eraneos

Whitepaper

How to start with DevOps A practical guide to transform your organization

June 2023

Content

Introduction

Since the term DevOps was introduced in 2009 by Patrick Dubois and Andrew Shafer, it has become more popular by the year. Many companies are transitioning to a DevOps way of working to beat the competition. Practicing DevOps can lead to more efficient, effective, and innovative software development processes, helping organizations deliver value to customers more quickly and effectively.

"Doing anything and everything to overcome the friction created by silos... All the rest is plain engineering."

- Patrick Dubois

DevOps is about removing silo's and focus on bringing high quality products to customers. Some of the benefits can be:

- Faster time-to-market: DevOps practices can enable teams to release software faster, with more frequent deployments and shorter development cycles.
- Improved collaboration: By breaking down silos between development, operations, and other teams, DevOps promotes better communication, collaboration, and knowledge sharing across the organization.
- Increased agility: DevOps practices can help teams respond more quickly to changes in market conditions or customer needs, allowing organizations to stay competitive.
- Better quality: By integrating testing, monitoring, and other quality assurance practices into the development process, DevOps can help teams deliver higher-quality software that meets customer needs and expectations.
- Increased efficiency: DevOps practices can help teams automate manual processes, reducing the time and effort required to deploy and maintain software.

Many organisations are currently organised around professions instead of products with hierarchical command and control structures. Transforming the company is necessary to fully realize the benefits of DevOps. To start with DevOps, it is advisable to implement an agile way of working first and shift focus from rigid, leading processes to supporting processes. Let us dive into a way to become a DevOps organisation.

In this whitepaper we present some best practices of how to start with the transformation.

The Product and Service Catalogue

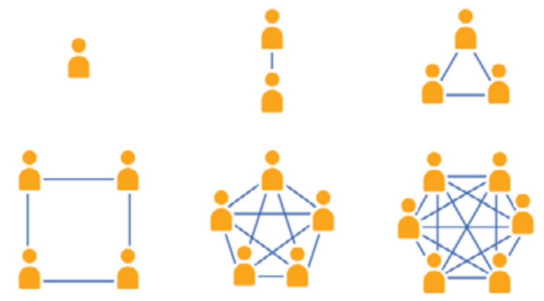
Begin with creating a simple product and service catalogue (PSC). A PSC states the service you sell and the underlying products (a service can have one or multiple products) and nothing more. Those services and products have customers and are the core of your company or department, it is the reason the organisation exists. A PSC is a list of all the services broken up into products. In a DevOps world one team is responsible for a product. A team can have one big product or several smaller ones. So, form teams based on the products you have listed.

The PSC lists only the services and products you sell to external customers, but we need to list the supporting products as well. Supporting products are the stuff that is underneath the products, like servers, network, and certificates. Let us call them platforms. So, products sell to external customers, platforms sell to internal customers (other teams). Form teams around those platforms as well. Now we have product teams (external customer teams) and platform teams supporting the products with a platform.

Setting up the teams

Simple work can be done in big teams, complex work needs smaller teams[1].

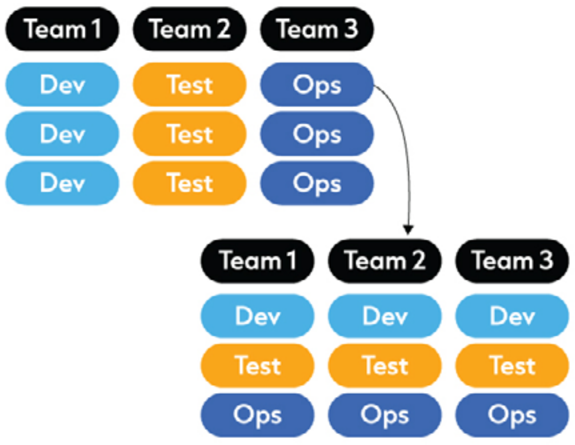
The complexity of work dictates the size of the teams. A small team building complex stuff needs a lot of experimentation and communication, therefor it should be about five to seven people to restrain the amount of communication lines[1]. Most of the time there is a gap between the people needed to do the work and the size of the team. You are free to make bigger teams but keep the complexity in the back of your head and work towards a smaller team, by decompositioning the product or service.



Brooks' law: with each extra member the team adds extra communication lines which can slow down the pace of the team.

Decomposing a product is also particularly useful when you want to move to the cloud, where smaller services are more flexible and scalable.

