computers are bad

You are receiving this facsimile because you signed up for fax delivery of this newsletter. To stop delivery, contact Computers Are Bad by email or fax.

https://computer.rip - me@computer.rip - fax: +1 (505) 926-5492

2023-03-24 docker

Lately I tend to stick to topics that are historic by at least twenty years, and that does have a lot of advantages. But I am supposedly a DevOps professional, and so I will occasionally indulge in giving DevOps advice... or at least opinions, which are sort of like advice but with less of a warranty.

There's been a lot of discussion lately about Docker, mostly about their boneheaded reversal following their boneheaded apology for their boneheaded decision to eliminate free teams. I don't really care much about this event in terms of how it impacts my professional work. I long ago wrote off Docker, Inc. as a positive part of the DevOps ecosystem. But what's very interesting to me is *how we got here*: The story of Docker, Docker Inc., Docker Hub, and their relation to the broader world of containerization is endlessly fascinating to me.

How is it that Docker Inc., creator of one of the most important and ubiquitous tools in the modern software industry, has become such a backwater of rent-seeking and foot-shooting? Silicon Valley continually produces some astounding failures, but Docker stands out to me. Docker *as a software product* is an incredible success; Docker *as a company* is a joke; and the work of computing professionals is complicated by the oddly distant and yet oddly close connection between the two.

Docker, from a technical perspective, is more evolutionary than revolutionary. It mostly glued together existing Linux kernel features, following a road that had at least been graded, if not paved and striped, by projects like LXC. Docker as a concept, though, had a revolutionary impact on the DevOps field. Docker quickly became one of the most common ways of distributing server-side software, and whole development workflows rearranged themselves around it. Orchestration tools like the ones we use today are hard to picture without Docker, and for many professionals Docker is on par with their text editor as a primary tool of the trade.

But underlying all of this there has always been sort of a question: what *is* Docker, exactly? I don't necessarily mean the software, but the concept. I have always felt that the software is not really all that great. Many aspects of Docker's user interface and API seem idiosyncratic; some of the abstraction it introduces is more confusing than useful. In particular, the union file system (UFS) image format is a choice that seems more academically aspirational than practical. Sure, it has tidy properties in theory, but my experience has been that developers spend a lot more time working around it than working with it.

All this is to say that I don't think that Docker, the tool, is really all that important. In a different world, LXC might have gained all this market share. Had Docker not come about, something like containerd would likely have emerged anyway. Or

perhaps we would all be using lightweight VMs instead; academic and commercial research tends to show that the advantages containers have over more conventional paravirtualization are far smaller than most believe.

I would argue that the Docker that matters is not software, but a *concept*. A workflow, you might say, although I don't think it's even that concrete. The Docker that swept DevOps like a savior come to spare us from Enterprise JavaBeans isn't really about the runtime at all. It's about the images, and more about the ease of programatically creating images. Much of this benefit comes from composition: perhaps the most important single feature of Docker is the FROM keyword.

So Docker is an open-source software product, one that is basically free (as in beer and as in freedom) although hindered by a history of messy licensing situations. Docker is also a company, and companies are expected to produce revenue. And that's where other facets of the greater identity we call "Docker" come to light: Docker Desktop and Docker Hub.

Docker Desktop isn't really that interesting to me. Docker is closely coupled to Linux in a way that makes it difficult to run on the predominant platform used by developers [1]. Docker Inc. developed Docker Desktop, a tool that runs Docker in a VM using fewer clicks than it would take to set that up yourself (which is still not that many clicks). Docker Inc. then needed to make money, so they slapped a licensing fee on Docker Desktop. I responded by switching to Podman, but I get that some people are willing to pay the monthly fee for the simplicity of Docker Desktop, even if I feel that the particular implementation of Docker Desktop often makes things harder rather than easier.

Also I find the Docker Desktop "GUI" to be incredibly, intensely annoying, especially since Docker Inc. seems to pressure you to use it in a desperate attempt to dig what Silicon Valley types call a moat. But I fully acknowledge that I am a weird computer curmudgeon who uses Thunderbird and pines for the better performance of, well, pine.

