lcjna@kth.se
Louis Nathan, March 30 2019

Why you should start using DevOps.

Introduction

You have probably heard the term DevOps used on Twitter, or read about it on Medium, but you were never quite sure what it actually meant. DevOps stands for Development (and) Operations, and is a term used to talk about the bridging of software development and the operation of the product both in terms of organisational culture and software tools.

What this means in practice is having agile infrastructure that makes it fast to deploy changes in your software to your production environment. The big ideas around DevOps tooling are what is known as Continuous Integration and Continuous Delivery. Integration means your code is always being pushed to the central repository and tested. Delivery is about delivering your central repository out to your users in a quick yet safe way. Continuous is the fact that this should be happening constantly with as little manual interaction as possible.

Tooling is great, but there is more to DevOps as you will soon learn. Making changes in your organisations culture is another critical aspect of DevOps that will make your engineers feel ownership of your product and increase productivity.

DevOps is best implemented at an organisational level but does take time for that to happen, so we will look in to some methods of implementing DevOps at a smaller scale starting with your team. In this article we will look in to why you should use it, how industry uses it, and then how you the reader can use it.

Benefits

So why should you start using DevOps? Well the State of DevOps Report from 2015 shows clear benefits to your team and organisation by adopting the DevOps culture and way of collaborating. The report states that IT teams utilising DevOps have up to 30x more frequent deployments, 60x fewer failures, 60x higher change success rate, and 160x faster recovery. More importantly, they are twice more likely to succeed in profit, market share, and productivity goals (Puppet 2015).

Value is delivered faster with a DevOps-improved development cycle and there is no easier sell to your higher-ups than this fact. The cultural benefits are also great for your teams, resulting in higher employee engagement, greater professional development opportunities, and happier, more productive teams (Puppet 2016).

DevOps Digest collected feedback from senior DevOps professionals that stated the following benefits. The alignment of IT and business, putting the business needs on the forefront of the software development to realise business goals by shortening feedback loops and putting the customer needs first (DevOps Digest 2016). Velocity is also stressed as a key benefit of DevOps – it is not enough to just develop products; they must be the right products that are in the right direction of what the customers need.

lcjna@kth.se
Louis Nathan, March 30 2019

Industry
The software development industry has known these benefits for a while now and DevOps is currently embraced by more and more companies each year (Puppet 2015, Puppet 2016).To give an overview of how companies utilise DevOps let’s take a look at some of the big names in tech, particularly unicorn start-ups which are more open to adopting new methodologies.

The way industry uses DevOps can change radically between companies as the needs and interpretations of the concept differ. Dave Hahn, a Senior Site Reliability Engineer at Netflix said “How does Netflix think about DevOps? Well the truth is, we don’t. […] The result of the Netflix culture looks a lot like DevOps” (Netflix 2018).

Netflix sees DevOps as an ideology that offers freedom and creative flexibility to get the job done with trust. It is not a set of tools and patterns that can be applied to their technology, but rather a cultural phenomenon that inspires and motivates their engineers to be their best. The operations engineers at Netflix develop tools that make it as seamless as possible for software developers to push to production, without having to worry about instances or operating systems or configuration files.

Meanwhile at Atlassian, DevOps is “a culture and environment where building, testing, and releasing software, can happen rapidly, frequently, and more reliably” (Atlassian 2016). They say that DevOps is not the sole responsibility of one person, but a shared responsibility on the entire team. Atlassian achieves the shared responsibility by including all disciplines in the feedback, plan, and build stages.

Rideshare company Lyft also utilises DevOps, by using their DevOps engineers to “[drive] consistency across all of Lyft’s services to create an environment that allows their developers to work autonomously. They call this “Scaling DevOps” (Saltstack 2015). Lyft reiterates the concept of making developers responsible for the work they are creating under the “if you build it, you run it” philosophy. What this means is the person who creates something is also responsible for getting it to run in the production environment, ensuring a sense of ownership while building cross-disciplinary skills.

Now that we have a few examples of how industry thinks about and implements DevOps we have identified some common themes. DevOps in industry is defined as a shared cultural and tooling mindset that focuses on productivity and ease of release by empowering developers to take ownership of their product and feel like they are genuinely contributing to the products success.

How - Culture
So how do you start implementing DevOps in your organisation? Here are some ways you can start developing the culture of collaboration and ease of development that we now know is fundamental to DevOps success.

TechRepublic recommends integrating the teams within your organisation to create an environment open to collaboration on an equal field, with common goals (TechRepublic 2018). The key to creating this environment is making sure everyone understands they are responsible for the success of the project. Developing in a consistent way across an organisation makes it easier to make changes and avoids siloing of information – something that holds back development speed across teams.

Another important aspect is having team members who can operate outside their normal area of knowledge. Utilising the "You build it, you run it" philosophy will develop skills in operations and give a better feeling of ownership of the product compared to having dedicated operations engineers who abstract the developer from the end result.

Building this mindset will empower your team members to work on any aspect of the product which improves motivation and productivity. If you followed the previous step of creating an open and collaborative environment this will happen naturally in your organisation.

