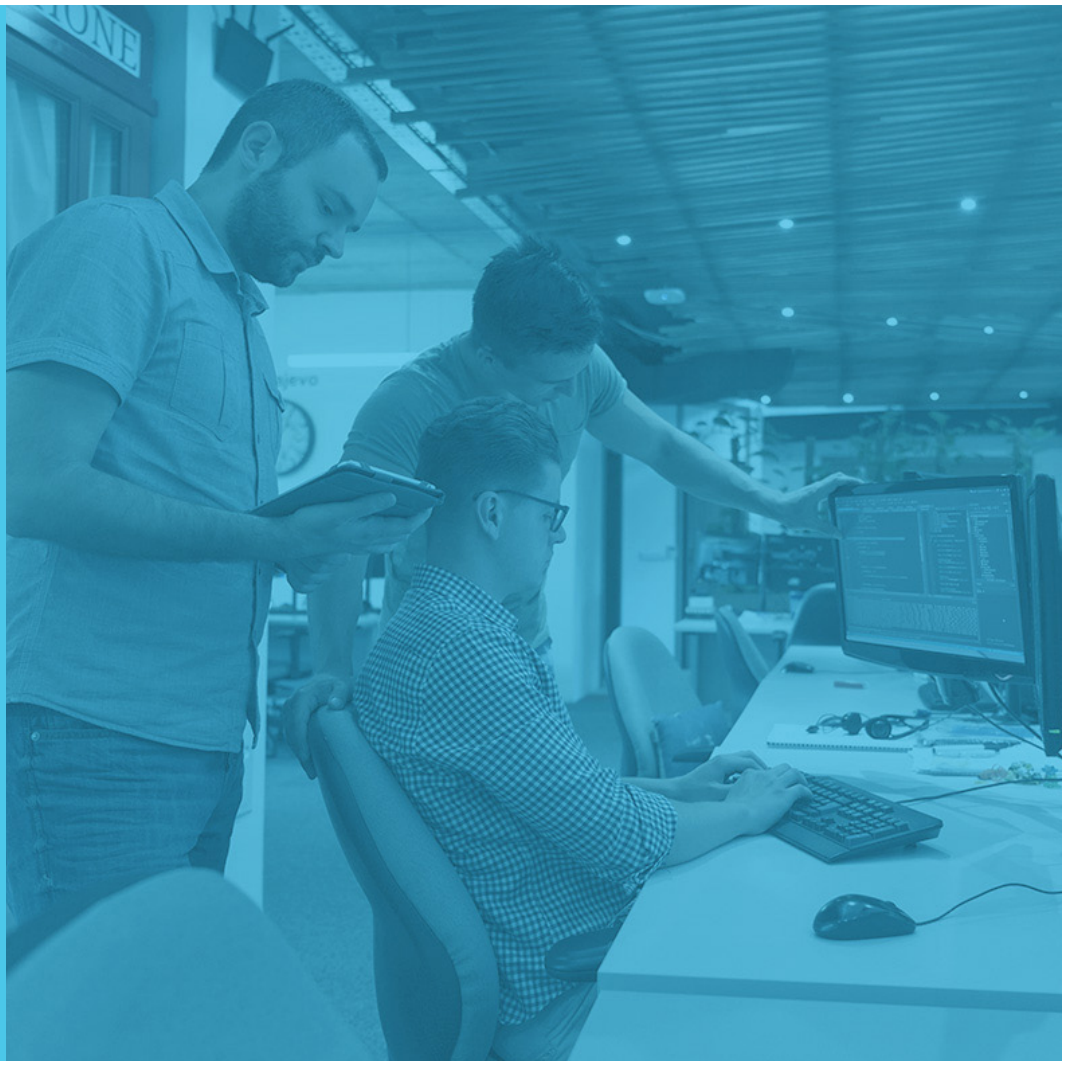
EMYODE

SecDevOps:
People, Process and Product

Part of your digital transformation



White Paper

Acknowledgements

Initiated and released by the Emyode DevOps Team, this document was developed with support from across the organization and in direct collaboration with the following:

Key Contributors

Jean-Paul Lizotte (Emyode)
Eduardo Mazza, PhD (Emyode)

Reviewers

Maxime Plante (Emyode)

Feedback

Please send comments or suggestions about this document to the Emyode feedback alias (info@emyode.com).

