 marked our transition away from physical infrastructure and into a wholly cloud-based hosting environment. After years of great service from the folks at Server Central (now Deft), the time had come for a leap into the clouds—and leap we did.

There were a few big reasons to make the change, but the ones that mattered most were feature- and cost-related. Ars fiercely believes in running its own tech stack, mainly because we can iterate new features faster that way, and our community platform is unique among other Condé Nast brands. So when the rest of the company was either moving to or already

on Amazon Web Services (AWS), we could hop on the bandwagon and take advantage of Condé's enterprise pricing. That—combined with no longer having to maintain physical reserve infrastructure to absorb big traffic spikes and being able to rely on scaling—fundamentally changed the equation for us.

In addition to cost, we also jumped at the chance to rearchitect how the Ars Technica website and its components were structured and served. We were using a "virtual private cloud" setup at our previous hosting—it was a pile of dedicated physical servers running VMWare vSphere—but rolling everything into AWS gave us the opportunity to reassess the site and adopt some solid reference architecture.

Advertisement

Cloudy with a chance of infrastructure

And now, with that redesign having been functional and stable for a couple of years and a few billion page views (really!), we want to invite you all behind the curtain to peek at how we keep a major site like Ars online and functional. This article will be the first in a four-part series on how Ars Technica works—we'll examine both the basic technology choices that power Ars and the software with which we hook everything together.

This first piece, which we're embarking on now, will look at the setup from a high level and then focus on the actual technology components—we'll show the building blocks and how those blocks are arranged. Another week, we'll follow up with a more detailed look at the applications that run Ars and how those applications fit together within the infrastructure; after that, we'll dig into the development environment and look at how Ars Tech Director Jason Marlin creates and deploys

changes to the site.

Finally, in part 4, we'll take a bit of a peek into the future. There are some changes that we're thinking of making—the lure (and price!) of 64-bit ARM offerings is a powerful thing—and in part 4, we'll look at that stuff and talk about our upcoming plans to migrate to it.

Ars Technica: What we're doing

But before we look at what we want to do *tomorrow,* let's look at what we're doing *today.* Gird your loins, dear readers, and let's dive in.

To start, here's a block diagram of the specific AWS services Ars uses. It's a relatively simple way to represent a complex interlinked structure:

Enlarge / A high-level diagram of the Ars AWS setup.
Lee Hutchinson

Ars leans on multiple pieces of the AWS tech stack. We're dependent on an Application Load Balancer (ALB) to first route incoming visitor traffic to the appropriate Ars back-end service (more on those services in part 2). Downstream of the ALB, we use two services called Elastic Container Services (ECS) and Fargate in conjunction with each other to spin up Docker-like containers to do work. Another service, Lambda, is used to run cron jobs for the WordPress application that forms the core of the Ars website (yes, Ars runs WordPress—we'll get into that in part 2).

https://arstechnica.com/?p=1952776