DevOps: understanding the evolution

Many organizations find themselves somehow confronted with the need to change the way their teams are organized and operating. Echoing in the hallways is this idea of DevOps. It sounds promising and has a lot of common ground with Scrum, which organizations are already familiar with. Moreover, it is drenched in Agile thinking. This blog is the first of a few concerning DevOps. In this blog I'll describe an organizational evolution which frequently ''happens" to IT organizations in their journey towards DevOps. Which phase is your organization in and what is there to gain and overcome?

Definitions

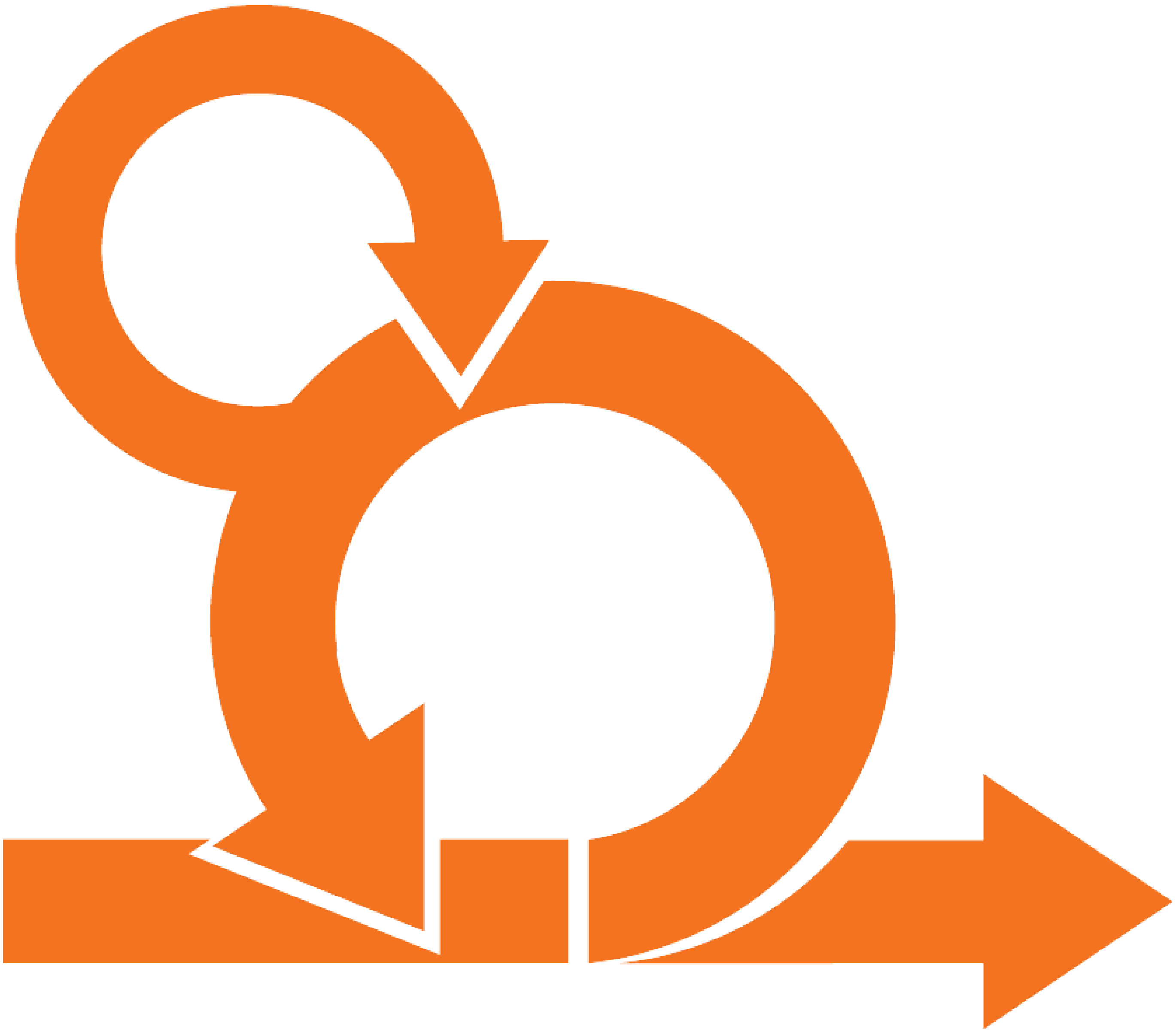
Before jumping into the evolutionary path, let's spend a minute chewing on three DevOps definitions.