Having the whole team learn together is another way to build up a successful DevOps culture. There is no point going out of your way to learn a better way of doing something for your team if no one else understands and it ends up costing more time than without having done so. Bringing everyone up to speed on the latest tools and methods will make each member feel valued and strengthen their feeling of contribution to the success of the project.

How – Tools

There is also the technical tooling side of DevOps. Not every tool needs to be picked up and implemented right away, simply becoming more aware of what can be done is powerful in that it gets you thinking of where you can start to fit these improvements in. Here are some ways you can improve your existing workflow to facilitate ease of development, deployment, and feedback.

As Dave Hahn said earlier, Netflix makes tools that make it easy to develop since you no longer have to worry about cloud instances, config files, or which operating system version is being ran. If you are currently a developer that has to worry about these when you want to push your code to production, there is an opportunity here to use DevOps techniques to find a way to automate this or simply take it out of the equation.

If you are tired of spending time figuring out how to get your application to run on different types of computers and fiddling with configuration files that differ on each version of the OS then containers are your new best friend. A container, according to Docker the leading containerization platform, is "a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another."

Containers let you package your environment up into a container image that is then deployed with all the dependencies and settings you have already defined, saving time by only making you write it just once. It takes seconds to set up the environment on a new computer that utilises the Docker engine and your pre-existing container images.

Continuous integration as I mentioned earlier is the practice of merging your code changes as often as possible to the central repository. This ensures that your code is tested often and will have smaller conflicts (if any). If you have ever had a patch day where everyone merged their code together and it took hours to fix the merge conflicts, this is the solution.

Manually running tests during continuous integration can be time consuming and it can often be skipped. To ensure your code is always tested while also making it easier for yourself test automation is another big improvement you can add to your workflow. There are many ways to automate tests, including different styles of where it happens and what happens afterwards. The main idea though is that once you push your latest commit, it automatically goes to a testing server where hundreds, or more likely thousands of tests are run against your code to ensure nothing breaks the current build.

Unit tests, platform tests, acceptance tests are all automated once a new commit is pushed. If the automated tests pass, you can set up continuous deployment which automatically deploys the test-passing code to your production environment. If the tests fail, the commit is rejected and the developer responsible is notified so they can fix the problem before re-submitting the code to be tested all over again.

You can find more information on automated testing by looking up automated interface tools such as Selenium, Unified Functional Testing, and TestComplete as a starting point. These are good for functional testing, but automated unit tests should be the backbone of your regression test infrastructure. The tools for unit testing change depending on your platform and new ones are always being created so I recommend starting out by searching "[Your programming language] unit test automation" in Google.

To set up continuous deployment you need to configure your deployment pipeline to be able to be automated. This allows you to quickly release new builds of your software, especially after you have implemented continuous integration. Deployment does not have to be fully automated but by creating a release environment that lets you deploy at the press of a button is a powerful tool. Atlassian has a great article on the extended benefits of continuous integration and deployment titled "Continuous integration vs. continuous delivery vs. continuous deployment" (Atlassian 2016).

Conclusion

Now we know DevOps is both a cultural perspective on development as well as some tools and processes that allow that culture to flourish. You have some ideas to think about automating your worries away and where to find out more about these tools. You also have insight in to how the industry uses DevOps and the key focus on culture.

While DevOps is an ever-changing idea, the concepts behind it will be ever beneficial – especially the cultural improvements for the work environment. A developer who feels they are responsible for the end product and has the freedom and ability to do so will outperform those that do not. For further reading look in to this wonderful article by Irma Kornilova titled "DevOps is a culture, not a role!" (Irma Kornilova 2017).

lcjna@kth.se
Louis Nathan, March 30 2019

References

Puppet 2015, *2015 State of DevOps Report*, Puppet, viewed 23 March 2019, <https://puppet.com/resources/whitepaper/2015-state-devops-report>

Puppet 2016, *2016 State of DevOps Report*, Puppet, viewed 23 March 2019, <https://puppet.com/resources/whitepaper/2016-state-of-devops-report>

DevOps Digest 2016, *25 Advantages of DevOps – Part 1*, DevOps Digest, viewed 25 March 2019, <https://www.devopsdigest.com/devops-advantages-1>

Netflix 2018, *How Netflix Thinks of DevOps*, video, YouTube, March 19, viewed 22 March 2019,<https://www.youtube.com/watch?v=UTKIT6STSVM>

Atlassian 2016, How Atlassian does DevOps, Atlassian, viewed 24 March 2019, <https://www.atlassian.com/it-unplugged/devops/how-atlassian-does-devops>

Saltstack 2015, *Scaling DevOps at Lyft*, Saltstack, viewed March 25 2019, <https://s.saltstack.com/scaling-devops-at-lyft/>

TechRepublic 2018, *5 tips to developing a successful DevOps culture*, TechRepublic, viewed 25 March 2019, <https://www.techrepublic.com/article/5-tips-to-developing-a-successful-devops-culture/>

Atlassian 2016, *Continuous integration vs. continuous delivery vs. continuous deployment*, Atlassian, viewed March 26 2019, <https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>

Irma Kornilova 2017, *DevOps is a culture, not a role!,* Medium, viewed March 30 2019, <https://medium.com/@neonrocket/devops-is-a-culture-not-a-role-be1bed149b0>