We are still early with the cloud: why software development is overdue for a change

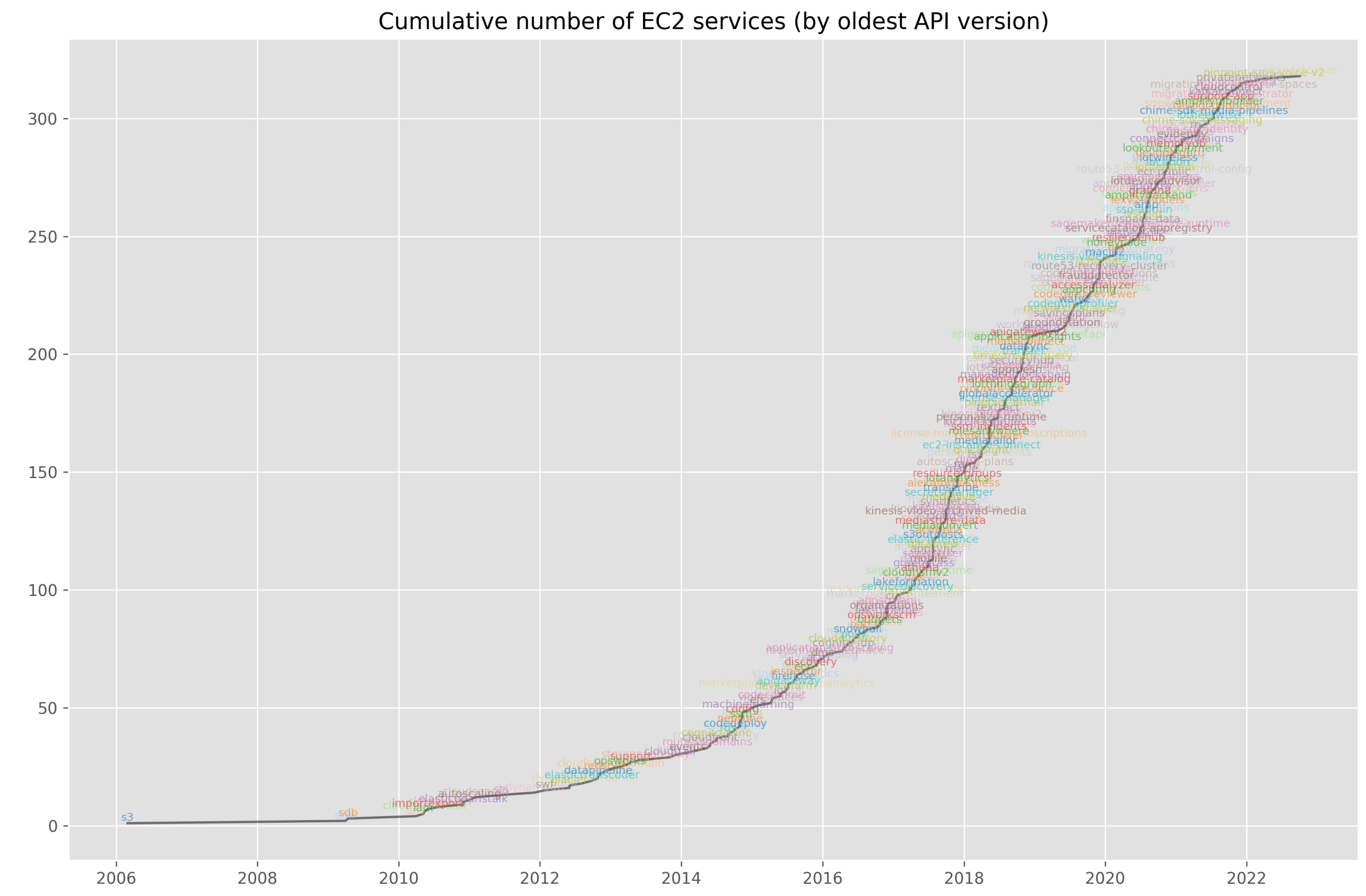
2022-10-19

This is is in many respects a successor to a blog post I wrote last year about what I want from software infrastructure, but the ideas morphed in my head into something sort of wider.