Legal Notice

The information contained in this document represents the current view of Emyode Inc on the issues discussed as of the date of publication. Because Emyode must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Emyode, and Emyode cannot guarantee the accuracy of any information presented after the date of publication (September 2018). This White Paper is for informational purposes only. EMYODE MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Emyode Inc. Emyode may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Emyode, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2019 Emyode Inc. All rights reserved.

Table of contents

Preface

The Emyode Centre of Excellence: a SecDevOps curb to DevOps.

Working closely with contacts in a variety of technical, support, and field roles, the Emyode Center of Excellence has spent a great deal of time researching the subject.

This paper is based on the results of a detailed analysis consisting of various ways to improve software delivery with people skills, process improvements and proper tooling.

— Jean-Paul Lizotte

1. Executive Summary

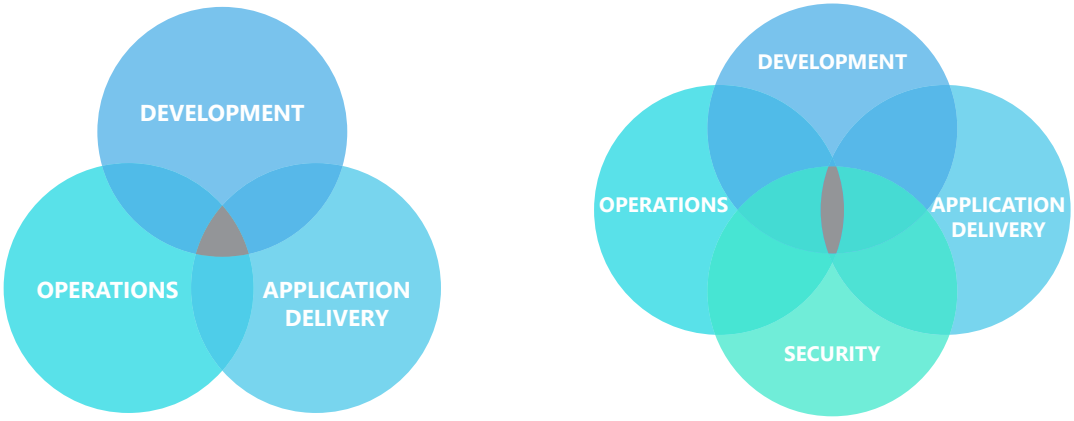
DevOps is a recent field of expertise that balances an art of human values and the precision required for efficient software delivery. It is, at its base, a corporate culture movement meant to revolutionize the ways things are done, with transparence, objective measurements and a “quick win” low-risk approach to automation.

We believe the use of SecDevOps (aka DevSecOps) as a term can be a bit of a misnomer. It is our opinion, that DevOps which would seem, as opposed to SecDevOps, to exclude security processes in its conception, is not an added value to software delivery without quality assurance, and quality implies security.

So in a sense, for us, and hopefully the nascent DevOps community, any DevOps practice should just as well consider security as an integral part of its processes. The only reason that we agree to use the SecDevOps title or term, is to demonstrate and indeed identify the discrepancies between traditional DevOps and one that had integrated security assurance in its processes.

2. A DevOps overview

The following diagram expresses the basic tenet of a DevOps process as compared to the SecDevOps (aka DevSecOps). In its simplest expression, SecDevOps includes a concern for security. DevOps considers improving Development, Operations and Delivery alone.



DevOps VS. SecDevOps

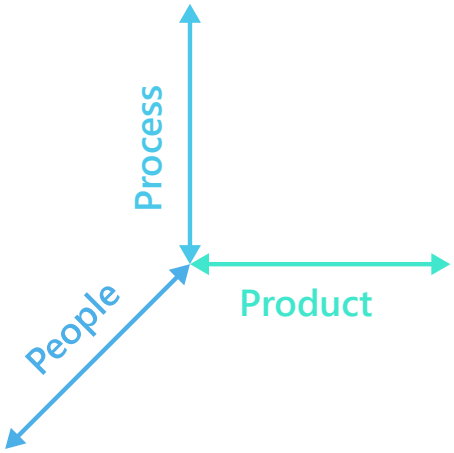
Key Points

Before deep diving into our approach to SecDevOps we want to review a few key related concepts:

2.1 Three dimensions

To cover the areas required by SecDevOps we must first consider the key points where DevOps processes usually intervene. If you are to consider DevOps as a growth space, the three principal axes, replacing X, Y, Z, are: People, Process and Tools (aka Product).

