I use some of my work tools outside of work. Without really needing them. There is no better way to get to know a piece of software - especially as big as an operating system - than by using it whenever one can, and for various tasks and purposes. Fiddle, hack, go beyond whats necessary, achieve things that are not strictly required. Even if just for fun. And when one learns an OS well, other OSs may start feeling less welcoming, sometimes even more hostile. Commercial desktop operating systems require increasingly more attention by causing various distractions. I use FreeBSD with Linux in virtual machines as my daily driver. Do I have all the nice stuff Not all, but necessary things - yes. Do I have anything I dont want Nope - no ads, unwelcome interface changes shipped with unrelated bug patches, no invasive phoning home, unsolicited files scanning, ongoing configurability deprivation nor other unpleasantries. I dont need to apologize to my OS for not wanting something, pretend that I will do something later, avoid a question, forced to suggest that Im willing to talk about it the next day, over and over. No means no. Theres this mutual understanding and lack of need for little, temporary non-aggression pacts. There is no war going on, nor theres any feeling of defeat. My OS doesnt try to change my behavior for someone elses benefit. Overall, its somewhat crude but efficient and doesnt cause distractions. I dont need my driver to be fancy. Im not Miss Daisy. Intrusion detectionprevention systems are the kind of software I would use even if no clients of mine would be using, to protect both publicly-exposed services and software, and my workstations. On Linux I usually utilize Fail2ban with iptables or external firewall, on FreeBSD its the built-in blacklistd which integrates nicely with the pf firewall. Often just one of public systems is hit with an unsolicited vulnerability scan, brute force or other kind of attack detectable by Fail2ban. Fail2ban sends info on particular attack attempts to a hub. Hub then generates firewall-ready rules and, separately, lists of addresses and networks being large sources of abuse, for different kind of services. Separate lists are created with ongoing web attacks, mail attacks, SSH attacks and so on. Other systems apply those firewall rules or use particular address lists to filter such unwanted traffic via access lists or on their respective proxies or CDNs. In short, on top of an IDS Intrusion Detection System, I built a blocklist to prevent certain attacks before they hit more servers and services. Two more pieces of software which I would be using even if my customers wouldnt Terraform for cloud deployments and Ansible for systems provisioning. Personal projects need disaster recovery, too. In fact, I started using both Ansible and Terraform before having paid duties involving any of them. Later, I have introduced IaC Infrastructure as Code to several companies and organizations. Prior to that I was provisioning systems with shell scripts, and cloud resources with provided web panels. Thats no way to go with larger infrastructures or in case of a disaster. One position I held, required that all systems were to be manually checklisted by two engineers before going to production. Needless to say - this company was provisioning systems rather rarely and such requirement would not fly with modern cloud setups. I keep my Terraform cloud deployment definitions and Ansible provisioning roles and playbooks up-to-date to always be able to quickly spin up a fleet of VMs when needed for some tests. Not all customers have proper labtesting environments, sometimes obtaining such access takes time, and that unnecessarily disrupts the flow. Apart from managing servers, I also find Ansible quite ok for provisioning software on my laptops. With this part automated, potentially problematic upgrades like disk firmware and OS reinstalls are less painful and recovery process requires little to no user attention. A very valuable lesson that I learned regarding Terraform - and this was when I already held HashiCorps certification - is that Terraform is only as good as used providers Terraform name for modulesplugins responsible for talking to particular APIs or at the very least - providers ability to handle API errors. Im looking at you, cloud vendor with liquid in its name. For over 20 years now, Im handling my own mail. Call me a mad man but thats my preferred way for multiple reasons, list of which doesnt fit this post. Postfix was my go-to MTA Mail Transport Agent, server to server mail software since day one. Even when I worked with Exim on customers systems. It may just be the most reliable piece of software I have ever used and you simply dont replace things that are this good. As for IMAP, its been Dovecot for years. Its even handling my 465587 traffic submissions - client sending mail as a proxy to Postfix. To me, its the most ridiculous thing on this list. I use Kubernetes only to keep up-to-date with its updates and to test stuff. Its my training ground, lab and I build and test stuff on it before proposing or shipping to clients. And its always on. I host the bare Kubernetes version on a dedicated server instead of using cloud-provided managed SaaS option for a few reasons to be able to test upgrades of Kubernetes itself or existing workflows on new k8s versions, prepare for deprecations, to not be time-pressured by a cloud operator adding every hour of use up to my billing. Working with metered option tends to negatively impact my flow and the dedicated box in question is in use and had spare resources. I host some projects for myself, both self-built and open source self-hosted alternatives to other services. Just one of these projects is on Kubernetes. And only for monitoring purposes liveness and readiness probes, 3rd-party monitoring, etc. Total overkill for such a tiny project built for own use. But it serves me well for development of DevOps-y stuff as a target for monitoring and automation of disaster recovery actions, scalability and high-availability tests, and so I can be a user sometimes, and dont lose developers perspective. I mis-use KubernetesYep - I said it. I mis-use k8s all its nodes reside as virtual machines on one physical server.