The genesis

I encountered AWS in 2006 or 2007 and remember thinking that it's crazy — why would anyone want to put their stuff in someone else's data center? But only a couple of years later, I was running a bunch of stuff on top of AWS.

Back then, AWS had something like two services: EC2 and S3. Today, that number is closer to 350: [1]



And today, the cloud is obviously here… I mean, despite what some people may think about cloud adoption [2] it's clear that building technology is vastly different today than it was a decade ago, and the cloud deserves a big part of the credits for it.

I think we might be early though?

In some sort of theoretical abstract platonic form type thing, the cloud should offer us

- Infinite scalability
- Less time spent on infrastructure
- Fewer constraints
- Lower costs.

Here's a random assortment of things I feel like we *should* have, if the cloud had truly delivered. But we don't:

- When I compile code, I want to fire up 1000 serverless container and compile tiny parts of my code in parallel.

- When I run tests, I want to parallelize all of them. Or define a grid with a 1000 combinations of parameters, or whatever.
- I never ever again want to think about IP rules. I want to tell the cloud to connect service A and B!
- Why is Bob in the ops team sending the engineers a bunch of shell commands they need to run to update their dev environment to support the latest Frobnicator version? For the third time this month?
- Why do I need to SSH into a CI runner to debug some test failure that I can't repro locally?

I could go on!

The current state doesn't strike me as a slam dunk improvement along every axis. Most egregiously, why have the feedback loops writing code become *longer*? And the environment difference between local and prod *larger*? I don't know…I look at modern programming and cloud computing and it feels like we have most of the building blocks, but we're still so far from it? I can't help but getting the feeling that we basically just did this thing so far: [3]

Docker is an incredible piece of technology for many reasons. But just using it the way the meme above implies, seems as if we didn't fundamentally change development to take advantage of this new magic technology, we just sort of just think of it as an *adapter*. We've seen Cloud 1.0 and maybe Cloud 1.5 but there's a lot more remaining for us to reach Cloud 2.0.

