

(<https://jaxenter.com/>)
 Angular v6: The wait is over! Angular v6 is here (<https://jaxenter.com/road-to-angular-6-139479.html>)

SEARCH

A brighter DevOps future

The 4 roles of DevOps leadership

🕒 June 22, 2017 👤 Anton Weiss

#devops (<https://jaxenter.com/tag/devops>)

reddit Twitter LinkedIn Facebook G+ Google+

(http://www.reddit.com/r/devops/comments/134912/the_4_roles_of_devops_leadership/) (<https://www.linkedin.com/pulse/the-4-roles-of-devops-leadership-anton-weiss/>) (https://www.facebook.com/slagelcream/share?_u=https://jaxenter.com/4-roles-devops-leadership-134912.html) (<https://www.instagram.com/p/134912/>)

There's no magic wand to make a company become better at DevOps. Leadership from management is necessary. In this article, Anton Weiss goes over the history of DevOps, what makes a good leader, and how managers can lead their teams towards a better and brighter DevOps future.



© Shutterstock / SewCream

Several years ago when we started having DevOps conferences, there were a lot of talks about how to get management buy-in to our new wonderful paradigm.

Engineers from operations and development would come on stage and tell stories of wins and losses in their struggle to demonstrate that this DevOps thing is valuable for the business.

Fast forward to here and now and we suddenly realize the situation is upside down. What we as DevOps consultants now see is that it is managers who come to us and say – we know we want to do DevOps – how do we get engineers on board? Things change fast, don't they?

Devops was born as a grassroots movement. Like any true revolution, it was a fire that started from an ever growing haystack of despair over pains and inefficiencies of the old ways of working and a spark of creativity. The spark that led the brighter minds among us to rethink our modus operandi and realize there's a better way.



(https://jaxlondon.com?utm_medium=banner&utm_source=jaxenter.com&utm_campaign=sands_om&utm_content=article_ad)

Meet us in London: The Conference for Java & Software Innovation (https://jaxlondon.com/?utm_medium=banner&utm_source=jaxenter.com&utm_campaign=sands_om&utm_content=article_ad)



ECLIPSE MICROPROFILE: ACCELERATING JAVA MICROSERVICES

(<https://jaxlondon.com/microservices/eclipse-microprofile-accelerating-the-adoption-of-java-microservices/>)

utm_medium=banner&utm_source=jaxenter.com&utm_campaign=sands_om&utm_content=article_ad)

Emily Jiang (IBM)



JAVA SE 9 MODULES: AN INTRODUCTION (<https://jaxlondon.com/java-core-languages/java-se-9-modules-an-introduction/>)

utm_medium=banner&utm_source=jaxenter.com&utm_campaign=sands_om&utm_content=article_ad)

Stephen Colebourne (OpenGamma)

DevOps and Agile – the two revolutions

<https://jaxenter.com/> DevOps, the question of its relationship with Agile development methodology comes up often. Some say DevOps is a continuation of Agile, or an extension of it. “Agile didn’t take the operations into account” – they say – “that’s why DevOps was needed.” And if we think about it – Agile in itself was a revolution – with a manifesto and noble ideas of collaboration, communication and humanity inscribed on its flag.

But – as it happens with all revolutionary ideas – Agile got institutionalized, formalized and translated into today widely known methodologies such as Scrum, LESS, SAFE and their various mutations. And that was a blessing to all the new adepts. All those managers saw the potential of Agile practices to make their organization more efficient, responsive and fast. They now had a framework that allowed one to have Agile in “a few easy steps”.

However, it was a mixed blessing. First of all, reality wasn’t all that easy. When building human organizations there’s never a one-size-fits-all recipe. And second, as happens with any institutionalization, in many cases this led to a purely formalistic approach. Quite adversely, processes were valued over people and their interactions. Backlog grooming became more important than working software.

Quite a number of engineers I’ve talked to in the last 10 years expressed their sincere disbelief in Agile practices, saying it’s a meaningless game. Or worse: a manipulation mechanism meant to make engineers helpless and bend their will and professional opinion, all in favor of achieving short term financial goals.