All the axes will play a key role in increasing effectiveness in the software delivery.



DevOps is commonly described as... "the union of people, process, and product to enable continuous delivery of value to our end users" — Darwish 2018

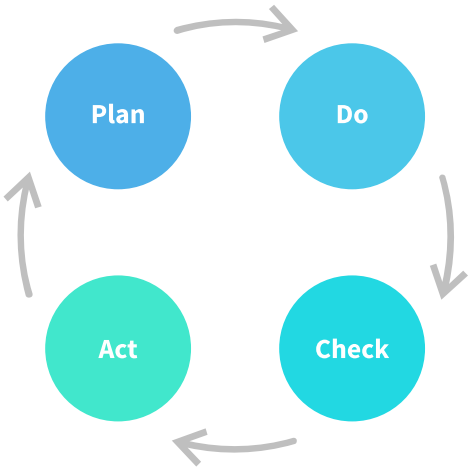
2.2 The lifecycle or "PDCA (plan, do, check, act) cycle"

All the previous dimensions will be able to grow or improve according to their progress. But to achieve progress, it is necessary that action be taken.

Thus 4 parts of a basic cycle can be defined. If people are the force driving the growth, this process certainly is the lever on which they act.

"The PDCA cycle influences the functionality of the software, the release plans, requirements, testing, environments, and the process of delivering the software." — Sharma, 2014

As you can see, there are four points to the cycle, each representing two checkpoints for reflection and two for action. Hence, a balance between acting and thinking.



The planning is certainly the most abstract. Though certain methodologies such as Agile, Scrum, CMMI and Kanban are helpful in planning and bettering software development, it does not cover continuous improvement for delivery.

Hence, one can consider DevOps as a practice, as a culture change, and for each item planned, there should be an action, a measure of the success (the delta between what is delivered and what was expected), an adjustment and a lesson on how to change to ensure things will improve.

Objective measures, such as defects discovered, delivery times and customer satisfaction, for example, allow to see if the planning or the checks were effective in proving what was done.

Note: Sometimes this loop includes second phases, such as problem finding, displaying them, and clearing the problems and acknowledgement. There is not just one "right way" of doing things.

It is exactly this mutability that will be exploited in order to attain "DevOps" and ergo reuse in SecDevOps.

2.3. Communication changes

"DevOps requires that every project have clear communication channels between development teams and individual." — Vickrey, 2017

Another of the proposed changes that DevOps brings is the abolition of work silos or, classically, calling it an "increase in visibility". In fact, strictly speaking, the sharing of information along the different actors in the production of software as in different domains can easily be viewed as a tool for improvement.

Two of the main factors argued by the DevOps proponents are:

1. That the ability to see what is good in how things are done and the ability to have a stake in what needs to improve have a positive effect on team morale. It is also empowering for the actors to see the evolution and delivery of their work.

2. Having their hand on the data, that will allow the actors to improve their work and allow for autonomy, hence the will to breakdown silos. And, also, having all the data flowing between all the actors required to deliver software, removing any dependencies on bureaucracy (or artificial blockers) as the case may be.

   Presence of the actors, working as a single team is another way of removing silos. If having a free data flow is useful, it is ever so if all actors can communicate verbally with each other at critical planning points.

If we consider Agile as a planning process, during the "planning" phase and grooming, the proposal under DevOps is to add, for instance, infrastructure or "operation" actors into the foray. Physical proximity, or the simulation thereof, is the planned way to reduce delays in communication in these phases.

Traditionnal siloed work VS. DevOps - Open communication

2.4 Persistance of data

The last and very important part, additional to the breakdown of communication blockers, is to have data accessible when needed. This can best be done by making all shared data accessible on a common platform. This brushes the subject of "tools". Having the proper tools to make the delivery easier and safer is one of the uses, tying in to communication, having a better way to make all the communications and delivery data relevant.

A working, collaborating and data sharing platform is highly recommended. As much as we want to eliminate the silos that can affect effectiveness of a team, we want to also ensure that knowledge is shared among team members. Hence avoiding yet another liability in communications.

A configuration management database is a strong tool to leverage for proper persistence of the requirements for continuous improvement.

A trifecta, to hold it all

While people, process and tools have already been targeted for continuous improvement, here we will enumerate some of the main techniques used to do so.