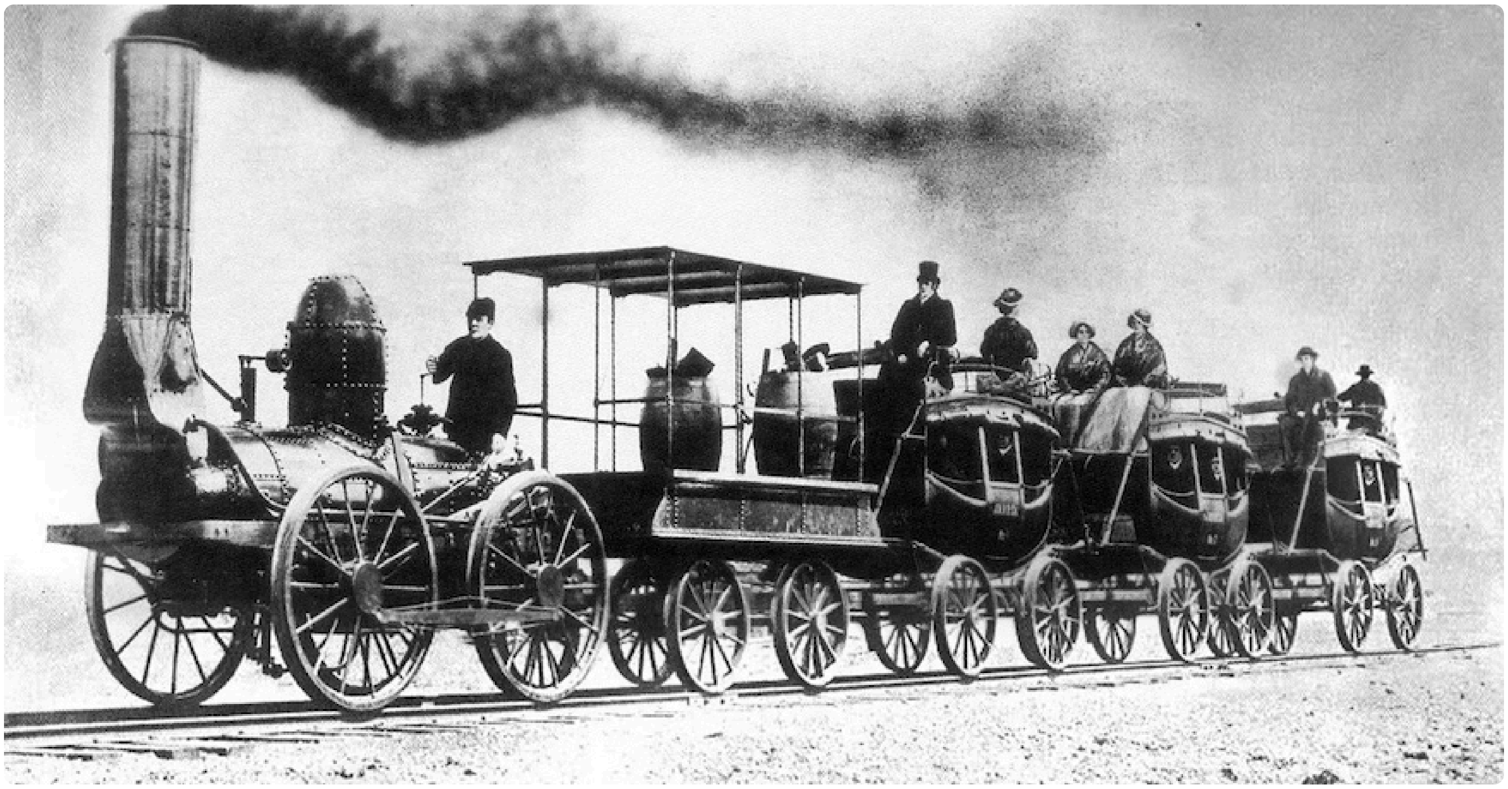
Adapters and toys

I spent seven years at Spotify so let me tell you a story. It might seem tangential but hear me out:

Music consumption in 2008 was going in two directions. iTunes had successfully shifted the consumption of music from physical to the cloud, but the consumption model of *owning* content remained the same. At the same time, Napster et. al. showed the promise of the cloud but was unable to license content and do it the legal way.

Increasing broadband penetration initially enabled music downloading, but quickly also made the location of the music bits irrelevant.

Spotify then (imo!) pulled off a massive Pareto improvement — they offered a service that were *both* better than iTunes and piracy for consumers, and somehow also had a business model that was sufficiently producer friendly. I remember the magic moment from the first time I tried Spotify — it was like having 30M tracks *on my own computer.*



A steam locomotive pulling horse carriages: [4]

I find this useful to illustrate how music consumption went through a couple of shifts in rapid succession:

The first shift is trying to put the old stuff into the new thing. This is the “lift and shift” or “adapter” type approach. These things take advantage of the new tech to some extent, but only partially, and are stuck with legacy technology. The second shift involves products that are native to the new technology, but look like “toys”. Maybe there's no obvious business model, or the tool solves just one particular use case. But it gives some conceptual insight into what's coming.

The exciting thing happens when the toys evolve into something that is obviously better than the legacy technology, and also better than the adapters. Online streaming services basically represents a third and final shift: take the toys but turn it mainstream.

Cloud is coming for your workflow

Somewhat ironically, software development is one of a vanishingly small subset of knowledge jobs for which the main work tool hasn't moved to the cloud. We still write code locally, thus we're constrained to things that work in the same way both locally and in the cloud. Thus, adapter tools like Docker.