SEE MORE: “We want to do DevOps, how do we get engineers on board?” (<https://jaxenter.com/interview-anton-weiss-devops-133745.html>)

So when you think about it, the DevOps bonfire rose so quick and high because Agile was losing its edge. Because it wasn’t serving its purpose anymore. Because Agile failed to make engineers lives any better. Unhappy engineers write crappy code and stop caring about the systems they manage. As a result, customers suffer, business suffers, and everybody loses.

Now, DevOps may be seen as an extension of Agile, but it’s not the same. For one, there is no framework. Yes, there are some methodologies and tools: continuous delivery, infrastructure as code, containers, integrated monitoring. There are architectural patterns like microservices, feature toggles, loose coupling. But these are only the tools.

If you’ve done some reading on DevOps you probably heard of CALMS. It stands for

Culture
Automation
Lean practices
Measurement and
Sharing.

CALMS can be seen as the cornerstone of DevOps. The acronym was originally coined by John Willis and Damon Edwards and then further developed by Jez Humble. So if we now go back to the tools and methodologies, we’ll realize they only contribute to automation, measurement and some of the leanness. They don’t give us neither culture nor sharing.

Talking about acronyms, they only give us ALM. In a way it is interesting as this is a totally different acronym that stands for Application Lifecycle Management. But if we only implement the ALM. Otherwise, we risk falling into the same trap of formalistic implementation as happened with many Agile initiatives.

Escaping the trap

Ok, so there’s a trap here!

Now, let’s say you are a manager who wants to transform her organization to work in a DevOps way. You know there is no framework . You know there are some guiding principles. And you know there’s a trap. (You’ve been there or you’ve learned from your predecessors.) And well, you know the way to escape the trap is to make sure you take care of culture and sharing, of communication and collaboration, of compassion and creativity. But how the hell do you go about this?

Can we just tell our devs, testers and ops to cooperate, have empathy, be innovative? God, it would be so great if you could just tell people to cooperate and they would listen.

Unfortunately, we all know that it’s more complicated than this. Cooperation and sharing aren’t based on mechanical actions; they are driven by emotion and emotions aren’t activated on instruction.

SEE MORE: “At GitHub, we collaborate far more than just between Devs and Ops” (<https://jaxenter.com/devops-interview-github-133749.html>)

The question becomes – how do we influence emotions? How do we inspire and motivate? Can we maybe pay our engineers more for cooperation, or fire them for not being compassionate enough? Should we pay bonuses to ops who show more empathy or devs who do more experiments? How the hell do we enable the culture of sharing and experimentation – the third way of DevOps according to Gene Kim?

The answer is – you guessed it – leadership.

(<https://jaxenter.com/>) is a trick answer. So you're asking me how to go about something as intangible and hard to define as culture and I'm just giving you leadership – another hard to define concept as a solution. As David Wheeler said, “All problems in computer science can be solved by another level of indirection... except of course for the problem of too many indirections.”

I definitely believe that in the case of large organizational transformation (which DevOps is), leadership is our vehicle of building culture. And you know what, leadership may actually be easier than it seems. Social scientists have been studying leadership for years!

In fact we can even outline all the major leadership theories in less than 5 minutes. And as Kurt Lewin, the renowned psychologist and researcher, said, “There's nothing so practical as a good theory.” So bear with me as we're going to run through all theories quickly and then see how it all connects to transforming our IT organizations in the end.

A quick rundown of leadership theories

The first attempts at studying and defining what allows some people to lead and makes others follow them were made a couple of centuries ago. It all started with what was originally called “**The Great Man Theory**”. It basically said that great leaders are born rather than made. You either have it or you don't. You cannot become Alexander of Macedon. Some of us may agree, but this theory was so impractical that the researchers (or rather philosophers) then tried to outline the set of attributes that all of these natural born leaders have in common.

Which gave birth to the second theory: “**The Trait Theory of Leadership**”. This was a line of research that examines which individual characteristics we should pursue in order to lead effectively. Already, this theory is more down-to-earth: be like this and that and people will follow you. The downside was that the scientists identified dozens of traits. No single set has emerged as the ideal for all circumstances. This was already substantially better, but still not very applicable. Can you even develop character traits?

So, the search for something more susceptible to real-life implementation lead to “**The Skills Theory of Leadership**”. Just like trait theory, it tries to identify a set of key attributes. In this case, it identifies practical skills rather than just general qualities. The bottom line on this one is that if you want people to follow you, you need technical skills in your field, so you have some credibility; people skills, like persuasion, diplomacy, and affability; and conceptual skills, like the ability to see the big picture and to think strategically. This is already something we can work with! But it doesn't stop there.