The reason why we focus on these elements is that there are common points between DevOps and SecDevOps when it comes to these considerations.

2.5 People

We've already covered how breakdown in communications is a factor in human collaboration. One of the ways to improve communication and efficiency is to encourage proximity, as described also earlier.

The specific technique often used to break down a major barrier is to have the "ops" teams' members participate at the inception of the project and all along the different planning phases of the software development stages.

Another technique is to give latitude and autonomy to the developers as to building infrastructure (sometimes known as "infrastructure as code").

Distributing merit and liability among all the team members helps in bringing motivation and awareness of the traditional individual roles' usual issues. Everyone sees the chain as a whole and how each link is critical. This is as much true for security as it was for the "ops".

2.6 Process

Having some formality, in the form of a workflow or a documented (and evolving) process is also considered an effective way to improve efficiency. Since this process is treated as an artifact, just like when code changes, we can monitor and audit what changes had which effect on performance.

"A process is a series of steps and decisions involved in the way work is completed." — Baird

It is difficult to express in simple terms how this can be crucial in the success or failure of a software delivery process. If the people are the raw power behind a project, the process is a contract or a recipe for success.

Understanding that the software development process itself (DevOps) equates to a method to rule the software delivery project is a necessity. It exists to create order and structure in the daily work and to ensure nothing known or visible is left to chance. All these controls are directed at delivering value to the end user.

"When it comes to optimizing your delivery model for speed, it's essential to take a look at your processes and "cut out the fat". To me, this equates to an organization's ability to make as many repeatable processes as possible, while also eliminating as many manual tasks as possible in order to accelerate the process from code check-in to production." — (Gene Kim, 2013)

2.7 Product (Tools)

There is certainly a great deal of enthusiasm in the industry for this part. A special focus on automation and artifact control is often cited as the main reason to start a DevOps endeavor. Arguably, the whole idea of having a software platform in business is to speed up processes and impose some rigor when applying business workflows. If we adhere to this argument then, certainly, having software tools to do just that while developing software seems necessary. And indeed, there are already many products to do so. Having them in the delivery phases is no less relevant.

The "tools" are also what are used to keep track of the progress or any regress in the software delivery cycle. In every company, there is a constant effort to improve the efficiency of processes using tools, including software solutions. It is therefore more than reasonable to think that, upstream, software creators use it themselves.

(The specifics of these techniques are out of scope for the needs of this paper, so we will not get into them any deeper. We just need to keep them in mind to understand how SecDevOps will relate to them.)

3.0 The SecDevOps Arch

We've spent some time understanding how DevOps principles are meant to affect software development and delivery. To better see why these two principles are actually very similar, we've tried to establish some of the nuances and basic approaches meant to improve the software development process.

"DevSecOps is like DevOps, but with security principles baked in." — (Matteson, 2017)

What has changed

The changes are either, fortunately, subtle enough to be easily implemented or unfortunately, subtle enough to slip through the same considerations, which should make their implementation simple.

To be clear, simple is not necessarily easy. Just as the implementation of DevOps practices are meant to be done at an incremental rate, it must be done carefully so as to not negatively bias the production of an effective software delivery system. This was a very good lesson learned described in the book "The Phoenix Project".

Thus, if we are to consider any change in the ways of doing them, they are worth being well measured as added value and how well they are connected to the business priorities.

Certainly, it is empirically evident that security in information technology is important. To integrate (intrinsically) security requirements in the software development process is certainly a laudable goal, but it is easier said than done.

This is where the proximity between DevOps and SecDevOps becomes more apparent: certainly, if DevOps is a practical thing that can be done, might we not simply use the same basic techniques to implement SecDevOps? We at Emyode think so.

Revisiting the lessons from the past

In this next section we will see how some of the original propositions for implementing a DevOps culture also applies to a SecDevOps one. All the while highlighting some of the differences and potential additional challenges.

The irony, if any, in fact is that the DevOps mentality creates a perfect staging area to insert security in the three axes we described earlier. With small incremental changes, it becomes very attainable.

3.1 SecDevOps and People

What have we learned about people that applies to SecDevOps? There should be proximity between security operatives in the IT space and the software development teams. This is a deceptively simple statement.

