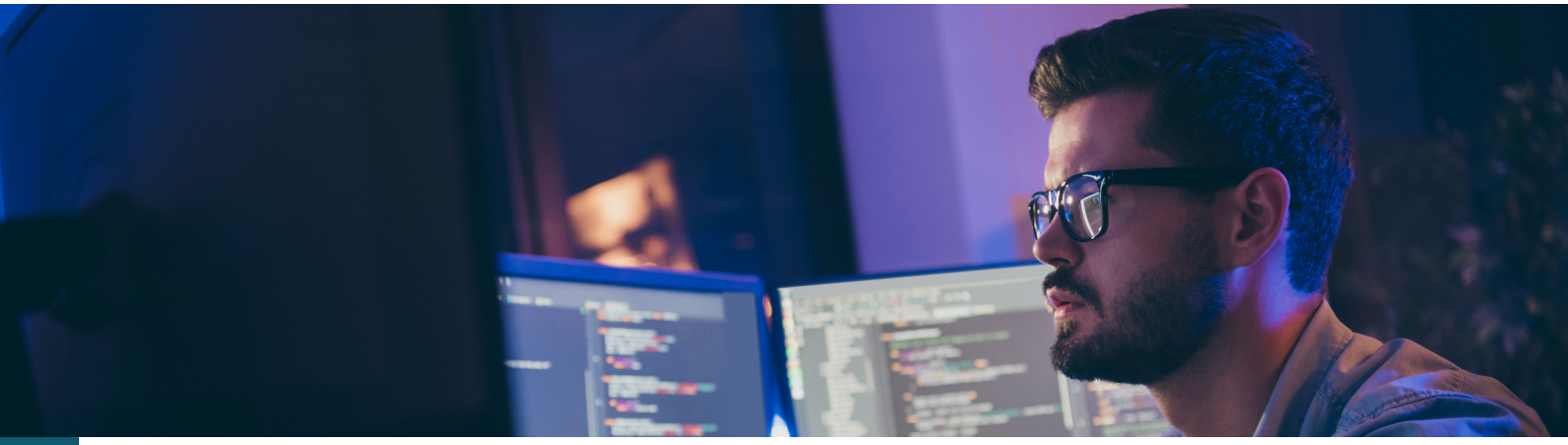
Strypes

DevOps: The big brother of agile

Building Smart Solution that Create Business Impact

DevOps ensures alignment between code production and service delivery

+31 6 156 883 17 business@strypes.eu



DEVOPS

Regardless of whether you've been in the tech business for a while and need to reconsider your product development practices or you've just founded a tech start-up and want to do everything the right way – you've most likely heard of **DevOps.**

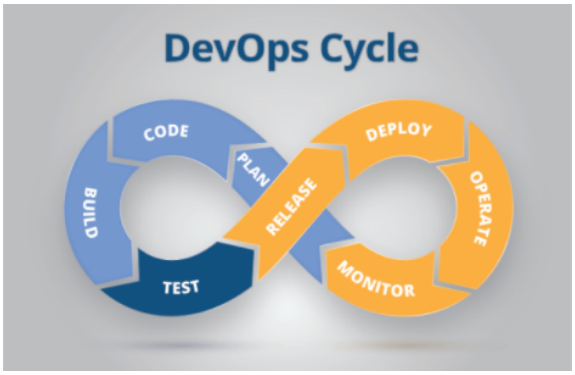
DevOps is a popular and fancy word nowadays, and it seems to be used without an understanding of what really stands behind it. The intent of this paper is to hopefully try and change that.

FRAGILE TO AGILE: WHAT IS DEVOPS, REALLY?

DevOps is commonly identified as "a fancy new profession". Many believe that this role can simply be given to System Administrators, because they know the product and the infra-structure; as well as most of the tools "DevOps people" use. But the truth is DevOps isn't just a role. In fact, in some companies DevOps might not be a separate role at all. DevOps is a product development philosophy and methodology, which can (and should) be employed by the whole organization.

Not too long ago, the classical structure of a software development company was split in silos. We had development on one side – planning, creating and launching the product; and then operations on the other side - supporting the app and the infrastructure. There was nearly no collaboration between these departments. The problem is that, in the long run, Dev and Ops start contradicting each other. The Dev team might want to deliver as many features as possible, as fast as possible; while Ops tries to hold things under control - "If it works, don't touch it."

DevOps eliminates the clash, and serves as a bridge that facilitates communication and encourages mutual effort to be faster and more efficient with product development. To achieve that, DevOps makes great use of Agile principles, similar to a new extension of the Agile framework. What's new and improved is that unlike Agile, DevOps focuses not only on code production but service delivery as well. It preaches that product development should be done with use, scalability, and support in mind.



