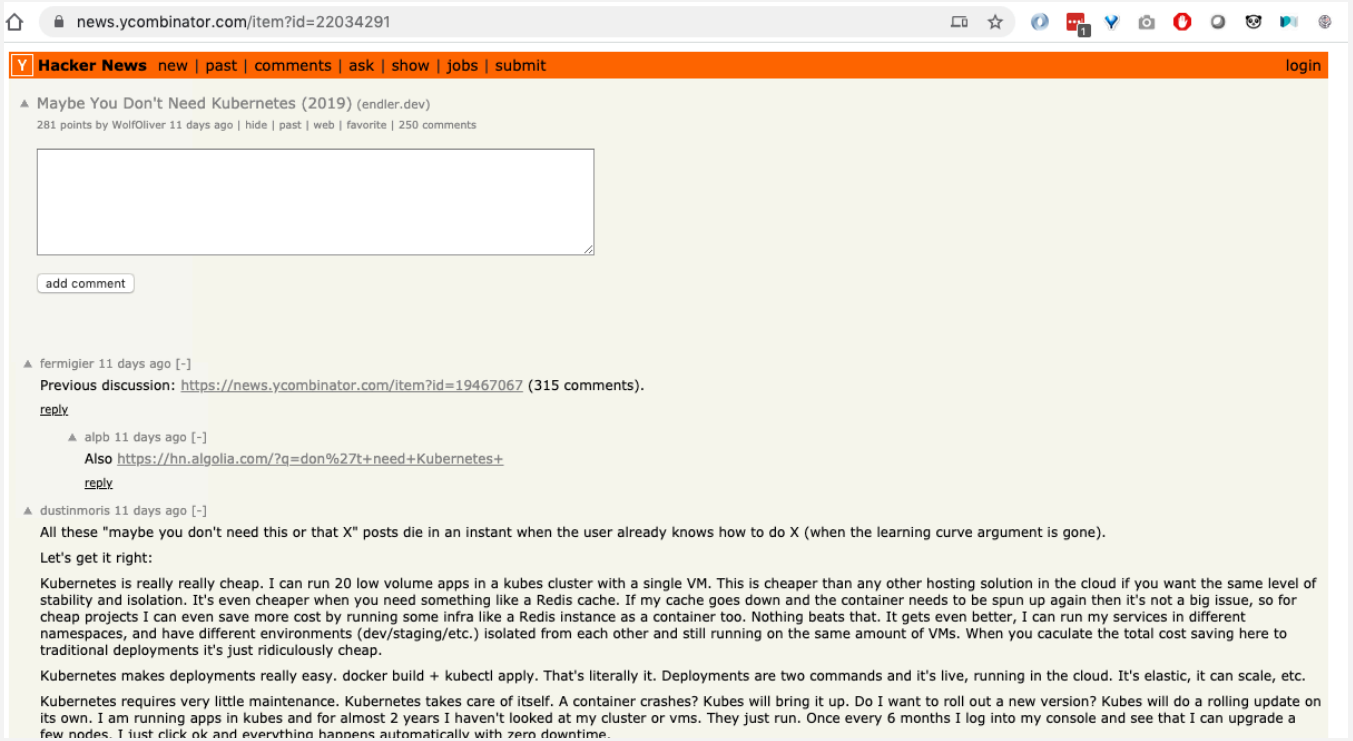
Hacker News “Maybe You Don’t Need Kubernetes” Synthesis

—

January 24, 2020

Methodology

250 comments read



news.ycombinator.com/item?id=22034291

Y **Hacker News** new | past | comments | ask | show | jobs | submit login

▲ Maybe You Don't Need Kubernetes (2019) (endler.dev)
281 points by WolfOliver 11 days ago | hide | past | web | favorite | 250 comments

add comment

▲ fermigier 11 days ago [-]
Previous discussion: https://news.ycombinator.com/item?id=19467067 (315 comments).
reply

▲ alpb 11 days ago [-]
Also https://hn.algolia.com/?q=don%27t+need+Kubernetes+
reply

▲ dustinmoris 11 days ago [-]
All these "maybe you don't need this or that X" posts die in an instant when the user already knows how to do X (when the learning curve argument is gone).

Let's get it right:

Kubernetes is really really cheap. I can run 20 low volume apps in a kubes cluster with a single VM. This is cheaper than any other hosting solution in the cloud if you want the same level of stability and isolation. It's even cheaper when you need something like a Redis cache. If my cache goes down and the container needs to be spun up again then it's not a big issue, so for cheap projects I can even save more cost by running some infra like a Redis instance as a container too. Nothing beats that. It gets even better, I can run my services in different namespaces, and have different environments (dev/staging/etc.) isolated from each other and still running on the same amount of VMs. When you caculate the total cost saving here to traditional deployments it's just ridiculously cheap.