I get this. I *love* writing code just on my laptop because of the fast feedback loops it creates. My laptop is full of weird-ass scripts and half-finished projects, and I haven't encountered any other tool that lets me iterate on these things as quickly, maybe with the exception of `ssh` into a devbox, which is something we've done for about 50 years now: [5]

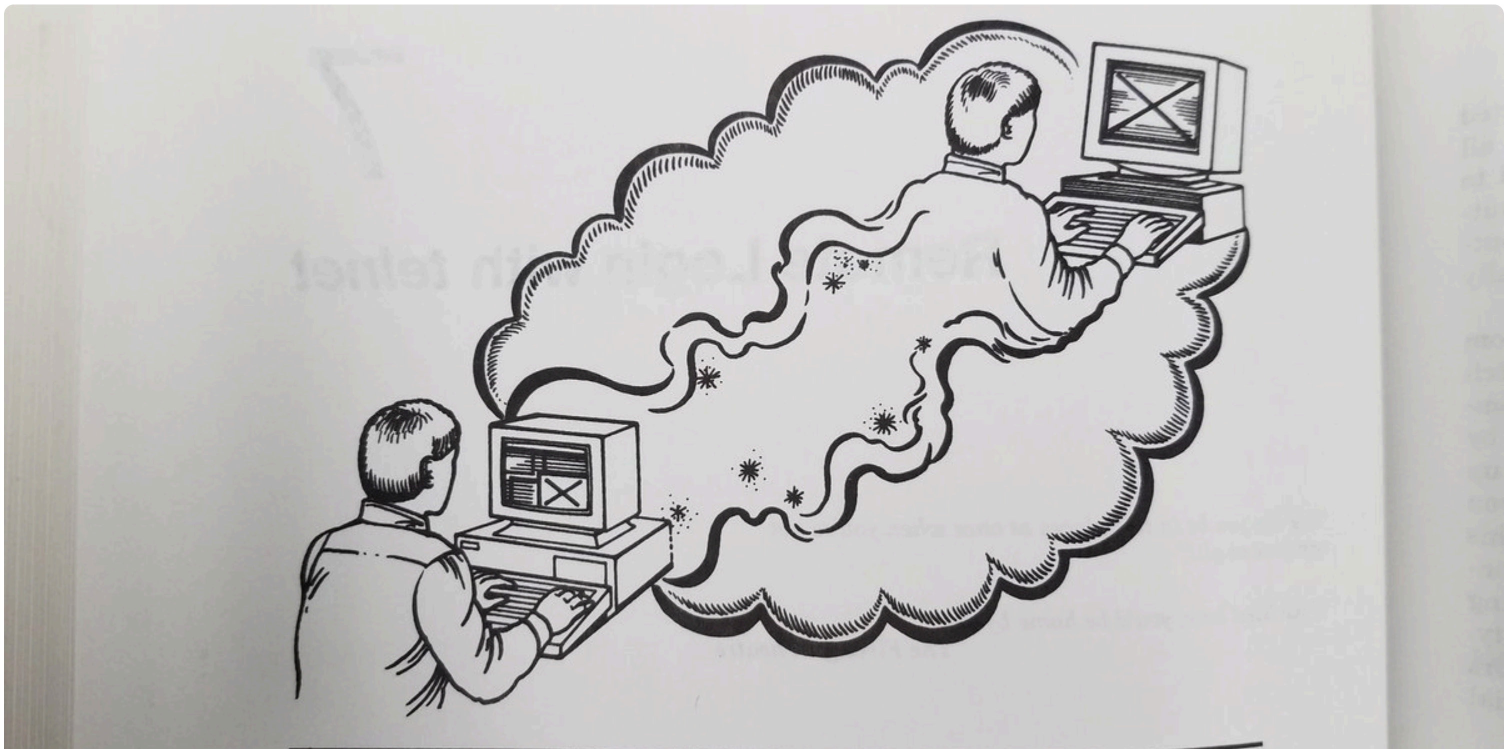


Figure 7.1 Remote login is a lot like astral projection.

We could of course put the whole development environment in a VM in the cloud, but I'm not sure that's much more than the “lift and shift” I already talked about — in the sense that we're not taking advantage of all the cloud stuff — scaling up/down (including to zero), flexible resource needs, pay-as-you-go, etc. I think we're overdue for some larger change in how we develop, deploy, and run code.

That doesn't mean someone is confiscating your beloved local development environment! Or I mean, it sort of does, but it won't happen until the new tools are *better* — which frankly is not the case today, outside of a few use cases. Like I said, the fast feedback loop of local coding is hard to beat *today*, but there's a lot of good reasons why an environment native to the cloud would let you iterate faster eventually. And there are other benefits too, like reducing/removing big environment differences which represent a massive tax on productivity.