+31 40 311 6025 business@strypes.eu

DevOps makes it faster to push new features and iterations to your customers, without compromising on quality, stability, and security. This is done by introducing three game-changing methods of work that build upon each other.

1. **Continuous integration (CI)** – this practice helps get rid of the original approach to merge changes only when it's time to release. It applies the rule that developers should merge their work at least once daily, but preferably - as often as possible. Continuous integration helps you make sure that each change won't break the system once committed into its main branch, by taking it through validation and testing. CI really puts an emphasis on test automation and high test coverage. It's faster and less risky to test iterations one by one and fix them on the go, than to test every- thing at once before a release, and then to have to go back and deal with multiple bugs. Ideally you can alter the good influence of continuous integration even more, by implementing Test-Driven Development **(TDD)** into your methodology as well. TDD suggests that you develop tests first, and then you write the production code in a way that matches these tests. It's a bigger investment in the beginning, but pays off in time by yielding higher test success and lower release failure rate.

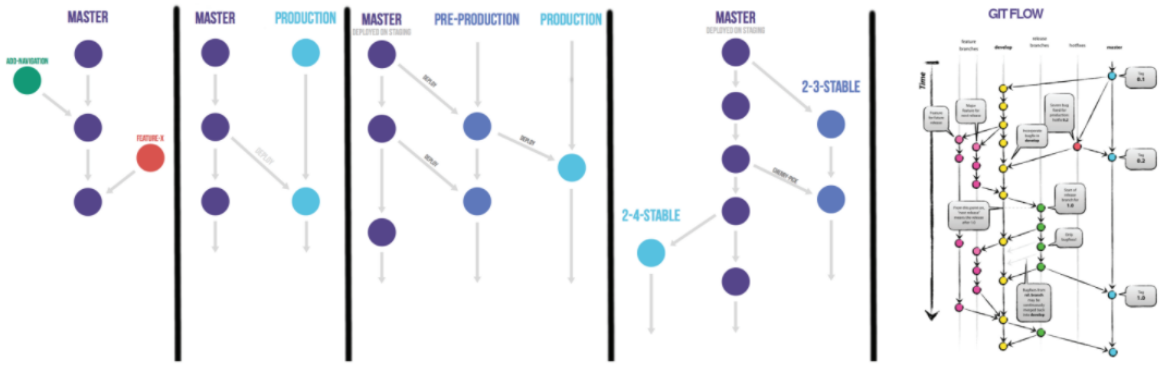
2. **Continuous delivery** is an "upgrade" of CI. Aside from automating your testing and validation process, you embrace a cyclic approach to work (you build and test periodically), and you also automate the release process, using Application release automation **(ARA)** tools. This kind of workflow ensures that your source code is always in a deployable state, and can be released literally with the push of a button.

3. **Continuous deployment** - People often confuse continuous delivery with continuous deployment, especially because the same abbreviation is used for both **(CD)**. In reality, they are similar but continuous deployment is the "next level" approach. Both practices ensure your code is in a deployable state at all times, however with continuous delivery you deploy your releases manually, while continuous deployment goes even further – every build that passes testing and validation, gets deployed to production automatically.

While continuous deployment is implemented by choice, and might not be practical for every company out there, continuous integration and delivery are a must if you need a proper DevOps setup.

THE TOOLS

As you might have noticed, DevOps is all about efficiency - optimizing and automating. This can't be done without the use of appropriate tools. To be able to run this software development "machine", we take advantage of the following software:



+31 40 311 6025 business@strypes.eu

1. Version control systems

(Git) In DevOps you perform version control of both the code base and the infrastructure. Version control is important, because it allows you to track history and roll back to a previous system state in case of an incident.

2. Continuous integration and delivery tools

(Bamboo, Jenkins) These tools automate every part of the software development process that doesn't necessarily have to be performed by a human (such as the build jobs and continuous integration).

3. Test automation tools

(like **Selenium, JUnit, SoapUI)** Selenium automates web browser interaction by testing screens, forms, buttons, and all other parts of the interface, in the same way that a human user would use them. JUnit (and similar tools of the xUnit type) allows for automated unit testing, whereas SoapUI automates web services testing.

