The CAMS model to better understand the DevOps movement

Bruno Delb

Jul 22, 2018 · 5 min read

Today we are going to talk about DevOps. What is DevOps concretely? The definition of DevOps is rather clear, but what is important is to understand what it means concretely. And the CAMS model answers this question very clearly and precisely.

. . .

DevOps is mainly a culture, as highlighted by the CAMS model. The CAMS model was invented by Damon Edwards and John Willis, authors of the famous Podcast DevOps Cafe. CAMS stands for Culture, Automation, Measurement and Sharing.

Let’s start with the first part, Culture.

C for Culture

Culture is the most important part of the DevOps movement. But what are we talking about here? Culture will bring a whole set of practices:

- the use of Scrum,
- the presence of silos,
- the willness to limit the level of technical debt,
- the organization of retrospectives, …

However, be careful : don’t try to replicate practices of other organizations as they are, only for the reason that it works in these organizations. This is called the Cargo Culting effect. You must first examine the problems that these other organizations have tried to solve and see how they solve them. Only after you can determine what

A for Automation

Automation meets many needs and solves many problems of course. But beware, they only account for 25% of DevOps work!

So why automate? Above all to speed up the flow of information and avoid recruiting people.

Speeding up the flow of information, of course, brings the software to the client sooner, but it also helps to get results earlier. And so if a problem occurs earlier (such as the detection of a bug), it costs much less than if it is discovered later.

Avoid recruiting people, why? Of course for financial reasons, but also to avoid potential problems of communication between humans, to avoid the conflicts of people and to avoid mistakes made by humans. And yes, because the more humans there are, the more mistakes there are …

The investment is heavy. The return on investment will be on time.

When we talk about automation, we think about the notions of:

- “infrastructure as code” (for this we can use tools like Ansible or Chef),
- continuous delivery pipelines (here we are talking about Jenkins for example).

The concept of “infrastructure as code” makes it possible to automate the tests on the infrastructure, to ensure that the infrastructure is continuously tested. It also allows to set up pre-production environments identical to production environments.

Continuous delivery pipelines are in fact in two parts: Continuous Integration and Continuous Delivery.

M for Measurement

The continuous improvement process (Kaizen) is at the heart of DevOps. But for that, it is still necessary to be able to know if one has improved and to be able to prove it. No miracle, measures are the key, because decisions must be made on facts, data, not opinions.

So what can we measure? Well we could measure well the response time of the system

performance or on the contrary has degraded it.

Therefore, on the developer side, it is very important that they provide on-board monitoring services in the provided applications.

From these measures, KPIs (Key Performance Indicators) can be established to answer important questions, such as:

- How many users have registered today?
- What are the incomes today?
- What are the operating costs?
- What is the number of tickets opened to the Call Center today?
- Etc …

There are many instrumentation platforms to produce graphics (Graphite, Grafana, …).

It is also good practice to set up information radiators in the organization's offices, for example on a big screen. These information radiators must show very clearly what the normal looks like, so that you can more easily notice the strange behaviors.

S for Sharing

The last component of CAMS is Sharing. Sharing has three components:

- the visibility,
- the transparency,
- the transfer of knowledge.

Let's talk about visibility first. Visibility is what allows everyone to see the progress of other parts of the organization. So that's what you did.

Concretely, visibility makes it possible to know if the work of a team can introduce a problem to another team. And it also allows for early feedback, which helps to ensure that failures occur early rather than late.

Let’s move on to transparency. Transparency is what allows everyone to work towards a common goal. So that’s why we did what we did.

In practice, a lack of transparency can lead to misalignment between teams and lead to inappropriate developments, for example.

Finally, the transfer of knowledge. The transfer of knowledge aims to:

- avoid constraints in the organization,
- promote the collective intelligence.

What does it mean to avoid constraints? To better understand, let’s take an example. Often, in organizations, only one person has some know-how. And when this person goes on vacation, the whole team gets stuck. This is called a constraint.

To avoid this, the solution is to share knowledge between people. Let’s take a concrete example: the deployment of an application. If only one person knows how to deploy an application, then when this person is away, the team will no longer be able to deploy the application.

And what does collective intelligence mean? Collective intelligence materializes the fact that we are more intelligent collectively than individually. Why ? Simply because it is possible to retain the best ideas to solve problems.

There is a lot of way to put knowledge sharing in place. If you are in Agile, then you already share knowledge through Daily Stand-up and retrospectives. On the development side, documentation in general and well-documented code are ways to share knowledge. The ChatOps, which will be the subject of an article later, makes it possible to let everyone know the result of the work done. Events such as internal conferences, technical presentations or less formal discussions during lunches also help to promote the sharing of knowledge.

. . .

In the light of this model, we become more aware that the pure automation part of the build chain and delivery are only a small part of DevOps. DevOps is not just tools, it’s a lot more, it’s a culture, a state of mind, using the same concepts than Agility, as Agility uses Lean principles. In other words, DevOps relies heavily on Lean.

> *More articles on my blog http://www.agilelabtest.com.*
>
> *My LinkedIn profile: https://fr.linkedin.com/in/brunodelb*

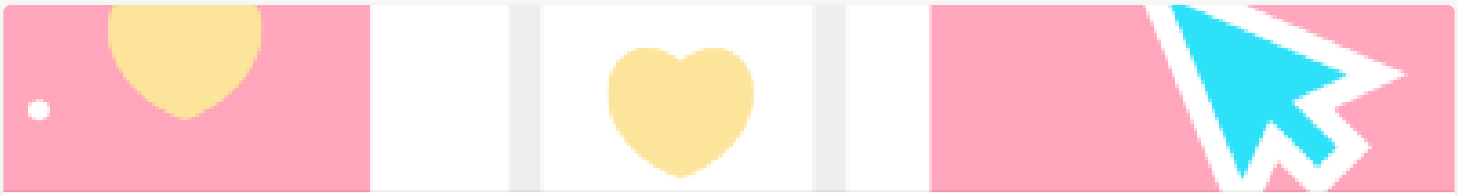
DevOps Cams

4 claps

Bruno Delb

Agile Coach working in the Medical Device Software domain, DevOps and former Web & mobile app developer

Follow



Related reads

How to get a Job in Tech Without a Degree

Eleanor McKenna
7 min read

79

Related reads

Vagrant Provisioning

Joaquin Menchaca
6 min read

37

Also tagged DevOps

Why you should care about Docker?

Chinmay Shah
6 min read

571

Responses

Write a response...