DASA: *"DevOps is a cultural and operational model that fosters collaboration to enable high performance IT to achieve business goals."* [1]

Humble: *"a cross-disciplinary community of practice dedicated to the study of building, evolving and operating rapidly-changing resilient systems at scale."* [2]

Len Bass et al. *"a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality"* [3]

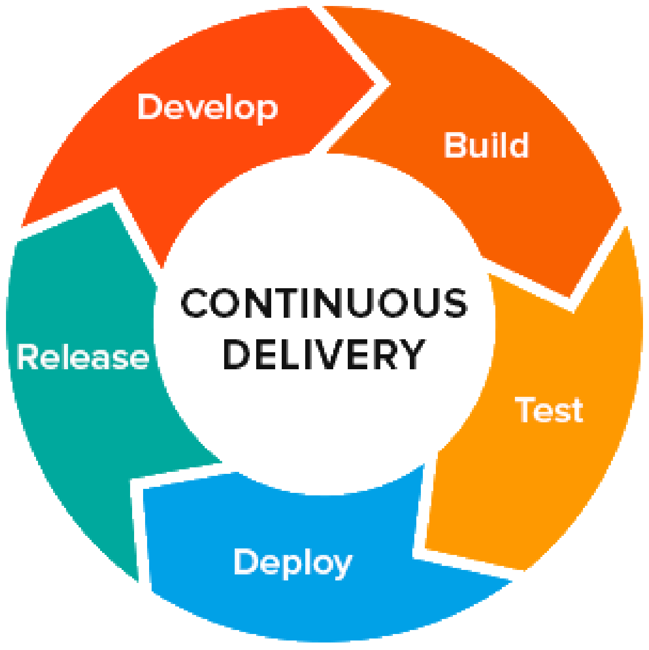
As you can see, the definitions have some overlap but aim their focus differently. The term DevOps has not one clear and uniform definition that is shared in the community. Therefore, it is a good practice to find some common ground before starting any discussion on DevOps in your own organization. For now, it is good enough to have a feeling of what a definition could look like.



1. Pre-DevOps: Scrum teams

Scrum is still the most widely applied framework which is used to organize software development teams. There is no need to differentiate in level of maturity or quality by teams using this framework. In this phase, teams just focus on getting their next Increment of Done software ready and possible even release it. In the team all capabilities to get the software Done should be present, for example designing, programming, testing & even appropriate documenting. However, these are all capabilities and not roles and they are preferably not tight to only one person. All members are simply called developers and there is a chasm between teams and production environment.

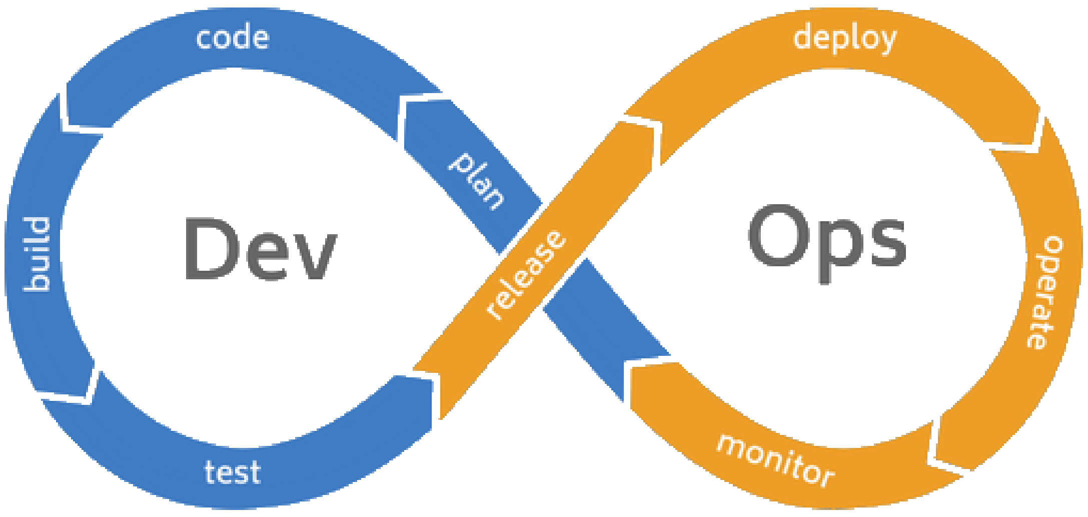
COMMON CHALLENGES: managing external dependencies, including limiting waiting times and hand-overs to other departments for deployments.



2. Pre-DevOps: CD/CI

Since the Continuous Delivery (& Integration) revolution was ignited by Jez and Farley in 2010, a lot more attention is paid to automate and integrate, test execution, deployments on pipelines and feedback loops. The availability of tools that could execute and monitor these activities via – more or less – user friendly interfaces helped getting momentum. In this phase teams are able to deploy much faster and more reliable to different stages. The long-winded deployment manuals become something of a quickly forgotten past, instead the teams are empowered with new tools.