SEE MORE: An insight into the corporate culture at SoundCloud (<https://jaxenter.com/devops-culture-soundcloud-interview-133716.html>)

Next up there is the “**Leadership style theory**”. This states that your style of leading is the key to success. It talks about different styles like “be autocratic and demanding” or “be democratic and participative” or “be *laissez faire* and leave people alone”.

Probably the best known style-based theory is called “**The Managerial Grid**”. This says people should “adopt a leadership style that's both people-friendly and uncompromising on performance”. Easier said than done, huh? It's a solid foundation, but there's a bit more to leadership effectiveness. That's where these next couple theories came from.

The “**Situational Leadership**” theory argues that there is no “one-size-fits all” model. Certain traits and skills and styles fit better in one situation than another, so the leader must adapt. If we take it to our field, this would mean that implementing DevOps in a fast-growing startup requires a different approach than doing it at a large enterprise dinosaur. Same objectives and standards, perhaps, but to get great results might require more of a disciplinarian for the crazy startup bunch, or a highly inspirational approach for the disillusioned and burnt-out corporate IT crowd.

A closely connected idea is called “**The Contingency Theory of Leadership**.” Whereas the situational leadership approach assumes that the situation is static and leaders should adapt to it, the contingency theory assumes that the leader's default style is also pretty much fixed. Maybe the leader is much more task-oriented than people-oriented. So, the trick is to fit the right leader to the situation. Bottom line: effective leadership is contingent on matching the leader's style to the setting. Again – taking it to the IT field this would mean that someone who's successfully lead DevOps implementation at a unicorn isn't necessarily a good fit for doing this kind of transformation in enterprise settings.

SEE MORE: “In 10 years, DevOps will probably experience a midlife crisis” (<https://jaxenter.com/devops-interview-simon-wardley-134886.html>)

The next two theories are commonly seen as opposites. As the term implies, “**Transactional Leadership**” is based on the principle of reciprocity. People will follow based on the incentives in place, so the leader's job is to find the right mix of rewards and punishments and then closely monitor what's going on.

By contrast, the theory of “**Transformational Leadership**” says that leaders gain buy-in and commitment from encouraging their followers and not a *quid pro quo* approach. Caring for them and inspiring them toward a vision increases buy-in. In short, they get results by proactively transforming the environment and the relationships. They aim to cultivate followers rather than pay for it or punish non-compliance like a transactional leader would.

(<https://jxenter.com/>) **Servant Leadership Theory**, which is kind of a blend between transformational and transactional leadership. Boiled down to its essentials, it says that if a leader makes a priority of identifying and meeting followers' needs – serving rather than being served – that leader creates an environment of trust and cooperation and reciprocal service, and ultimately higher performance. It's been popularized in recent decades by many researchers, but it goes back a lot further than that. Much of Jesus's influence, for example, was and still is a result of compassion and service and sacrifice. People follow out of love and gratitude rather than out of compulsion or fear.

The 4 roles of DevOps leadership

All right, that's ten of the major theories in leadership and there are important truths in each. Now the important question here is how to apply these in the context of DevOps. After all we are DevOps practitioners, not DevOps theorists.

So, in order to provide some practical advice we've built the following short list called "The 4 Roles of DevOps Leadership".

Role 1 – Tell the Story

Looking at our theories list, I certainly believe the most important lesson comes from the situational leadership theory and its offspring. We have to start with evaluating our situation. For a greenfield project involving a small cross-functional team, a skill-based leadership always works best. Small scale and relatively low complexity will allow the skilled leader to be in control of the technical implementation, the human relationships and the strategic vision. But greenfield projects aren't the real challenge.

The actual benefit of DevOps is most obvious in large complex environments. Environments that weren't originally built with DevOps principles in mind. Environments in need of transformation. And for transformation we need – you guessed it – transformational leadership.

We need reformers who can explain the why behind DevOps, who can create the vision of a better world, who can inspire and motivate their followers. You need to be a culture builder. Culture is easier than it seems. Culture is just a story you tell. But in order for it to work, it has to be your authentic story. It has to be believable and the leadership must be a part of it just like everyone else.

