SentinelOne blog

GET A DEMO

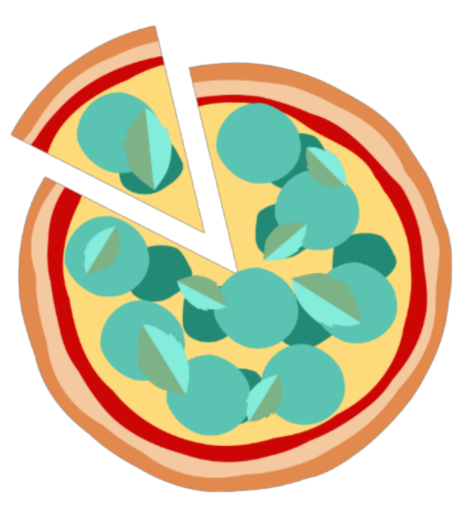
5 Critical DevOps Practices

January 23, 2018
by SentinelOne

DevOps is like pizza. We can't think of pizza without considering critical ingredients: dough, sauce, cheese, and your preferred choice for vegetables and proteins. Everyone likes different toppings. In my case, I can't think about pizza without extra cheese and meat. You might choose differently, but I think we can agree there are some ingredients that are critical for this food to be called pizza. Quality and ingredients will vary, but some things will always remain true.

Well, it's the same with DevOps practices. There are some critical practices, and you can't think about DevOps without considering them. Everyone will have preferred choices regarding the tools and the process, but the practice will remain and each practice complements the other.

Every critical DevOps practice takes time to get down, but the end result will be magnificent. So, let's discuss what they are and how to implement them.



1. Involvement From Everyone In the Process

Group involvement is like the dough of our pizza. It's the foundation, the base. This is the DevOps practice that you need to start before any other. Gone are the days when you would define DevOps as a collaboration between **just** development and operations. What about QA? What about SecOps? And more importantly, what about management? In order to get a good base for DevOps, you have to stop thinking about teams or people. Everyone involved in the process of developing the product should have a mindset of collaboration.

If this doesn't happen, you'll constantly have people questioning everything or ignoring important aspects of the project. It's not like you're working with saboteurs—no one's trying to destroy the initiative. But people have to understand the impact of their actions. When you involve everyone in the process, people can grasp the whole picture. And just like that, you've laid the foundation for DevOps.

2. Trust Only What You Have In Version Control

Trusting only what you have in version control is like the sauce of our pizza. It's the first ingredient you put on it. The flavor will vary, but it's a critical practice.

At work, every time someone interrupts me with a question about a certain functionality—especially one that I coded a while back—I answer with what I remember, but I always add, "Don't trust me, though. Let's go and see what's in the repository." Now it's not just code. You'll have to put in version control infrastructure specifications, dependencies, and configuration templates. Some of those configuration values will be visible to everyone, and others won't. It becomes critical to have it just for the sake of repetition. Repetition and consistency are key.

The only source of truth should be what you have in version control. Avoid doing manual changes. If something needs to change, it should be done here. Not anywhere else. The team will then know the "what," "when," "why," and "who" when it comes to changes, and that knowledge is the difference between a team with poor communication and a high-functioning DevOps team. As this practice becomes routine, team confidence will increase. There's nothing like trusting the process.

3. Be Confident With Automated Testing

Cheese! Automated testing is like cheese. And there's no pizza without cheese. You can't afford to not have the practice of automated testing in place.

Forget about DevOps for a moment. This practice is critical now more than ever in our industry. If you want to succeed and deploy with confidence, a good suite of tests you can trust is crucial. You need tests that will fail if the app is not providing the expected output. Otherwise, the team will be slower to call something is done, thanks to manual testing.

Without the practice of automated testing, all DevOps efforts will be compromised.

There are several types of tests that you can choose, and today's not the day to talk about them. But know that the speed of the tests is a critical factor. If it takes too much time to receive feedback, everyone will avoid them. Some people might go to the extreme of commenting them out or removing them. As long as almost no human intervention is needed and they run fast, your tests are good.

Without the practice of automated testing, all DevOps efforts will be compromised because you'll instead be dealing with a constant barrage of errors.

4. Regularly Integrate Your Codebase

The regular integration of your codebase is like our pizza's vegetables. In order to have a balanced diet, you need to include them. As kids, we didn't like them that much, but as adults, we realized that we need them. You might not like this practice at the beginning, but you will after time. This practice consists of regularly integrating your codebase into the mainline, trunk or master (depending on the version control tool). Avoid long-lived feature branches, and integrate your code at minimum once a day. Everyone should always have the same working copy of the codebase each morning.

Things can go wrong, I know. That's why, in addition to integrating your codebase, you'll need to have an automated way to check if everything went well. Compile the code, do your static code analysis, check that all unit tests passed, and then pack to deploy the same code everywhere. You'll build only once. If you don't do this, you'll add more time to delivery, and you might encounter new problems every time. This practice might sound crazy if it's the first time you've heard of it.

Architecture plays a big role here. You will need to have a loosely coupled one. Things need to be easy to change and evolve all the time.

Are you still in doubt that this is necessary? I understand. The first time I read about this practice, I thought, "It's not for me." Try it out and see the difference it can make. You can leverage feature flags to always have the option of turning off what you've checked in.

5. Deterministic Deployments in All Environments

Deterministic deployments in all environments are like proteins. A good pizza will have them, and so will a good DevOps implementation. I never get tired of telling people that deployments should become a boring task: predictable, without randomness. You'll need to have the above practices in place in order to practice this one, though. It's about having a deployment pipeline. There are gates that require the human skill of critical thinking in order to continue.

You don't want to be surprised on release day.

It's of the utmost importance that the delivery pipeline is the same in every environment. Remember, you don't want to be surprised on release day. Development, QA, staging, and production should have the same pipeline. For development, you'll want everything automated without anyone approving. In QA, you'll want to have a bigger suite of tests approving it. Staging might be the same but with a different suite of tests, usually that take more time to validate. And for production, it should be a business decision.

Let me tell you also that it's not just about code. It's about everything involved in the process of deploying the app in a working state. This means that aspects like infrastructure and configuration should be considered too. Processes like infrastructure as code and configuration management are critical. Of course, automation will always be key because the idea is to provide value to your customers in a timely manner. That's why this practice is critical. You can't let fate take care of this. There's nothing worse than calling things done when developers have delivered.

It's All About Providing Value

As I've been saying, it's all about pizza. Wait, sorry. It's all about providing value with this set of practices.

It's important to have at least these five elements in place in order to succeed. Practicing these will help to increase your team's confidence more and more with each release. Persevere through the pain of implementing something new, because the more you practice these, the better you'll become. Customers will be impressed when you're mature enough in DevOps that you deliver value to them — and in a timely fashion — each time they make a request.

This post was written by Christian Meléndez. Christian is a technologist that started as a software developer and has more recently become a cloud architect focused on implementing continuous delivery pipelines with applications in several flavors, including .NET, Node.js, and Java, often using Docker containers.

Like this article? Follow us on LinkedIn, Twitter, YouTube or Facebook to see the content we post.

Read more about Cyber Security

- Sexy But Useless DevOps Trends