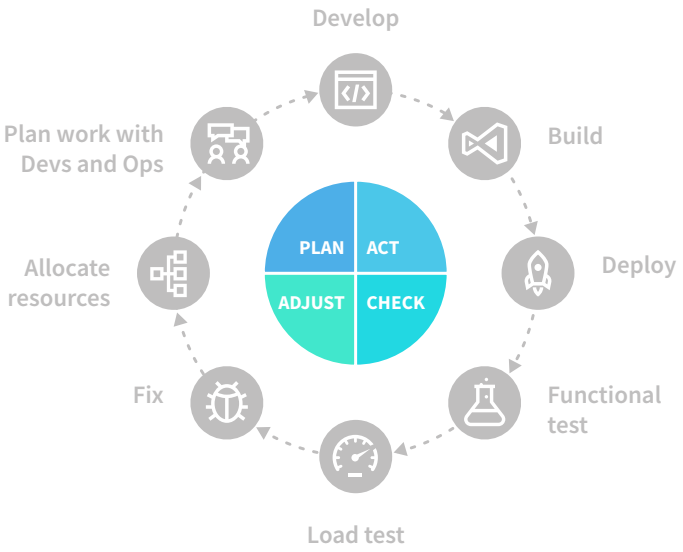
Indeed, when we consider that there might be approximately one to six security administrators per 100 developers (Aubuchon, 2010), it becomes quite apparent that we might encounter a difficulty getting these two teams to spend time together.

One of our proposed solutions is an awareness campaign. When every member of a software development team is concerned with security, especially if they are specifically trained to, the likelihood that they will either query or collaborate with security OPS before starting the work, preventively and seamlessly, becomes much higher. And as mentioned earlier, physical proximity also supports this method.

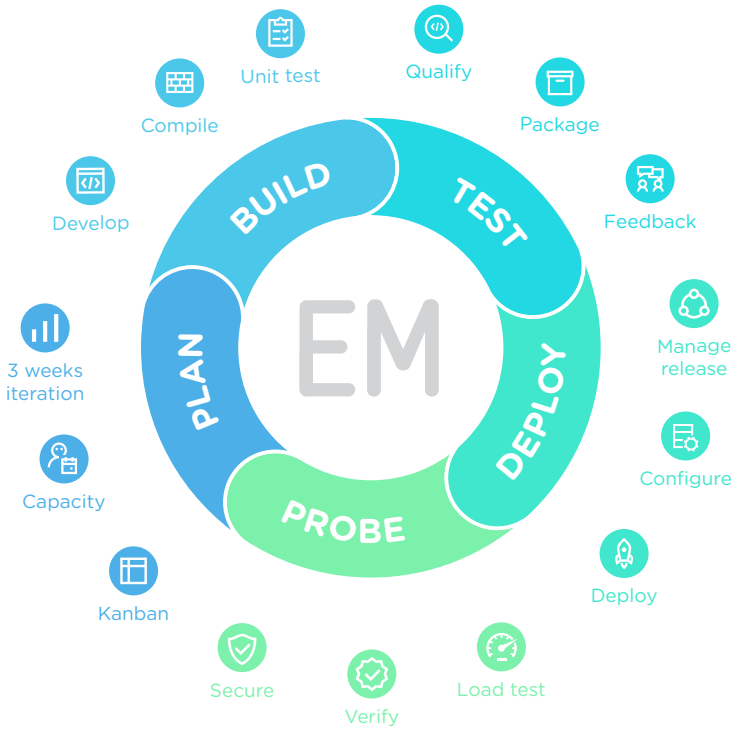
3.2 SecDevOps and Process

For processes, it is a little more tangible to identify what can be done. If we reuse a simple PDCA concept, we can demonstrate how inserting security in an existing process can be done.

Let us make the hypothesis that the figure below represents an example of a typical DevOps process*.



If we are to represent the same process but with SecDevOps considerations it will look like the following.



Lessons are learned from traditional Devops, but there is awareness of security interests. And to include security, you first need to plan for it in resources and team right from the beginning of the project up to the post-deployment stage:

- Include security people in planning
- Train and sensitize the development teams
- Put acceptability tags
- Isolate critical steps of generic accesses
- Qualify deployment
- Include post-production validation

Note: The "Secure" phase represents all adjustments, functional bugs and security/quality assurance issues.

Though this model could be called PDCDCDA, the principle holds. We begin with planning, and for each "do" we have a "check" and an "adjust" just before the beginning of the next cycle.