4. Configuration management and orchestration tools

(Puppet, Chef, Ansible) These tools are used for automating the setup and configuration of devices, systems and even environments - also known as orchestration. In combination with Virtualization and Cloud hosting (explained further below) you can (re)create and subsequently manage your Infrastructure as a Code **(IaC)** with the push of a button. Automated configuration changes are tracked in Git – so version control is performed for the infrastructure as well. With these tools you can also describe the production environment and then copy it to create one or more test environments. This allows you to test and validate changes in one of the mirrors, without causing harm to the systems in production.

5. Virtualization

(VMWare, HyperV, KVM) and containerization **(Docker, Kubernetes)** tools Virtualization enables separation of a physical machine into multiple logical ones. Such virtual machines can host and run different operating systems and multiple business applications. They are easier to manage and can assist in achieving high availability and resource utilization. Containers are lightweight, stand alone executable packages that include everything needed to run your software, and reside on one common operating system. They make it possible to encapsulate an app and all its resources while separating its configuration and access, resulting in easier deployment and providing the flexibility to easily move everything to a new environment. They are enablers for rapid scalability and even higher resource utilization.

6. Cloud hosting

(AWS, Azure, Google Cloud) You can host the applications on your own hardware environment, on a rented cloud service, or even on hybrid solution - combining the best of both worlds. This depends on the business setup and the compliance requirements and regulations (such as ISO 27001, ISO 13485, etc.).

7. Monitoring

(Nagios, Zabbix, ELK) Monitoring is key for identifying issues with both your application, and your infrastructure. By using the appropriate tools, like Elastic stack (a.k.a. ELK), you can detect issues and collect visual, actionable information about them.

8. Workflow management and documentation

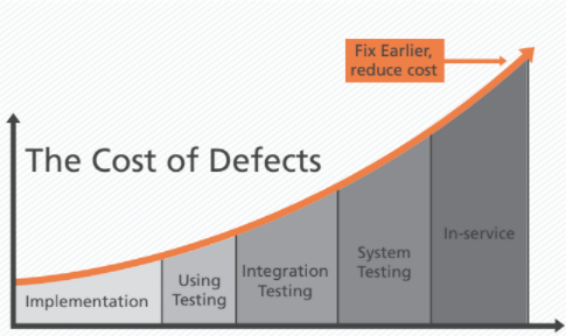
(JIRA, Confluence, Trello) DevOps is all about efficiency. Project management tools, such as JIRA and Trello, help you visualize various workflows, manage tasks and deadlines, track bugs and bug resolution, and run multiple projects at once. Documentation tools like Confluence help you keep a centralized database of your most needed documents: user guides, project descriptions, etc.

WHAT ARE THE BENEFITS?

If you weren't convinced by the introduction, here is a list of more specific benefits of DevOps.

Decreased time-to-market

The high efficiency of the DevOps model allows for fast and regular iterations. You can deliver new features to your customers and fix issues more quickly because of the cyclic, mostly automated workflow.



Decreased failure rate of releases

Finding a bug late down the line, or in production, can cost you a lot of hours (and money) to fix – you will have to go all the way back to determine its context. Since DevOps relies on automated testing of each change that's committed by your developers, you can detect issues at a very early stage. Fixing issues early on reduces the chances of release failure to a minimum. This saves you money and minimizes risk.

Easier support and maintenance

The close collaboration between development and operations ensures planning from both points of view. Operations can advise on future support needs and potential maintenance issues, and developers can make adjustments along the way as needed.

Easier scaling and improved security

With its cyclic character and frequent delivery, DevOps ensures faster scaling. Scalability is enhanced by having strong communication with Operations, who can introduce a contingency approach to product development, i.e. predict future demand and help Development plan system resources and architecture accordingly. Security features can be also integrated early in the development cycle.

Faster time to recovery

Continuous delivery and continuous deployment allow for easier system rollback. If you release a change that happens to impact the system, you can easily revert to a previous system state, while investigating and resolving the issue in parallel. Going even further, you can automate the resolution of specific events. Some companies do drill tests, a.k.a. stress tests - they intro- duce random issues to the system, usually in a test environment (but this can also be done in production), and they monitor the system behavior. In the next development cycle they set-up an automated response to those problems. Over time, the system will automatically diagnose and resolve various events, without making any impact on usual operations.

CONCLUSION

In short, DevOps is the big brother of Agile. If you are considering the implementation of DevOps as your new methodology, or you are currently relying on Agile and want to upscale – don't hesitate to reach out and schedule a meeting with us!

+31 6 156 883 17 business@strypes.eu

NETHERLANDS
Prof. Dr. Dorgelolaan 30
5613 AM Eindhoven (4th floor)

+31 40 3116025

NETHERLANDS
Kopenhagen 9
2993 LL Barendrecht

+31 40 3116025