Still, the point of this tangent about Docker Desktop is that Docker's decision to monetize via Desktop---and in a pretty irritating way that caused a great deal of heartburn to many software companies---was probably the first tangible sign that Docker Inc. is not the benevolent force that it had long seemed to be. Suddenly Docker, the open-source tool that made our work so much easier, had an ugly clash with capitalism. Docker became a FOSS engine behind a commercial tool that Docker Inc. badly wanted us to pay for.

Docker Desktop also illustrates a recurring problem with Docker: the borders between free and paid within the scope of their commercial products. Docker Desktop became free for certain use-cases including personal use and use in small businesses, but requires a paid subscription for use in larger companies. This kind of arrangement might seem like a charitable compromise but is also sort of a worst-of-both-worlds: Docker Desktop is free enough to be ubiquitous but commercial enough to pose an alarming liability to large companies. Some companies exceeding Docker's definition of a small company have gone as far as using their device management tools to forcibly remove Docker Desktop, in order to mitigate the risk of a lawsuit for violating its license.

There is a fundamental problem with "free for some, paid for others": it requires that users determine whether or not they are permitted to use the tool for free. Even well-intentioned users will screw this up when the rules require knowledge of their employer's financials and, moreover, are in small print at the very bottom of a pricing page that says "free" at the top. Personally, I think that Docker Inc.'s pricing page borders on outright deception by making the licensing restrictions on Docker Desktop so

unobvious.

Docker Hub, though: Docker Hub is really something.

That most compelling feature of Docker, the ability to easily pull images from somewhere else and even build on top of them, depends on there being a place to pull images from. It's easy to see how, at first, Docker Inc. figured that the most important thing was to have a ubiquitous, open Docker registry that made it easy for people to get started. In this way, we might view Docker Hub as having been a sort of scaffolding for the Docker movement. The fact that you could just run 'docker pull ubuntu' and have it work was probably actually quite important to the early adoption of Docker, and many continue to depend on it today.

Docker Hub, though, may yet be Docker's undoing. I can only assume that Docker did not realize the situation they were getting into. Docker images are relatively large, and Docker Hub became so central to the use of Docker that it became common for DevOps toolchains to pull images to production nodes straight from Docker Hub. Bandwidth is relatively expensive even before cloud provider margins; the cost of operating Docker Hub must have become huge. Docker Inc.'s scaffolding for the Docker community suddenly became core infrastructure for endless cloud environments, and effectively a subsidy to Docker's many users.

It's hard to blame Docker Inc. too much for flailing. Docker Hub's operating costs were probably unsustainable, and there aren't a lot of options to fix this other than making Docker Hub expensive, or making Docker Hub worse, or both. Docker Inc. seems to have opted for both. Docker Hub is not especially fast, in fact it's pretty slow compared to almost any other option. Docker Hub now imposes per-IP quotas, which probably would have been totally reasonable at the start but was a total disaster when it was introduced post-hoc and suddenly caused thousands, if not millions, of DevOps pipelines to intermittently fail.

Docker Inc.'s goal was presumably that users would start using paid Docker plans to raise the quotas but, well, that's only attractive for users that either don't know about caching proxies or judge the overhead of using one to be more costly than Docker Hub... and I have a hard time picturing an organization where that would be true.

That's the strange thing about Docker Hub. It is both totally replaceable and totally unreplaceable.

Docker Hub is totally replaceable in that the Docker registry API is really pretty simple and easy to implement in other products. There are tons of options for Docker registries other than Docker Hub, and frankly most of them are much better options. I'm not just saying that because GitLab [2] has a built-in Docker registry, but that sort of illustrates the point. *Of course* GitLab has a built-in Docker registry, it's no big deal. It's not even that GitLab introduced it as a competitor to Docker Hub, that's sort of absurd, Docker Hub doesn't even really figure. GitLab introduced it as a competitor to Sonatype Nexus and JFrog Artifactory, to say nothing of the docker registries offered by just about every cloud provider. For someone choosing a Docker registry to deploy or subscribe to, Docker Hub has no clear advantage, and probably ranks pretty low among the options.

And yet Docker Hub is *the* Docker registry, and the whole teetering tower of DevOps is deeply dependent on it! What an odd contradiction, and yet it's completely obvious why:

First, Docker Hub is free. Implausibly free, and as it turns out, probably unsustainably

free. There's an old maxim that if you're not paying, you're the product. But Docker Hub reminds us that in the VC-driven (and not particularly results-driven) world of Silicon Valley there is a potent second possibility: if you're not paying, there may be no product at all. At least not once your vendor gets to the end of the runway [3].

