From 0 To DevOps in 80 Days: The Dynatrace Transformation Story!

An excerpt from About:Performance blog

Market disruption can spark innovation and radical change, and DevOps — as a set of best practices — has emerged from software industry disruptions. Why? Because, over the years, delivering software in many organizations has become harder, slower and more error prone. Outdated technology became a disadvantage for older, established companies competing against startups without years of accumulated technical debt. Rigid processes and an old fashioned mindset blinded these enterprises to new business models hindered their ability to innovate.

Best Practices

About the Author

Andreas Grabner has been helping companies improve their application performance for 15+ years. He is a regular contributor within Web Performance and DevOps communities and a prolific speaker at user groups and conferences around the world. Reach him at @grabnerandi

For the complete version of this blog, go HERE

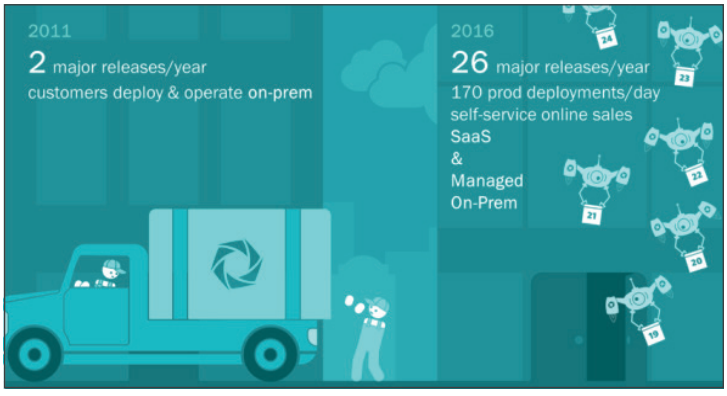
Motivation: Disrupted Market

The same happened for vendors in our market — Application Performance Management (APM). Dynatrace has existed since 2005 and, even back then, was disrupting and pioneering the 2nd generation of APM tools. In 2011 it became apparent that the market was going to be disrupted once again. Our existing customers and many software companies started to change the way they implement software which meant that monitoring also had to change and adapt. We saw:

- Applications migrating from On-Premise to Virtual, Cloud, Containers and PaaS
- Architects started including micro-services, on-demand scaling, failure and self-healing
- "Cloud Natives" demanded SaaS-based vs self-hosted monitoring solutions
- Monitoring had to bridge the gap between "Enterprise Stack" (Mainframe, WebSphere, WebLogic, ...) and "New Stack" (js, NGINX, Mobile, ...) technologies
- Digital Transformers started demanding Analytics for Biz, Dev, Sec and Ops and not just Traditional Monitoring
- New players in the APM space saw the opportunity and entered the market

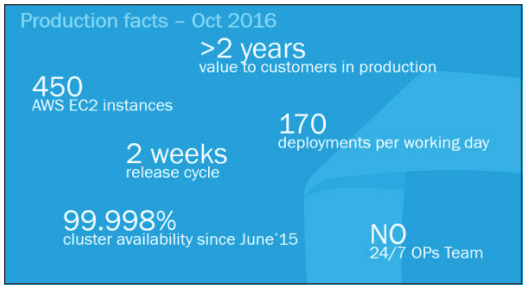
Goal: Introducing SaaS/Managed Offering with 1-hour Code Deploys

In order to maintain and extend the leadership role in such a disrupting market Dynatrace made a strategic decision to transform the way software was designed, developed, tested, built and operate within the engineering organization. The decision to move from 2 releases / year of an On Premise product to a 26 releases / year SaaS & On-Premise Managed offering was made, supported and pushed from the top down.



Transforming from On-Prem to SaaS/Managed model promised to deliver value faster to a changing market.

The way we achieved this transformation is explained in a recent webinar and in our PurePerformance Podcast, with our founder and CTO Bernd Greifeneder, in the webinar "From 0 to DevOps in 80 Days". We also captured the story from the viewpoint of Anita Engleder, DevOps Lead at Dynatrace, who helped building the delivery pipeline, automatic deployment and feedback loops for Dynatrace SaaS which for the past two years since its "go live" date has shown very impressive performance, scalability and success values:



Dynatrace SaaS Production Facts tell a success story on their own!