Within the teams, there is a need to change direction from dev-test-ops silos to an all-inclusive team with all those capabilities. Every team is completely responsible for their product or platform. With responsibility comes autonomy. Teams need to have autonomy on the product they make to make good decisions about the future of the product. To support those teams and give them the autonomy there is also a need to change the way the teams are managed. From traditional, hierarchical command-and-control (think thou shalt, checklists etc) to managing by facilitation. Facilitation is about asking what the teams need, how management can help achieving team goals, in other words what does the team need to achieve their goals. This management style asks for a 180 degree turn in how a company does management[2]. To manage in this style, it is important for management to have a clear vision, mission, and strategic goals. Why does the company sell the services they do, how do they want to achieve that and what needs to be done. Management needs to answer to the question of "why" so to give direction and guidance[3].



Tilting teams: from silo to all-inclusive.

1 Brooks' law.
2 Read Corporate Rebels for some excellent examples of how to manage a company differently
3 Read Why by Simon Sinek to get a better understanding of this concept.

Enabling teams

Once we have a basic setup of a product-oriented organisation, a base for working according to DevOps principles has been created. Some products are delivered directly to the customers, those teams are called product teams (or stream aligned teams). Some products are considered building blocks for other teams, those teams are called platform teams. There is a special type of team for which many product and platform teams are customers. Those teams help the other teams and are called "enabling teams"[5]. Those enabling teams support the product and platform teams with ready to use products. They also give support on those products on how to use them wisely by building reference cases and best practices. Some common enabling teams provide monitoring, dashboarding, build tools, test tools etc. Engineers in enabling teams are broadly developed in using the right tool for the right operation (these engineers are rare to find). They act just like any other product team with the other teams as their stakeholders. But there is a caveat here, the enabling teams should listen to the teams what they need and should not dictate the way of working. The enabling teams are facilitating teams and the success of other teams depends on them to deliver great tools and to make great products.

5 Read about Product, Platform and Enabling teams in Team Topologies.

Skills and capabilities

When the teams are in place there is a need to check if they are up to par on skills and capabilities. This can be done in a conversation with the team about what they are missing, but it is also possible to do an assessment. It is normal that teams do not have all the skills. To develop those skills, it is important to create an environment where they can fail and learn in a safe way.

The next thing to do is adding skills to a team. This can be done in three different ways:

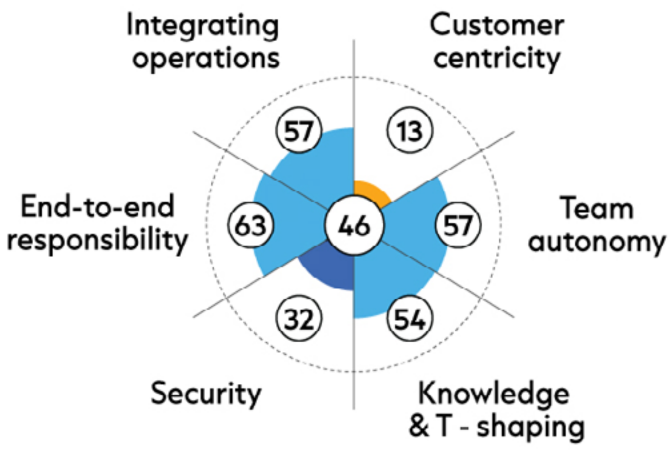
1. Hiring external engineers that are temporarily added to the team to exchange knowledge. Make sure that the external engineers are actively training the team members what they know, so they are not dependent on that engineer for long.

2. The second way is to hire new internal engineers that bring the knowledge as they are added to the team (do focus on sharing knowledge as well; it is a general good practise to share).

3. The third way is to develop capabilities of team members. This can be done by following a course or workshop about the subject or knowledge-sharing within the team to ensure more team members have specific knowledge. Visiting conferences can be a great inspiration as well to find new ways to solve problems. In all situations the minimum time an engineer needs to reserve for learning is about 10% of their total allocated time.

Many engineers are used to feature driven development where the customer comes first, and they lack time allocation for learning. This calls for a change in priority and behaviour because learning has become more important. The world is developing at a fast pace, IT-engineering develops even faster, about twice as fast[6]. Keeping up with this development is important for the engineer and the company.