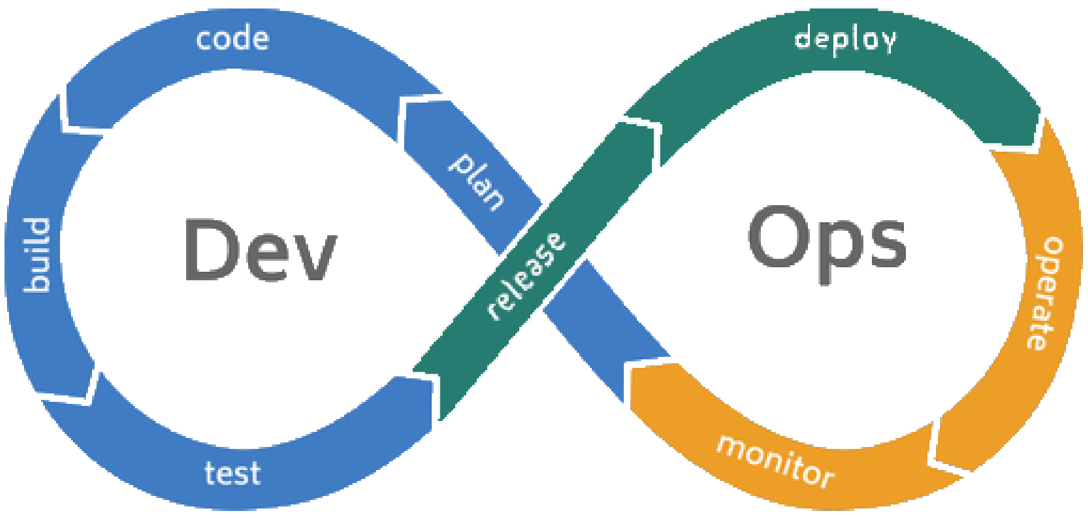
COMMON CHALLENGES: setting up reliable pipelines, changing existing software with CD in mind, depending on the team that supports the pipeline tooling and operator who execute the deployments to production.



3. DevOps round I: Functional Operations

Coming from the previous phases, teams have a rhythm of delivering working software. They should start to live by the mantra “You build it, you run it”. Teams become more interested in the health and status of the production environment. That is an excellent reason for reaching out to Functional Operations to obtain this - more business - side of the software. New questions arise. How is the software used? Which features are the most valuable ones? What is the logging or monitoring of the production environment telling us? The usual approach to enable this organizational need is by extending the existing delivery teams with people who have this Operational role. Often these people come from a more centrally organized department. In doing so, the teams receive more responsibility in the shape of team members who are authorized to see certain data on production and who have in-depth knowledge of how the software is being used. This Functional Ops-role is embodied by an Ops-Engineer who joins the teams.

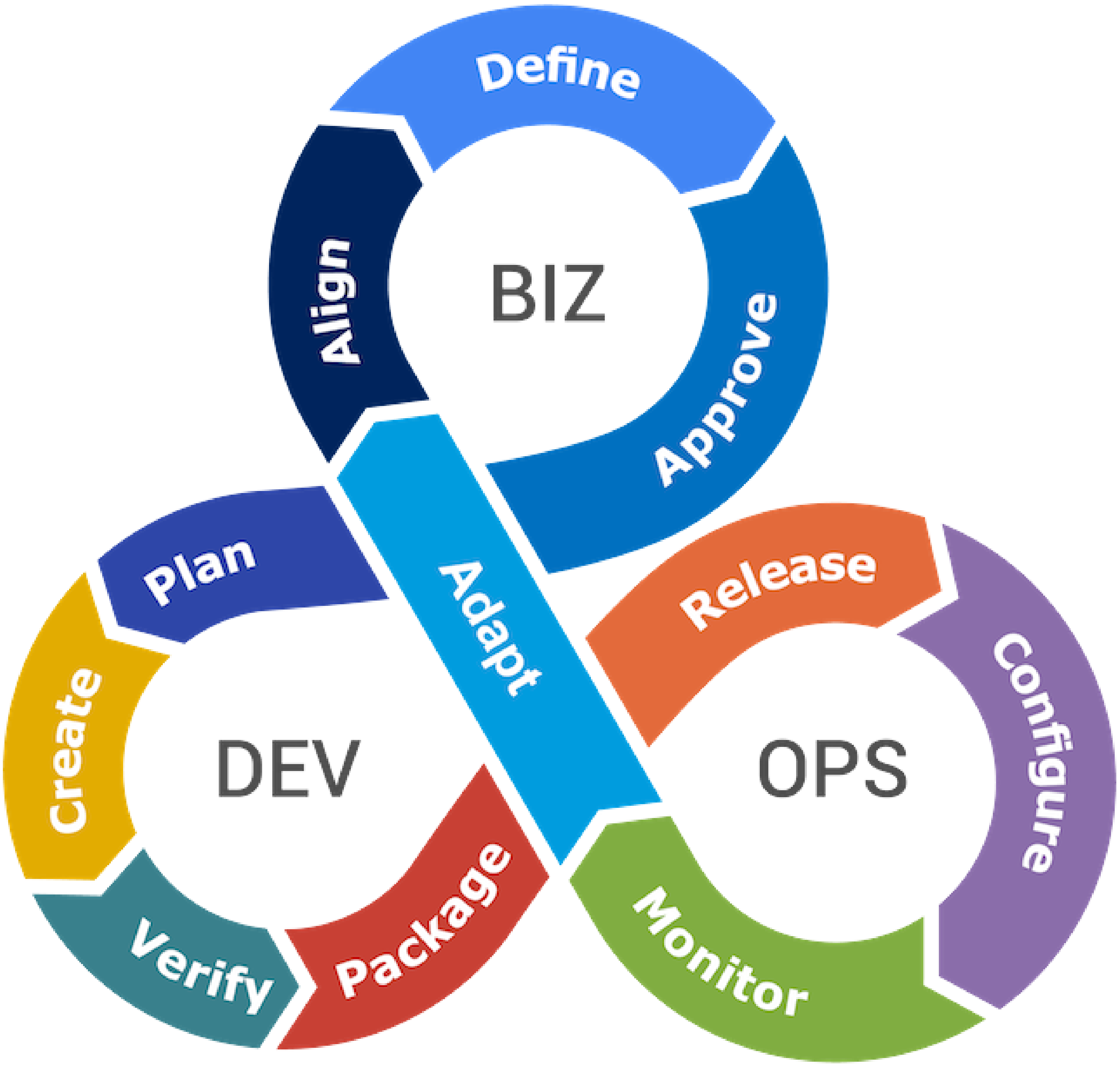
COMMON CHALLENGES: embedding the Functional Operators in the team (and not creating a split team), getting the required rights available for more team members, getting all the information from production to the team e.g. logging and monitoring information.