Second, Docker Hub is the default. Being the default can be a big deal, and this is *painfully* true for Docker. The dominance of short, convenient "user/image" or even just "image" references is so strong that Docker image references that actually specify a registry feel almost feels like an off-label hack, a workaround for how Docker is really supposed to be used. What's more, Docker Hub's original quotas (or rather lack thereof) left no need for authentication in many situations, so having to authenticate to a registry also feels like an extra hassle. Many tools built around Docker don't make the use of a non-Docker Hub registry, or any authentication to a registry, as convenient as it probably should be. Tutorials and guides for Docker often omit setup of any registry other than Docker Hub, since Docker Hub is already configured and has everything available in it. You only find out the mistake you've made when your pipelines stop working until the quota period resets, or worse, pulls in production start failing and you have to hope you're lucky enough to check the Kubernetes events before digging around a dozen other places.

So the solution to the Docker Hub problem is obvious: stop using Docker Hub. It was probably a bad idea all along. But the reality of the situation is much harder. Moving off of Docker Hub is a *pain*, and one that has a way of staying pretty far down priority lists. Docker Hub references, or rather references with no registry at all that default to Docker Hub, are so ubiquitous that any project moving their official builds off of Docker Hub will probably break a tremendous number of downstream users.

Docker Inc.'s behavior with Docker Desktop and especially Docker Hub feels like rent-seeking at best, and potentially extortionate. It's not exactly fair to blame all of this on Docker Inc.; both commercial users and the open-source community should have foreseen the retrospectively obvious risk of Docker actually thinking about the economics. Nonetheless, a cynical and not entirely unreasonable take on this story is that Docker hoodwinked us. Perhaps Docker has simply stumbled upon the "Embrace, Extend, Extinguish" of our age: employ FOSS software defaults and lazy developer practices (that were inculcated by Docker's documentation) to make everyone dependent on Docker Inc.'s free registry, then tighten the quota screws until they have no choice than to pay in. This is a very cynical take indeed! I don't really believe it, mostly because it involves far more strategic vision than I would credit Docker Inc. with.

I decided to write about this because I think there are lessons to be learned. Important lessons. No doubt some of this problem is directly attributable to the economic conditions that dominated Silicon Valley for the last decade. Docker Inc. probably wouldn't have gotten so far, burning so much money, had there not been an incredible amount of money to burn. Still, it seems inevitable that this kind of relationship between open-source software and corporate strategy, and between free and paid services, will happen again.

I propose these takeaways, as discussion topics if nothing else:

1. Be skeptical of free services, especially ones that are required for any part of your business (or open source venture, or hobby project, etc). Free services should never become a deeply embedded dependency unless there is very good reason to believe they will remain free. Perhaps the backing of a large foundation or corporate sponsor with a good history with open source would count, but even that is

no promise. Consider the example of Red Hat, its acquisition by IBM, and the impact of that business event on projects previously considered extremely reliable like CentOS.

2. Free tools that rely on third-party services are only free for the time being. Sure, this might be obvious, but it's probably a deeper problem than you realize. Docker never *relied* on Docker Hub in that it has always been possible to use other registries. But Docker and the community *strongly encouraged* the use of Docker Hub through technical, economic, and social means. This had the result of making Docker Hub a de facto hard requirement for many projects and systems.

3. When writing documentation, guides, blog posts, advice to coworkers, etc., think about long-term sustainability even when it is less convenient. I suspect that the ongoing slow-motion meltdown over Docker Hub would have been greatly mitigated if the use of multiple Docker registries, or at least the easy ability to specify a third-party registry and authenticate, were considered best practices and more common in the Docker community.

[1] I mean MacOS, but you can assume I mean Windows and it still works.

[2] My employer whose opinions these are not.

[3] I am here resisting the urge to write a convoluted aviation metaphor. Something about being passengers on a whale-shaped plane that is hitting the last thousand feet and still short of V_r, so the captain says we only get 100 builds per 6 hours per IP and the rest are going out the window.

p.s. I took so long to write this so late at night that now the date in the title is wrong, haha whoops not fixing it