Lets dig into some of the lessons learned, and let them be food for thought for your own transformation:

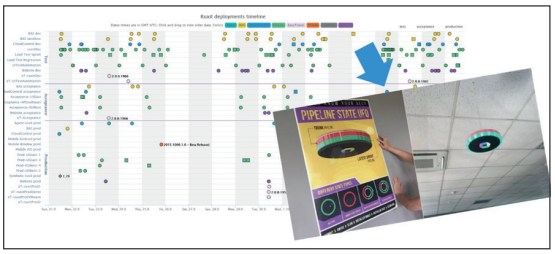
#1: Shift-Left Quality: Make Dev depend on Trunk Quality!

Even in 2011 we already implemented software using Agile principles. We had two-week sprints including proper planning and review meetings, as well as a Continuous Integration environment where code was built and unit tests executed on a regular basis. Yet it happened that during sprint reviews developers showed their features on their own dev workstation, without proofing whether the feature is really working in integrated sprint build.

We ended up deploying sprint builds from trunk on an automatically deployed environment we call dynaSprint. A feature was only considered "complete" when it could be demoed on that dynaSprint environment. This was the first step to make developers depend on the quality in trunk.

At that time, we also already used the latest Dynatrace official release to monitor our internal systems such as our website (custom built), our community (running on Confluence) and our bug tracking system (running on JIRA). We had a monitoring product – so – we of course used it internally. What we changed however was that we went from using the latest official release to deploying the latest sprint build. This gave another boost to quality considerations to the whole engineering team. Why? Because if a bad Dynatrace sprint build impacted the applications we monitored, we all immediately felt the impact, e.g: our customers can't access our website or community, and our developers can't keep track of their work in JIRA. While we had several bad deployments in the beginning, we learned from these mistakes. Instead of blaming, we made sure that the pipeline stopped bad code changes early so that critical bugs were identified earlier in the pipeline.

This was also the time we introduced our Pipeline State UFO which we use to visualize the status of the current trunk, as well as the current sprint build that is out there – visible to everyone in the engineering organization.



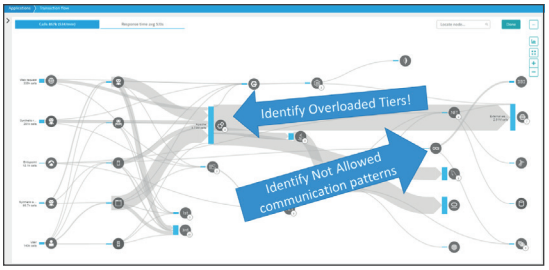
Raise quality awareness: Make engineering depending on Trunk and visualize the quality status.

If you are interested in these UFOs feel free to leave a comment or check out our project on GitHub. We're happy to show you how to get access to such an UFO and how to hook it up with your Jenkins, Bamboo, or even Dynatrace monitoring environment to visualize current state of dev, test or production!

#2: Make Architects Responsible for Production

In addition to raising quality awareness for every individual member of the engineering team, the Chief Software Architects were made responsible for production quality, performance and scalability.

This meant that architects were responsible to define the production monitoring strategy, which metrics we needed to capture, and the dashboards they looked at during the regular day of operations, as well as during their daily morning standups to investigate any open issues. They also came up with new and more-intuitive ways to visualize architectural bottlenecks, which is now a benefit for them as well as for our users!

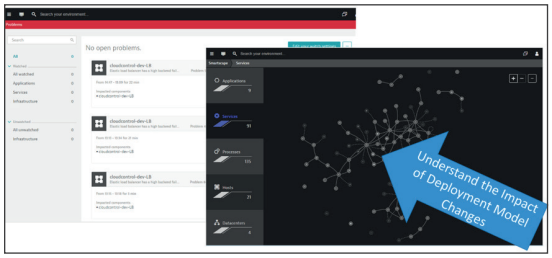


Transaction Flow visualization is a key instrument for architects to understand how their architecture really works under load.

Being an architect of an APM product, and using it on the same product, turned out to be very beneficial for us. This "first-hand experience" influenced many of the features and capabilities we now have in Dynatrace SaaS/Managed (aka Ruxit) as well as Dynatrace AppMon which are capable of monitoring automatically scaling, self-healing, and failure tolerant systems running both On-Premise and in the Cloud.

