-by Jeff Knupp,

There are two recent trends I really hate: DevOps and the notion of the "full-stack" developer. The DevOps movement is so popular that I may as well say I hate the x86 architecture or monolithic kernels. But it's true: I can't stand it. The underlying cause of my pain? This fact: *not every company is a start-up, though it appears that every company must act as though they were.*

DevOps

"DevOps" is meant to denote a close collaboration and cross-pollination between what were previously purely development roles, purely operations roles, and purely QA roles. Because software needs to be released at an ever-increasing rate, the old "waterfall" develop-test-release cycle is seen as broken. Developers must also take responsibility for the quality of the testing and release environments.

The increasing scope of responsibility of the "developer" (whether or not that term is even appropriate anymore is debatable) has given rise to a chimera-like job candidate: the "full-stack" developer. Such a developer is capable of doing the job of developer, QA team member, operations analyst, sysadmin, and DBA. Before you accuse me of hyperbole, go back and read that list again. Is there any role in the list whose duties you *wouldn't* expect a "full-stack" developer to be well versed in?

Where did these concepts come from? Start-ups, of course (and the Agile methodology). Start-ups are a peculiar beast and need to function in a very lean way to survive their first few years. **I don't deny this**. Unfortunately, we've taken the multiple technical roles that engineers at start-ups *were forced to play due to lack of resources* into a set of minimum qualifications for the role of "developer".

Many Hats

Imagine you're at a start-up with a development team of seven. You're one year into development of a web applications that X's all the Y's and things are going well, though it's always a frantic scramble to keep everything going. If there's a particularly nasty issue that seems to require deep database knowledge, you don't have the liberty of saying "that's not my specialty," and handing it off to a DBA team to investigate. **Due to constrained resources,** you're forced to take on the role of DBA and fix the issue yourself.

Now expand that scenario across all the roles listed earlier. At any one time, a developer at a start-up may be acting as a developer, QA tester, deployment/operations analyst, sysadmin, or DBA. That's just the nature of the business, and some people thrive in that type of environment. Somewhere along the way, however, we tricked ourselves into thinking that because, at any one time, a start-up developer *had* to take on different roles he or she *should actually be all those things at once.*

If such people **even existed**, "full-stack" developers *still* wouldn't be used as they should. Rather than temporarily taking on *a single role* for a short period of time, then transitioning into the next role, they are meant to be performing **all the roles, all the time**. And here's what really sucks: most good developers can almost pull this off.

The Totem Pole

Good developers are smart people. I know I'm going to get a ton of hate mail, but there *is* a hierarchy of usefulness of technology roles in an organization. Developer is at the top, followed by sysadmin and DBA. QA teams, "operations" people, release coordinators and the like are at the bottom of the totem pole. Why is it arranged like this?

Because each role can do the job of all roles below it if necessary.

Start-ups taught us this. Good developers can be passable DBAs if need be. They make decent testers, "deployment engineers", and whatever other ridiculous term you'd like to use. *Their job requires them to know much of the domain of "lower" roles.* There's one big problem with this, and hopefully by now you see it:

It doesn't work in the opposite direction.

A QA person can't just do the job of a developer in a pinch, nor can a build-engineer do the job of a DBA. *They never acquired the specialized knowledge required to perform the*

role. And that's fine. Like it or not, there are hierarchies in every organization, and people have different skill sets and levels of ability. However, *when you make developers take on other roles, you don't have anyone to take on the role of development!*

An example will make this more clear. My dad is a dentist running his own practice. He employs a secretary, hygienist, and dental assistant. Under some sort of "DentOps" movement, my dad would be making appointments and cleaning people's teeth while trying to find time to drill cavities, perform root canals, etc. My dad *can* do all of the other jobs in his office, because he has all the specialized knowledge required to do so.

But no one, not even all of his employees combined, can do his job.

Such a movement does a disservice to everyone involved, except (of course) employers. What began as an experiment aimed at increasing software quality has become a farce, where the most talented employees are overworked (while doing less, less useful work) and lower-level positions simply don't exist.

And this is the crux of the issue. All of the positions previously held by people of various levels of ability are made redundant by the "full-stack" engineer. Large companies love this, as it means they can hire far fewer people to do the same amount of work. In the process, though, *actual development becomes a vanishingly small*

part of a developer's job. This is why we see so many developers that can't pass FizzBuzz: they never really had to write any code. All too common a question now, can you imagine interviewing a chef and asking him what portion of the day he actually devotes to cooking?

Jack of All Trades, Master of None

If you are a developer of moderately sized software, you need a deployment system in place. Quick, what are the benefits and drawbacks of the following such systems: Puppet, Chef, Salt, Ansible, Vagrant, Docker. Now implement your deployment solution! Did you even realize which systems had no business being in that list?

We specialize for a reason: human beings are only capable of retaining so much knowledge. Task-switching is cognitively expensive. Forcing developers to take on additional roles traditionally performed by specialists means that they:

- aren't spending their time developing
- need to keep up with an enormous domain of knowledge
- are going to burn out

What's more, by forcing developers to take on "full-stack" responsibilities, they are paying their employees far *more* than the market average for most of those tasks. If a developer makes 100K a year, you can pay four developers 100K per year to do 50% development and 50% release management on a single, two-person task. Or,

simply hire a release manager at, say, 75K
and *two* developers who develop full-time. And notice the time wasted by developers who are part time release-managers but don't always have releases to manage.

Don't Kill the Developer

The effect of all of this is to destroy the role of "developer" and replace it with a sort of "technology utility-player". Every developer I know got into programming because they actually enjoyed doing it (at one point). You do a disservice to everyone involved when you force your brightest people to take on additional roles.

Not every company is a start-up. Start-ups don't make developers wear multiple hats by choice, they do so out of necessity. Your company likely has enough resource constraints without you inventing some. Please, don't confuse "being lean" with "running with the fewest possible employees". And for God's sake, **let developers write code!**