To be free, we need to break free of our past

Our stacks are based on 50 years of technology. On top of that, we also have a lot of legacy portability needs: where we needed the same software to run in the cloud, locally, and on-prem. These complex interdependencies will take a while to break free of.

Rethinking these abstractions to be native to the new world let's us start over and redefine the abstractions for what we need? Do we need IP addresses, CIDR blocks, and NATs, or can we focus on which services have access to what resources? Do we need to reserve resources advance (like having a cluster of n workers, each having x GB of RAM), or can we let the runtime expand/shrink in a split second? You can probably guess which option I would put my money on.

These low-level primitives will still be there of course, under the hood, and some people will still think about hardware interrupts and dangling pointers. But just like most developers don't think about these things, the abstraction layers will keep moving up. I'm excited for a world where a normal software developer doesn't need to know about CIDR blocks to connect a Lambda with an RDS instance.

Vertical toys

Building and running software was convoluted enough pre-cloud, and has gotten a lot more complex over the years. I think it's quite likely that the easiest way to start over is to focus on particular use cases and optimize end-to-end for the experience. “Repackage the cloud” for different end users, be it frontend developers, data scientists, and so one. I mean, I'm extremely biased here, but I'm essentialy doing this for data teams, so I'm obviously a believer in this.

You don't have to look far to see some pockets of this. Analytics engineers who run all their SQL in a runtime in the cloud (the data warehouse). Data scientists who develop all their code in notebooks in the cloud. Some (very small subset of) backend developers who run all their code in Lambda containers, even during development. The broad userbase of people building and hosting apps on tools like Replit.

You might look at them and think of them as examples of “toys”, or niche use cases, but that's sort of the point that I'm making — it might look insignificant to the larger set of software engineers. It might not look like the software engineering *you* are doing, and it probably won't quite look like what software development will look like in the future. But it's very possible these people are in fact ahead of the curve.

Welcome, Cloud 2.0

This was just a long way of saying: the clouds are here, and a lot has already happened, but the bulk of what it means for building software is yet to come.

I also want to point out that I wrote a whole blog post about how we're early with the cloud, and I didn't even mention the massive amount of companies that aren't even in the cloud. Besides the massive changes to the cloud itself, there's so much growth just there. And there's something like an order of magnitude *more software engineers* today than 20 years ago, which will further drive demand, etc etc etc. It's going to be a fun decade for tech!

Addendum

This was on the front page of Hacker News and got a bunch of comments.

Separately, if this post resonates with you, and you're working on data and want better infrastructure, you might be interested in Modal, which is what I'm working on right now. We take your code and run it in the cloud for you, while letting you retain the productivity of writing code locally. You can scale things out, schedule things, run things on GPUs, and a bunch of other things.

---

1. This chart is pretty crude — I just looked at the earliest version number for each service in botocore. But it's a bit imprecise, e.g. EC2 was launched in 2006, but the oldest API version is from 2014. ↩

2. And despite some companies with different margin structures ↩

3. I would love to attribute this to the right person! This might to be the earliest source. ↩

4. Credit to Paul Graham for tweeting this. ↩

5. Credit to Cynde Moya for tweeting this. ↩

Tagged with: software, cloud

Want to get blog posts over email?

Enter your email address and get an email (roughly monthly) when there's a new post!

Your email address Subscribe!

Related posts

Storm in the stratosphere: how the cloud will be reshuffled 2021-11-30
Simple sabotage for software 2023-12-13
What I have been working on: Modal 2022-12-07
σ-driven project management: when is the optimal time to give up? 2022-04-05
Software infrastructure 2.0: a wishlist 2021-04-19

Erik Bernhardsson

... is the founder of Modal Labs which is working on some ideas in the data/infrastructure space. I used to be the CTO at Better. A long time ago, I built the music recommendation system at Spotify. You can follow me on Twitter or see some more facts about me.

© Erik Bernhardsson 2022