Consider how these changes are not excessive and, if already framed in a proven process, easy enough to absorb. If we used a checkbox type of workflow, it is only a simple matter of checking more boxes. Conceptually, adding security steps in a process is easy to grasp. Choosing the actual tasks and tools may be a bit more daunting, but in fact we can demonstrate that inserting security relevant tasks is very possible.

For example, we can imagine a detailed procedure of what should be investigated if a problem occurs and how we can ensure that information remains secure (e.g. by changing all the passwords in case of a security leak).

3.3 SecDevOps and Product (tools)

When it comes to producing something, just like in a modern manufacturing chain, the weakest link of the said chain will be the limiting factor to the effectiveness of the whole endeavor. So, the choice of tools is definitively a crucial point to examine.

There's no magic recipe here to guarantee success. But there are indeed some guidelines in the DevOps field to help you along and improve your chances of success. Below is a list of some of the most relevant considerations.

- Make sure the tools fit the need.
- Do not impose tools that are likely to be misused or mistrusted.
- Select tools based on proof of maintainability and reliability of the updates.
- Select tools that have a decent reliability track record.
- Make sure of the knowledge and mastery of the tools, the proper usage and that they are distributed among more than one person in the team.
- Make sure the tools collect and plainly display data that can measure the KPIs that will support your DevOps endeavor.

- Ensure the ease of integration of the different chain loops (i.e.: Planning tools, code source control, build tools, testing tools, deployment tools, etc.)
- Select tools that encourage negligible risk experimentation.
- Select tools that encourage autonomy.

If we agree that these points are worth considering, let's consider how SecDevOps will change the tools already used.

- We are already gathering data and telemetry for continuous improvement KPIs, we will leverage this to also alert of Security issues.
- We are already using planning tools to manage work, we will include artifacts to ensure security concerns are addressed.
- We have check points determined by our processes, where the tools must pass/fail, we will leverage this to address security issues.
- We have a multi stage toolchain that allows to integrate package/content validation. In these validations, we can insert inspection that can qualify or reject the content based on security criteria the same way other criteria are considered.
- In the case of test case management tools, simply add security assurance as a new criterion.

One example would be to have tools that automate security audits or that will automatically detect security flaws on the computer where the product is being installed.

The list could go on and on, but we must keep in mind that incremental, baby steps is the favored way of implementing any DevOps initiative. Making the changes to the toolchain in the most effective, small, low risk way, is in itself the essence of DevOps.

Conclusion

DevOps is change. It is continuous improvement. It is a gain in efficiency. It is a gain in visibility and better collaboration.

SecDevOps is not supposed to change this, it is supposed to leverage this. In all the axes that are important to a DevOps initiative, inserting security imperatives should maintain and keep these values. Considering how DevOps is meant to work, SecDevOps should as well.

Just don't forget to include the concerns from that standpoint in the people, process and product delivery pipelines and your endeavour should be a success.

Works Cited

Aubuchon, Kurt. 2010. How Many Information Security Staff Do We Need? Infosec Island. [Online] September 26, 2010. http://www.infosecisland.com/blogview/8327-How-Many-Information-Security-Staff-Do-We-Need.html.

Baird, Scott. What is a Process? Processmodel. [Online] https://www.processmodel.com/blog/what-is-a-process/.

Brown, Donovan. 2015. What is DevOps. donovanbrown.com. [Online] September 01, 2015. http://donovanbrown.com/post/what-is-devops.

Darwish, Wesam. 2018. Premier Developer. blogs.msdn.microsoft.com. [Online] May 20, 2018. https://blogs.msdn.microsoft.com/msind/2017/03/20/how-devops-are-changing-the-face-of-it/.

Gene Kim, Kevin Behr, George Spafford. 2013. The Phoenix Project. s.l. : IT Revolution Press, 2013.

Matteson, Scott. 2017. Riding the DevOps Revolution. ZDNET. [Online] April 3, 2017. https://www.zdnet.com/topic/riding-the-devops-revolution/.

Sharma, Sanjeev. 2014. Adopting DevOps for continuous innovation. IBM.com. [Online] June 02, 2014. https://www.ibm.com/developerworks/library/d-devops-continuous-innovation/index.html.

Vickrey, Nate. 2017. Best Practices for Internal Communication in DevOps. Devops.com. [Online] May 22, 2017. https://devops.com/best-practices-internal-communication-devops/.