So, this is our role number 1: tell the story! If you're a manager looking at implementing DevOps you need to either become the DevOps storyteller yourself or find someone inside or outside your organization who'll become that storyteller.

Role 2 – Be the safety guard!

Now, once you got your people to believe in your story, you need to provide them with the safety to experiment. The road to improvement isn't a straight path. The main reason we're so reluctant to take a step is the fear of stumbling. We do need a leader who'll be there to support us if we fall. This is the servant leader, the one ready to make sacrifices for their people and not to sacrifice them.

In the last year, we had a couple of clients who came to us asking, "how many people can I fire if I implement DevOps?" I always tell them, "If you need to fire them, fire them now. Because you can't build collaboration and innovation under the threat of a headcount reduction." On the other hand, while the scarcity of resources is challenging, it can also provide that much more of a boost to an engineer's creativity and imagination.

Role 3 – Build the Kernel team

But while transformational leadership and servant leadership are both a necessity, they are not sufficient. After all, we're talking about producing value here and for value to be generated — work has to be done. In DevOps, this work occurs in independent, self-organizing teams. We envision these teams as being as agile and lean as the best startup out there. Otherwise, that startup will come and win; first, your developers and then your business.

So, we still need the skill-based leaders at this level to coach, lead by example and coordinate the activities of their teams with the big picture. Note the importance of coordination; DevOps is about enabling end-to-end flow and the flow always gets stuck at the connections.

Our role number 3 will be focused around a group of skilled technical leaders to support the transformation. They will be backed and incentivized to cooperate by the transformational leader. In exchange, they will escalate the big picture of the direction organization is and should be moving in. I like to call them the Kernel team. However, the concept is otherwise known in management theory as a Synerteam, or a task force purposed with analyzing bottlenecks and enabling seamless cooperation of all business units.

Role 4 – Be the communication enabler

And for role number 4, I'd like to talk about Martin Luther (https://en.wikipedia.org/wiki/Martin_Luther). Luther wasn't the first reformer taking the Catholic church to task for their wrong-doings. But he definitely became the most significant figure in Protestant Reformation and the reason for that was mass communication. Luther's most seminal work were the Ninety-Five Theses (https://en.wikipedia.org/wiki/Ninety-five_Theses), in which he clearly outlined the abusive practices of Catholic clergy. He argued that to be a good Christian one doesn't need to buy indulgences. To believe in God is sufficient. Again, Luther wasn't the first to think about that. But he

(<https://jaxenter.com/>) was not taken advantage of the new technology, Gutenberg’s printing press (<https://www.youtube.com/watch?v=DLctAw4JZXE>). His ideas were printed and distributed as a pamphlet, which reached all over Europe. His ideas contributed to the establishment of Lutheranism and had not only theological but also social consequences.

In order for change to occur, it has to be communicated. We’ve all recently heard of the impact the social networks had on the Arab Spring. If you want spring to occur in your organization, you have to enable open communication in the form that feels the most natural to the modern engineers. Be it Slack, Yammer, Riot, Confluence or Bitrix. GitLab teams call this Conversational Development. The idea of ChatOps popularized by Github is also a favorite.

As Marshall McLuhan famously said, “the medium becomes the message”. If you want to bring the message of change, change the channels of communication. And don’t wait for this to occur spontaneously. In the same way the government is responsible for maintaining roads and electrical lines, organizational leadership is responsible for establishing, promoting and maintaining corporate communication channels. Make the work seen and you’ll see it getting done. Connect with the World and enjoy the Ride.

Conclusion

The ultimate goal of DevOps transformation is arriving at the self-organizing system in which teams collaborate effectively simply because that’s how things are done, simply because they trust, understand and respect each other. This is a beautiful vision. As beautiful as a large codebase with zero bugs.

However, the second law of thermodynamics clearly shows us that entropy or disorder will always increase in an isolated system. A system can not increase its order without an external relationship. That means that as your DevOps becomes reality, the role of leadership will have to change. It must change from being the forming and driving power to becoming the channel connecting your organization with the outside world in order to prevent entropy and allow change.

Good luck on your DevOps journey – and trust me – the further you’re down the road, the more you will enjoy what you’re doing. Because if we’re not doing DevOps to make our lives more enjoyable – then what the hell is it good for?