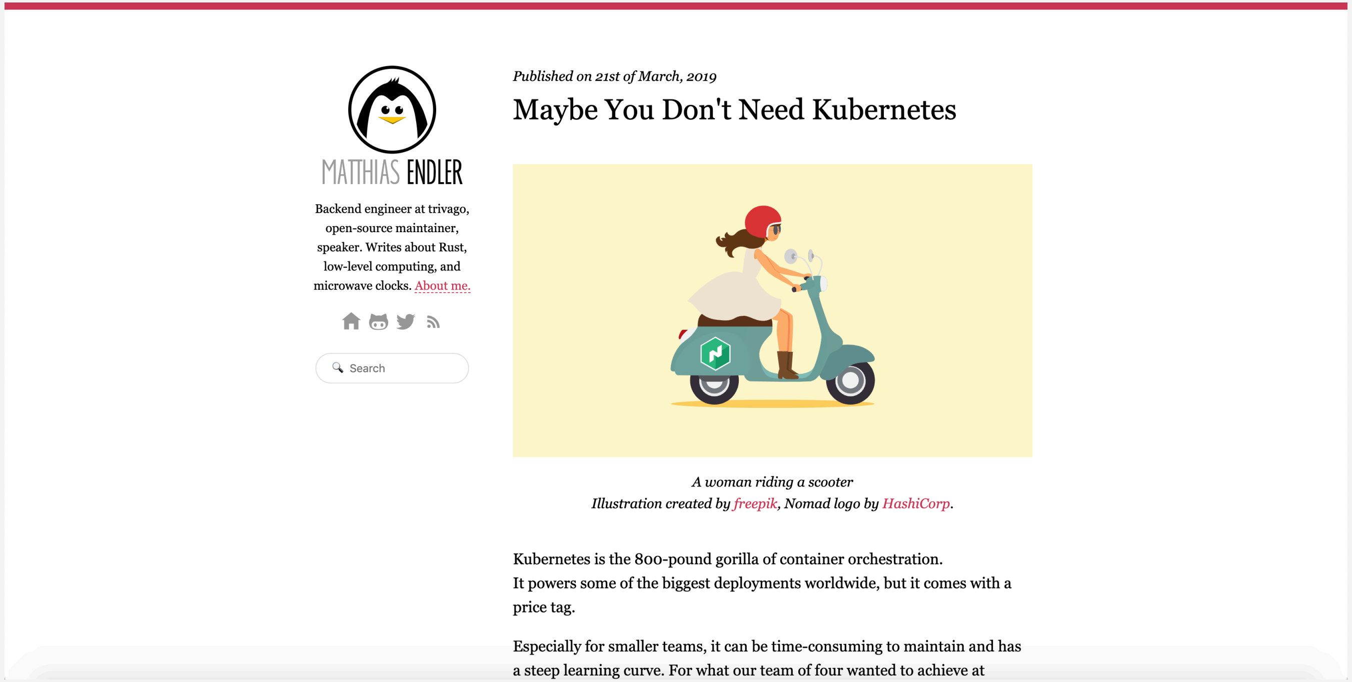
Kubernetes makes deployments really easy. docker build + kubectl apply. That's literally it. Deployments are two commands and it's live, running in the cloud. It's elastic, it can scale, etc.

Kubernetes requires very little maintenance. Kubernetes takes care of itself. A container crashes? Kubes will bring it up. Do I want to roll out a new version? Kubes will do a rolling update on its own. I am running apps in kubes and for almost 2 years I haven't looked at my cluster or vms. They just run. Once every 6 months I log into my console and see that I can upgrade a few nodes. I just click ok and everything happens automatically with zero downtime.

Source: news.ycombinator.com

Methodology

250 comments read
original article (republished)



MATTHIAS ENDLER

Backend engineer at trivago, open-source maintainer, speaker. Writes about Rust, low-level computing, and microwave clocks. About me.

Search

Published on 21st of March, 2019

Maybe You Don't Need Kubernetes

A woman riding a scooter
Illustration created by freepik, Nomad logo by HashiCorp.

Kubernetes is the 800-pound gorilla of container orchestration.
It powers some of the biggest deployments worldwide, but it comes with a price tag.

Especially for smaller teams, it can be time-consuming to maintain and has a steep learning curve. For what our team of four wanted to achieve at

Source: news.ycombinator.com

“Kubernetes makes easy things harder than they should be, and hard things easier than they should be.”

curryst

When Kubernetes is awesome

“Kubernetes, once you know it, is significantly easier than cobbling together an environment from "classical" solutions that combine Puppet/Chef/Ansible, homegrown shell scripts, static VMs, and SSH.

Sure, you can bring up a single VM with those technologies and be up and running quickly. But a real production environment will need automatic scaling (both of processes and nodes), CPU/memory limits, rolling app/infra upgrades, distributed log collection and monitoring, resilience to node failure, load balancing, stateful services (e.g. a database; anything that stores its state on disk and can't use a distributed file system), etc., and you end up building a very, very poor man's Kubernetes dealing with all of the above.

With Kubernetes, all of the work has been done, and you only need to deal with high-level primitives. "Nodes" become an abstraction. You just specify what should run, and the cluster takes care of it.