A high-performance team has profound knowledge about the different skillset of its members. They work cross functional to achieve goals. This means that we cannot talk about job descriptions or professions anymore: we need to focus on skills and individual interests. A great tool for assessing skills and interests is to have a T-shape workshop where every member states their profession (obviously) but also their skills and interest. With that in hand the team should start a conversation about training each other and sharing knowledge. Sharing knowledge can be done very effectively via pair-programming. In the end there will be no developers, testers and operations left, their skillset is mixed, and they help each other with their capabilities and interests to achieve their team goals. We call those people engineers.



Example of assessment results from the Eraneos DevOps Selfie.

6 From the World Economic Forum's Global Information Technology Report: The IT-sector growth in 2020 was 5.9% compared to 3.5% in other sectors.

The role of HR

Teams will develop at different paces; a skilled scrum master can develop a team faster than a starting scrum master. But the scrum master cannot do it alone. Most organisations have an HR department. This department needs to be converted into supporting engineers in their development. The way an HR-department is working can become an internal hurdle and it needs to be changed. HR is a supporting department for the teams that helps with developing the team members. Think about the hiring of new team members and setting out courses (internal or external). Stop doing performance reviews and start finding out what engineers want. A lot of knowledge is obtained with stuff that has nothing to do with software development. But those things can create new insights as well. So, if an engineer wants to follow a course on interior design, let them. They will grow and get new insights which they can convert to new ways of working on the product the team is responsible for (maybe even create a revised and nicer working place, increasing team performance).

It is all about dependencies

One of the biggest hurdles in becoming a DevOps organisation are dependencies. Dependencies are everywhere. They are in culture, processes, organisation, and technology. We can split dependencies up into two forms: hard dependencies where something or someone outside of the team is responsible for, something within the team (compromising autonomy), and soft dependencies where there is a certain slack on the dependency (this is called loosely coupled in software development terms). One of the goals in DevOps is to convert hard dependencies to soft dependencies and to get rid of dependencies where possible (or you can implement SAFe where dependencies are managed with PI-events, completely missing the point of making the dependencies soft).

To start with dependency management fall back to the PSC where products are defined. Start mapping all the dependencies between products, hence we see the dependencies between the teams as well. A great method for this is using Wardley Maps . It gives a perspective from the customer on top to supporting platforms at the bottom and everything in between. An advantage of using these maps is that you can also start strategic planning on improvements to the products and platforms by plotting their evolution from genesis to commodity.

At first most of the dependencies will be hard and cannot move independently. Some examples are signoffs and interconnected applications. The next task is to soften the dependencies and make them loosely coupled. This way a team can move a bit more independently . A loosely coupled dependency has a certain bandwidth to move in. A good example is a platform team providing two versions of the platform via an API (Application Programming Interface). The oldest version is deprecated (it becomes obsolete and will be phased out soon) but it can still be used for a while, and this provides the product team that consumes the platform some time to upgrade to the newer version. Yes, there is still a dependency, but it gained some bandwidth for the consuming team to upgrade.

Reducing dependencies is what makes a team achieve a higher performance. Enabling teams play a key role in this endeavour, but adhering to company rules regarding security, signoffs, and regulations should not be overlooked. Those rules are important for compliance with legal requirements and maintaining a safe and healthy work environment. Frameworks like COBIT and ITIL are valuable but need to be interpreted from a DevOps perspective. Instead of solely being the responsibility of management, the elements outlined in these frameworks should be embraced as shared responsibilities by the team.

Frameworks like COBIT and ITIL should be converted to a team responsibility, not a management responsibility