Be the first to share this article with your network!

 reddit

 Twitter

 LinkedIn

 Facebook

 Google+

(<http://www.reddit.com/submission?url=https://jaxenter.com/4-roles-devops-leadership-134912.html>)

(<https://twitter.com/status?url=https://jaxenter.com/4-roles-devops-leadership-134912.html>)

(<https://www.linkedin.com/sharing/share?mini=true&url=https://jaxenter.com/4-roles-devops-leadership-134912.html>)

(<https://www.facebook.com/sharer.php?u=https://jaxenter.com/4-roles-devops-leadership-134912.html>)

(<https://plus.google.com/share?url=https://jaxenter.com/4-roles-devops-leadership-134912.html>)

roles-devops-leadership-134912.html

roles-devops-leadership-134912.html

roles-devops-leadership-134912.html

roles-devops-leadership-134912.html

roles-devops-leadership-134912.html



Anton Weiss is Principal Consultant and CEO at Otomato – the effective software delivery company.

Recommended For You

Leave a Reply

Be the First to Comment!



Start the discussion

Subscribe ▾

[\(https://jaxenter.com/\)](https://jaxenter.com/)

SEARCH

**INTRODUCTION TO HASHICORP VAULT [SOLD OUT]**Nic Jackson (*HashiCorp*)**KUBERNETES SECURITY: FROM IMAGE HYGIENE TO NETWORK POLICIES**Michael Hausenblas (*Red Hat*)**HANDS-ON: LINUXKIT UND K8S**Erkan Yanar (*linsenraum.de*)**WORKING UP THE HIERARCHY OF SERVICE RELIABILITY**Björn Rabenstein (*SoundCloud Ltd.*)**THE SCIENCE OF CULTURE CHANGE**Morgan Martins (*Institute of Physics*)**Zum Programm****PDF MAGAZINE****Machine learning in practice:** Rules, steps, and success stories

free download

<http://jaxenter.com/jax-magazine>**FEATURED POSTS****On the road to Angular v6: Pop the champagne, it's here!**
(<https://jaxenter.com/road-to-angular-6-139479.html>)**What's new in Angular 6?**
(<https://jaxenter.com/new-angular6-143995.html>)**13 reasons why we all should be excited about Angular v6**
(<https://jaxenter.com/angular-6-feels-144154.html>)**Java SE release cadence FAQ: Benefits, migration, LTS & more**
(<https://jaxenter.com/java-se-release-cadence-faq-144094.html>)

(https://jaxenter.com/)



(http://onelink.to/dsjhrj)

SEARCH

DEVOPSCON 2018 SESSIONS



INTRODUCTION TO HASHICORP VAULT [SOLD OUT]

Nic Jackson (*HashiCorp*)

KUBERNETES SECURITY: FROM IMAGE HYGIENE TO NETWORK POLICIES

Michael Hausenblas (*Red Hat*)

HANDS-ON: LINUXKIT UND K8S

Erkan Yanar (*linsenraum.de*)

WORKING UP THE HIERARCHY OF SERVICE RELIABILITY

Björn Rabenstein (*SoundCloud Ltd.*)

THE SCIENCE OF CULTURE CHANGE

Morgan Martins (*Institute of Physics*)

Zum Programm

Tweets by @JAXenterCOM

**JAXenter.com**

@JAXenterCOM

Do you have a topic proposal about #blockchain that's burning away in your mind? Looking for a receptive audience to explain some of the finer points about blockchain technology to? The call for papers is open. buff.ly/2FGOfaM

Embed

View on Twitter

FINANCIAL IT STORIES





Welcome to the future:

"Blockchain and AI will collide for the first time with SingularityNET"


([https://jaxenter.com/blockchain-](https://jaxenter.com/blockchain-ai-singularitynet-139701.html)

<https://jaxenter.com/blockchain-ai-singularitynet-139701.html>)


(<https://jaxenter.com/62-insane-facts-about-bitcoin-the-infographic-https://jaxenter.com/bitcoin-62-facts-138752.html>)

<https://jaxenter.com/bitcoin-62-facts-138752.html>


 [Bit]coin flipping: Bitcoin Core 0.13.0 — what does it mean?
<https://jaxenter.com/bitcoin-flipping-bitcoin-core-0-13-0-what-does-it-mean-128691.html>
<https://jaxenter.com/bitcoin-flipping-bitcoin-core-0-13-0-what-does-it-mean-128691.html>