4. DevOps round II: Technical Operations

Once organizations have more than a few development teams, there are usually also some supporting teams around. The latter teams are usually specialized in the infrastructural layers of the software solutions: development machines, testing environments, disk management and operating systems are all good candidates. If such a central team or department is in place, this team is often responsible for the deployments of new software into production. But, the newly empowered teams in phase 2 or 3 have no such need anymore. More importantly, the teams themselves have the technological means to get their new Increment deployed on any stage or server that is required. The next question usually follows very soon: “Who is now allowed to perform the deployment to the production environment?”. This task had been one of the most eminent examples of the segregation between Dev & Ops, but is in this phase integrated into one.

COMMON CHALLENGES: again, embedding new people in the team, especially when mindset between Dev & Ops are colliding.



5. Post-DevOps: BizPerfSecDevOps?

In order to make teams more and more self-supporting, they need other capabilities, either technical or more business oriented. Depending on the nature of the business and – again – the size of the organization, any of the following experts' roles could be eligible for the team: DB-admin, security expert, platform engineer, performance engineer, UI/Interaction designer, SEO specialist, business expert.

COMMON CHALLENGES: which extra responsibilities are helping teams to create more value versus which capabilities should remain separate?

Conclusion: How to evolve?

From an organization view point any of the phases described above is just one of the almost infinite amounts of possibilities to get organized. Take a step back and see how your organization is just a collection of people, with skills, tools, tasks, responsibilities and mandates organized in a particular way. We aim to come up with the exact organization that is able to create the most value. In the evolution above there is an ongoing shift in roles and responsibilities to and from teams. The context of the team is very much leading in getting the optimum setup. Longstanding organizational setups and existing boundaries are often hard to get by.

One way of thinking more out of the box is by raising the question: “What if we only had two teams?” or even better “What if we only had one?” This forces you to elaborate on who is allowed to do what and why. This question is applicable in all phases, yet becomes paramount in the last one. Last but not least: don’t get stonewalled by the existing organizational structures and boundaries.

Notes:
1. The aforementioned common challenges are by no means complete.
2. Furthermore, being in a next phase does not exempt you from keeping attention to practices of previous phases (e.g. Scrum).

In the next Blog I will focus on the above used terminology in depth.

References & recommended reading:

[1] DASA: DevOps Fundamentals_Glossary_English
[2] https://theagileadmin.com/what-is-devops/
[3] Bass, Len; Weber, Ingo; Zhu, Liming (2015). DevOps: A Software Architect's Perspective.
[4] Scrum a Pocked Guide, Gunther Verheyen
[5] Continuous Delivery, Jez Humble, David Farley
[6] The DevOps Handbook, Jez Humble, Patrick Debois a.o
[7] DevOps for Developers, Michael Hüttermann
[8] The Phoenix project, Gene Kim, Kevin Behr a.o.