I've been there, many times. I ran stuff the "classical" Unix way -- successfully, but painfully -- for about 15 years and I'm not going back there.

There are alternatives, of course. Terraform and CloudFormation and things like that. There's Nomad. You can even cobble together something with Docker. **But those solutions all require a lot more custom glue from the ops team than Kubernetes.**”

atombender

A standard for running production systems

“I agree with you that [Kubernetes alternative] is a simpler deployment method when you don't need HA. As soon as you need HA, then all of a sudden you need to be able to fail over to new nodes and manage a load balancer and the backends for that load balancer. **Kubernetes makes that easy. Kubernetes makes easy things harder than they should be, and hard things easier than they should be.**”

curryst

“I really just don't think they understand [Kubernetes], because **any production environment should have many of the features Kubernetes helps provide**. So the argument becomes "I know how to do it this other way, so learning a new tool is too complex."”

Kubernetes helps standardize a lot of these things - I can very easily hop between different clusters running completely different apps/topologies and have a good sense of what's going on. A mish-mash of custom solutions re- inventing these wheels is, in my opinion, far more confusing.

ar_lan

But many don’t want to pay

the “Kubernetes tax”[1]

1. candiddevmike

When Kubernetes is not awesome

“...A year or so ago I thought, hey, maybe I should redo my personal infrastructure using Kubernetes. Long story short, it was way too much of a pain in the ass.

As background, I've done time as a professional sysadmin. My current infrastructure is all Chef-based, with maybe a dozen custom cookbooks. But Chef felt kinda heavy and clunky, and the many VMs I had definitely seemed heavy compared with containerization. I thought switching to Kubernetes would be pretty straightforward.

Surprise! It was not. I moved the least complex thing I run, my home lighting daemon to it; it's stateless and nothing connects to it, but it was still a struggle to get it up and running. Then I tried adding more stateful services and got bogged down in bugs, mysteries, and Kubernetes complexity. I set it aside, thinking I'd come back to it later when I had more time. That time never quite arrived, and a month or so ago my home lights stopped working. Why? I couldn't tell. A bunch of internal Kubernetes certificates expired, so none of the commands worked. Eventually, I just copy-pasted stuff out of Stack Overflow and randomly rebooted things, and eventually it started working again.

I'll happily look at it again when I have to do serious volume and can afford somebody to focus full-time on Kubernetes. But for anything small or casual, I'll be looking elsewhere.”

wpietri

Building a service platform

“At work we're building an entire service platform on top of managed kubernetes services, agnostic to cloud provider. We had already had bad experiences running K8s ourselves.

Going into it we knew how much of a PITA it would be but we vastly underestimated how much, IMO.

Would not do again -- I would quit first.”

`busterarm`

Discerning Kubernetes abstractions

“When creating a prototype with Kubernetes, we noticed that we started adding ever-more complex layers of logic to operate our services. Logic on which we implicitly relied on.

As an example, Kubernetes allows embedding service configurations using ConfigMaps. Especially when merging multiple config files or adding more services to a pod, this can get quite confusing quickly. Kubernetes - or helm, for that matter - allows injecting external configs dynamically to ensure separation of concerns. But this can lead to tight, implicit coupling between your project and Kubernetes. Helm and ConfigMaps are optional features so you don’t have to use them. You might as well just copy the config into the Docker image. However, it’s tempting to go down that path and build unnecessary abstractions that can later bite you.”

endler.dev

“Not because it's bad or especially hard, but because there's so much to unpack, and it's so tempting to unpack it all at once, and there's so much foundational stuff ... which you really ought to learn before you try to analyze in detail exactly how the system is built up.

I learned Kubernetes around v1.5 just before RBAC was enabled by default, and I resisted upgrading past 1.6 for a good long while (until about v1.12) because it was a feature I didn't need, and all the features after it appeared to be something else which I didn't need.”

yeben

Debugging now has more overhead

“The biggest issue I have with k8s as a developer is that while it simplifies the devops side of things, it complicates the development/testing cycle by adding an extra layer of complication when things go wrong.”

imron

"How do you troubleshoot an api-group that is failing intermittently?
How do you troubleshoot a CSI issue? Because CSI isn't a simple protocol like CNI.
What do you look at if kubectl gets randomly stuck?
What do you do if a node becomes notReady?
What do you do if the container runtime of a several nodes starts failing?
What do you do if one of the etcd nodes doesn't start?
What if you are misisng logs in the log aggregator or metrics?
What if when you create a project it's missing the service accounts?
What if kube-proxy randomly doesn't work?
What if the upgrade fails and ends up in an inconsistent state?
What if your pod is actually running but shows as pending?

Sure, you can learn how to deploy kubernetes and an application on top of it in a couple days, but learning how to run a production will take way longer than that.”

brozaman

Rapid pace of updates

The notion that one has to keep pace with Kubernetes upgrades is exactly the kind of thing that works fine if you have a full-time professional on the job, and very poorly if it's a sideline for people trying to get actual productive work done.

`wpietri`

Discussion questions

How do we process user “inputs” like this?
Common rubric for good experiences?