I used to work as an Infrastructure Architect for the e-commerce division of clothing retailer based in San Francisco. We had 4 different datacenters in the US (2 of which we owned that we used for development, and end to end testing, respectively. Another 2 that we rented for production and disaster recovery, respectively). In addition, we were multi public cloud with both Oracle (OCI) and Microsoft (Azure). We attempted to keep every aspect of these environments identical. If a certain database in production was oracle on baremetal, well, it should be oracle baremetal in the 3 other environments as well.

Our company faced a tremendous number of challenges, but the ones we had the most difficulty dealing with were largely cultural and organizational. For the better part of a 15 years, the company's portfolio of work (and project funding) was "feature" driven. New features for the e-commerce site were determined every year, which almost always resulted in new code running on new servers, with the latest linux OS, and with a new corresponding database. When virtualization came into play, we started virtualizing, then when private cloud came into play, we went that direction, but legacy equipment was often ignored. Repeat this pattern for 15 years and you're dealing with a 15 year spread of technologies. You've got 10Mbps switches, 10Gbit switches, SCSI drives, SSD drives, AIX, Linux (based on 2.2 kernel), Linux (based on 4.0 kernel), physical servers, type 2 virtualization, type 1 virtualization, VIO(virtualized AIX), VMware, KVM, openstack, public cloud, cfengine, puppet, chef, yum repos for every linux distro....you get the picture. Our infrastructure team went from 5 in 2005, to 75+ by 2016 (it declined afterwards, but not due to quantity of work...other reasons). At our peak we had over 80,000 OS's/IP's in the wild (sadly a significant chunk could not be accounted for, but everyone feared turning them off).

This is a long winded way of saying I've seen technology sprawl firsthand and understand the challenges of maintaining lower levels (hardware/network/OS) of the application stack (and I should note that we had so much work—OS patching, at least on linux, was simply not performed)

The obvious answer to improving our infrastructure footprint would be to say that we should make the move to a cloud based PaaS solution (We have so much custom code that relies heavily on proprietary databases and message queuing technologies, that SaaS is just to big a stretch for us at this time). The benefit of going PaaS would essentially eliminate management of some of our most troubling layers: physical datacenters, network equipment, servers, and disk arrays). We could likely remove cfengine, chef, and puppet, and delete our yum repos entirely.For our organization however, implementing PaaS would only provide benefit if we change our behavior. We don't just want to add PaaS to our long

growing list of technologies. We would need to commit to PaaS in the long term, and go thru the painful process of moving 15year old application stacks to modern PaaS runtimes. This won't be easy, but is still less complex than to trying to move the same application to IaaS and then maintaining that portion stack as well.

PaaS would not be a silver bullet solution however, and try as we might to standardize on a single platform, there will always be outliers. We do have the option for our customers to store their credit cards with us. As such, given the security requirements there, those databases would either likely stay on-prem or on a segregated (and encrypted) disk instance of a vertically scalable IaaS VM(s). Additionally there are other 3rd party proprietary software that is written in C and Cobol that is only supported on very specific hardware installations and OS's that would likely stay IaaS until the vendor approves a PaaS platform.