CONTAINERS AREN'T ALWAYS THE SOLUTION

Heralded as the solution to so many programming issues, companies are embracing container technology. Talk to most developers and you'll see why: containers offer a quick, easy-to-implement workaround that allows companies, who often rely on monolith programming environments, to respond to a competitive market that increasingly demands high-quality products on an ever-shorter timeline.

But, a more skeptical view holds maintains that containers are yet another trend where a tool—the container—is expected to fix larger issues like the computing environment itself or the wider company culture. Let's take a look.

Why are containers so popular?

Businesses today are hard-pressed to meet neverending demands from customers to regularly release new and innovative apps and updates. Most enterprises, however, are working within monolith systems that feel a bit old-school, with lots of structures to navigate and lots of libraries to populate with languages and frameworks. These older systems can slow the process down, as many environments (both tech and talent) aren't as agile as they could be. Developers are ready to rise to the challenge, with limitless imagination and so many resources to release code swiftly, but they may feel handcuffed by these old systems.

Containers, then, solve a lot of these programming issues. They make the underlying stack irrelevant, resulting in a normalized, speedier deployment. Containers also work in tandem with microservices, so you can roll out task- and feature-specific apps instead of an entire suite of software solutions, of which only a percentage may be used.

With containers and microservices circumventing monolith development environments in favor of cloud technology and quick-to-market product releases, many enterprises are lauding containers as the solution to all their programming problems – but it's a mistake.

Containers can't solve everything

Buyer beware: any solution that seems to solve all your problems is just not possible. Indeed, containers offer cleaner boundaries, abstractions, and handoffs, but they are no silver bullet. Like many major technology trends, containers are an appropriate tool, especially when used in the right environment, but they shouldn't be relied on in every situation simply because they are having a moment.

Here are a few problems containers alone can't solve:

- **Containers and microservices relocate complexity—but they don't minimize it**. A common belief is that monolith systems are complex and unnecessarily complicated, which slows down production, and containers can solve this by offering speedy deployment. In reality, it won't take your developers long to realize that containers and microservices are just as complex. Deploying containers for every single new feature can result in severe sprawl, which can be further exacerbated by a non-collaborative culture. Such sprawl also frequently results in technical debt, because once something is deployed, it's often forgotten. This debt is something you'll eventually have to spend time in order to fix (see the next point).
- **Containers can be tricky to evaluate**. Of course, containers can help you bypass problematic areas in order to stay on schedule for a release, but you may quickly forget about those problems as you work towards the next new release. This mindset can make it difficult to evaluate your true progress within a larger project, especially when the problems can't simply be "fixed" or "replaced". Developers point out that orchestration – itself a necessity for containers – is often just the start of a solution, but without an overarching evaluation or management tool, containers are unwieldy.
- **Containers don't support collaboration**. Containers are often deployed in agile or DevOps environments that promote continuous improvement, but the containers themselves don't support collaboration. While they can help define or provide structure from a programming-versus-operational standpoint, they don't make your talented engineers, programmers, and system administrators communicate any better or easier. If you're looking for true collaboration among your programming and operational talent, you'll have to work harder than simply reaching for containers.
- **Containers can't solve human problems**. Containers and microservices are great in plenty of situations, but they are merely tools. They can't fix organizational or cultural problems simply because you deployed features quickly. Adopting containers to improve your products may encourage your developers, but if you don't have a collaborative culture among your developers, your operational staff, and even business-minded employees, containers can't mend that rift.

What's the bottom line?

Many engineers agree that monoliths themselves aren't the problem, just as containers themselves aren't inherently a solution. As with any tool, it is *how* we work with them that proves either successful or hopeless.

In the classic equation of "*culture > strategy > process*", containers and their related microservices are in the bottom rung of processes. With neither the right strategy nor a culture that promotes the strategy, no process will last long.