 Bitcoin named by banks as major threat, MasterCard agrees
<https://jaxenter.com/bitcoin-named-by-banks-as-major-threat-mastercard-agrees-116010.html>
<https://jaxenter.com/bitcoin-named-by-banks-as-major-threat-mastercard-agrees-116010.html>


 IBM bets big on blockchain, contributes 44,000 lines of code to Hyperledger
<https://jaxenter.com/ibm-contributes-44000-lines-of-code-to-open-source-blockchain-project-124073.html>
<https://jaxenter.com/ibm-contributes-44000-lines-of-code-to-open-source-blockchain-project-124073.html>

TRENDING POSTS

 On the road to Angular v6: Pop the champagne, it's...
<https://jaxenter.com/road-to-angular-6-139479.html>
<https://jaxenter.com/road-to-angular-6-139479.html>

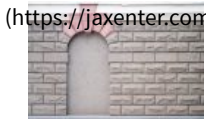
 How to implement a switch-case statement in Python
<https://jaxenter.com/implement-switch-case-statement-python-138315.html>
<https://jaxenter.com/implement-switch-case-statement-python-138315.html>

 10 SQL tricks that you didn't think were possible
<https://jaxenter.com/10-sql-tricks-that-you-didnt-think-were-possible-125934.html>
<https://jaxenter.com/10-sql-tricks-that-you-didnt-think-were-possible-125934.html>

 Spring Boot tutorial: REST services and microservices
<https://jaxenter.com/spring-boot-tutorial-rest-services-and-microservices-135148.html>
<https://jaxenter.com/spring-boot-tutorial-rest-services-and-microservices-135148.html>

TIPS, TRICKS AND TUTORIALS

(<http://jaxenter.com/tag/tutorial>)



Keystone – An OpenStack Identity Service tutorial
(<https://jaxenter.com/keystone-openstack-identity-service-143274.html>)

SEARCH

(<https://jaxenter.com/keystone-openstack-identity-service-143274.html>)



Programming a crypto mining rig: How does it work?
(<https://jaxenter.com/programming-crypto-mining-rig-141099.html>)

(<https://jaxenter.com/programming-crypto-mining-rig-141099.html>)

TOPICS

DevOps (<https://jaxenter.com/tag/devops>)
Agile (<https://jaxenter.com/tag/agile-2>)
Java (<https://jaxenter.com/tag/java-2>)
Careers (<https://jaxenter.com/tag/careers>)
Open Source (<https://jaxenter.com/tag/open-source>)
IoT (<https://jaxenter.com/tag/iot>)
NetBeans (<https://jaxenter.com/tag/netbeans>)
Eclipse (<https://jaxenter.com/tag/eclipse-2>)
JavaScript (<https://jaxenter.com/tag/javascript>)
Blockchain (<https://jaxenter.com/tag/blockchain>)
Tutorials (<https://jaxenter.com/tag/tutorial>)

PAGES

Contact (<https://jaxenter.com/contact>)
Newsletter (<https://jaxenter.com/newsletter>)
Authors (<https://jaxenter.com/authors>)
Found a bug? (<https://jaxenter.com/found-bug>)
Advertise (<https://jaxenter.com/advertise>)
Privacy Policy (<https://jaxenter.com/privacy-policy>)
Terms of Use (<https://jaxenter.com/terms>)
Imprint (<https://jaxenter.com/imprint>)

FOLLOW JAXENTER

Twitter (<https://twitter.com/jaxentercom>)
Facebook (<https://www.facebook.com/pages/JAXentercom/123294781032857?fref=ts>)
Google+ (<https://plus.google.com/108137535157006873567/>)
RSS (<http://jaxenter.com/rss>)

S&S MEDIA

JAXenter.de (<http://jaxenter.de/>)
JAX Finance (<http://jax-finance.com/>)
JAX London (<https://jaxlondon.com/>)
JAX Germany (<https://jax.de/>)
DevOpsCon (<http://devopsconference.de/en/>)
Developer.Press (<http://developerpress.com/>)
International PHP Conference (<https://phpconference.com/2014/en>)
Webinale (<https://webinale.de/2015/>)
WebMagazin (<https://webmagazin.de/english>)
S&S Media (<http://sandsmedia.com/en>)