Wrap up

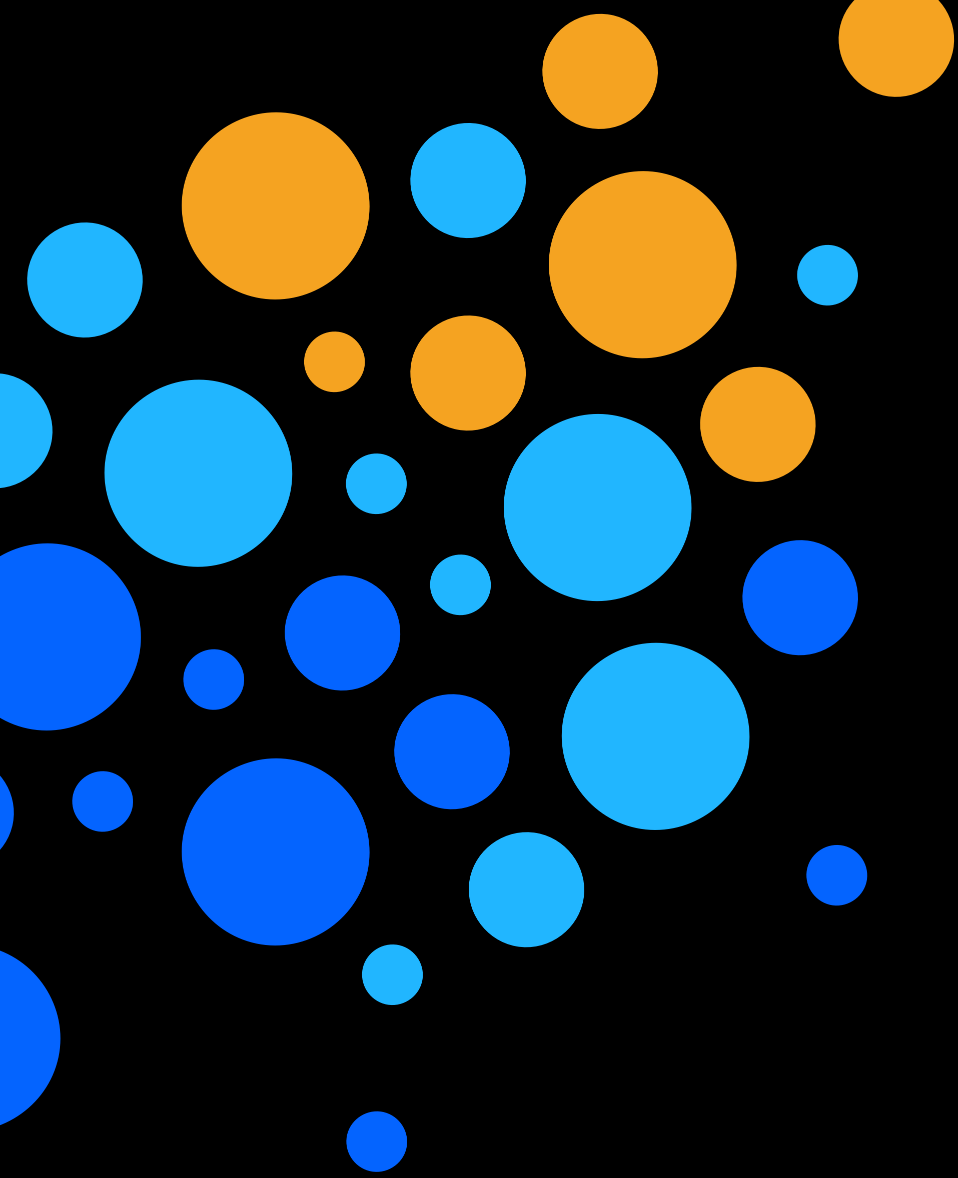
Overall, sticking Dev and Ops people together in a team has more consequences for an organisation than is thought of at first. Setting a goal to become a DevOps organisation has many implications for engineers, managers, and HR. Do not take it too lightly and do not be afraid. Transforming into a DevOps organisation is a great journey and will surely establish a great place to work but it takes a lot of time and effort.

7 https://learnwardleymapping.com

import "./index.css";
import { ReactComponent as ArrowIcon } from "./assets/icons/arrow.svg";
import { ReactComponent as BoltIcon } from "./assets/icons/bolt.svg";
import { ReactComponent as RightArrowIcon } from "./assets/icons/right-arrow.svg";
import React, { useState, useEffect, useRef } from "react";
import { CSSTransition } from "react-transition-group";
"eslintConfig": {
"extends": [
"react-app",
"react-app/"
]
},

Author:

Mick van der Most van Spijk
Consultant DevOps



eraneos

Experienced in a wide range of industries

ABOUT ERANEOS
As a global Management & Technology Consultancy Group, Eraneos supports organizations in not only designing but successfully implementing a future-proof digital transformation strategy that can make an ever-lasting impact.

By listening to what businesses want and understanding their needs, we can fast-track and embed transformation with ease by aligning people with technology, processes and leadership, effortlessly.

Knowing your industry, technology and local context alongside a global perspective, gives us the advantage you need to succeed.

It's this deep understanding that enables us to shape and implement strategic transformation within your organization while providing the best service. That's why our customers trust us with even the most complex of challenges, from strategic digital transformation in finance to the ethical application of A.I. in healthcare.

We don't just listen to your needs, we understand them. We're more than ready to help you realize your potential in the digital age.

Contact us >

Our offices >

Visit our website >