In order to not generate a lot of false warnings we learned about the importance of automatically detecting all dependencies between applications, services, processes, and the underlying infrastructure. Stopping and starting service instances after deployment events, or as a reaction of changed load behavior, might be normal as long

as it doesn't impact service-level baselines. All of this was factored into our software, which now makes it easier for our architects and our users to stay focused on real problems without having to battle excessive "noise".



Using our own software to monitor influenced many of the features we now take for granted, e.g: Automatic Problem, Deployment and Environment Detection.

#3: Shift-Right Metrics: Make Devs consume production metrics

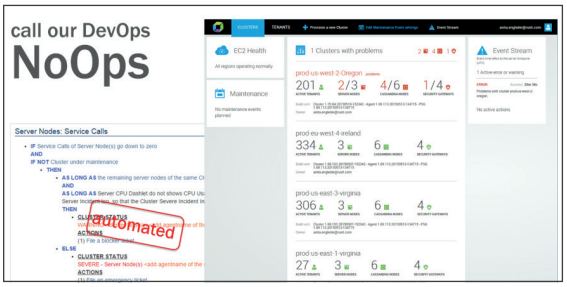
Even though monitoring production became the general responsibility of architects, we soon learned that developers must have full access to production monitoring data as well – especially focused on the features they are developing. Just because a feature could be demoed in the sprint review without breaking anything on our internal systems, doesn't mean that our real users don't have a problem. In order to get better and faster feedback for their features, we made it easy for developers to define their own custom metrics, instrumentation, and logging. They needed data to understand who is actually using their features, which browsers were used and, more importantly, how they navigate through their feature. Therefore, our engineers access our UEM (User Experience Management) data and draw their own conclusions about the effectiveness of a recent feature deployment. This level of feedback also allows them to start experimenting with different options to see how it impacts user behavior:

Engineering teams consume end user data to learn how their features are really used. User behavior analysis also enables quick experiments!

#4: NoOps: Automate Ops Runbooks

As we grew our SaaS business we learned how important it is to react to any problem impacting our end users. In traditional operations you would have a NOC team t responsible for "Keeping the Lights On", following Ops Runbooks in case certain conditions are met, e.g: restart processes or full servers should the system run into an error state.

When talking with our existing NOC team and defining all these runbooks, we realized that most of these actions could potentially be automated. The runbooks already resembled scripts with "If-Then" conditions followed by an action. Instead of keeping these runbooks alive the decision was made to automate at least one runbook per Sprint. This eventually led to a single remaining run book which said: IF the system can't heal itself through all the automation THEN Contact the Engineer on Call.



We automated all but one run book – achieving a situation where we do not need a traditional Ops Team to keep the system healthy.

More things we learned and did

These actions and many others allowed us to achieve the goal of transforming into a DevOps/NoOps cloud-native software company. We received support from top-level management that enabled teams to transform. We had motivated teams willing to go through the transformation. We also had the luxury of developing this new software that we could also use, while going through the transformation. We shaped the product to the needs of modern software delivery pipelines, and we believe that it will also greatly benefit all of our customers in their transformation journey.

As mentioned earlier – the full story has been discussed at webinars and podcasts. Check out the links:

- Webinar "From 0 to DevOps in 80 Days" with Bernd Greifeneder
- Webinar "From 6 Months to 1h Code Deploys" with Anita Engleder
- Podcast on "Leading the APM Market from Enterprise to Cloud Native" with Bernd Greifeneder
- Podcast on "Transforming 6 Months Release into 1h Code Deploys" with Anita
- Podcast on "Features and Feedback Loops @ Dynatrace" with Anita

If you want to see whether Dynatrace can help you, check out our Dynatrace AppMon & UEM Free Trial optimized for Development and the Continuous Delivery Pipeline. We also offer our new SaaS-based Monitoring as a trial – the first 1000h are on us!

Learn more at dynatrace.com

Dynatrace has redefined how you monitor today's digital ecosystems. AI-powered, full stack and completely automated, it's the only solution that provides answers, not just data, based on deep insight into every user, every transaction, across every application. More than 8,000 customers use Dynatrace to optimize customer experiences, innovate faster and modernize IT operations with absolute confidence.

05.15.17 2265_From0ToDevOps-BlogReprint_ww https://www.dynatrace.com/blog/from-0-to-devops-in-80-days-the-dynatrace-transformation-story/