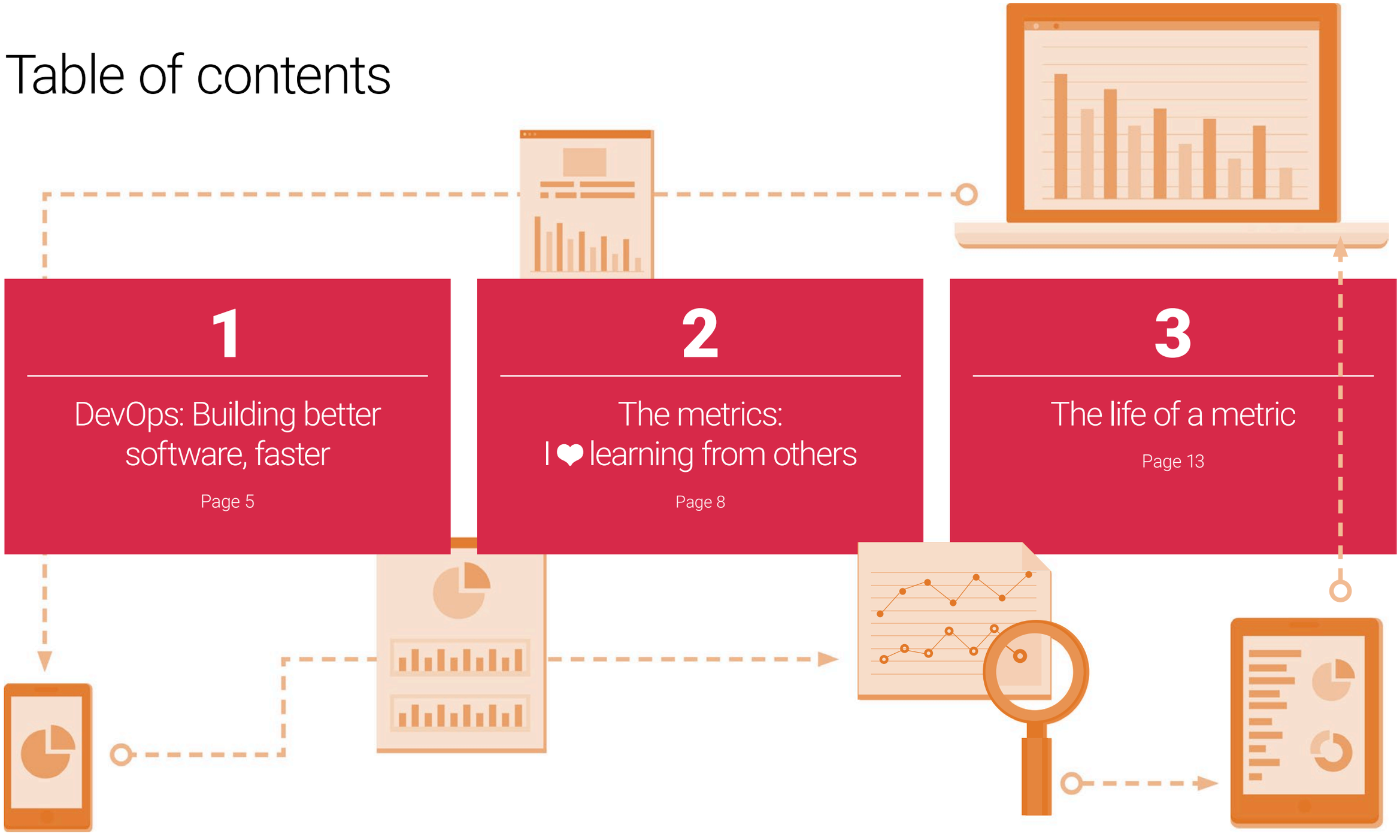
dynatrace

7 performance metrics to release better software, faster

With real-life examples of application failures and how to avoid them

Table of contents



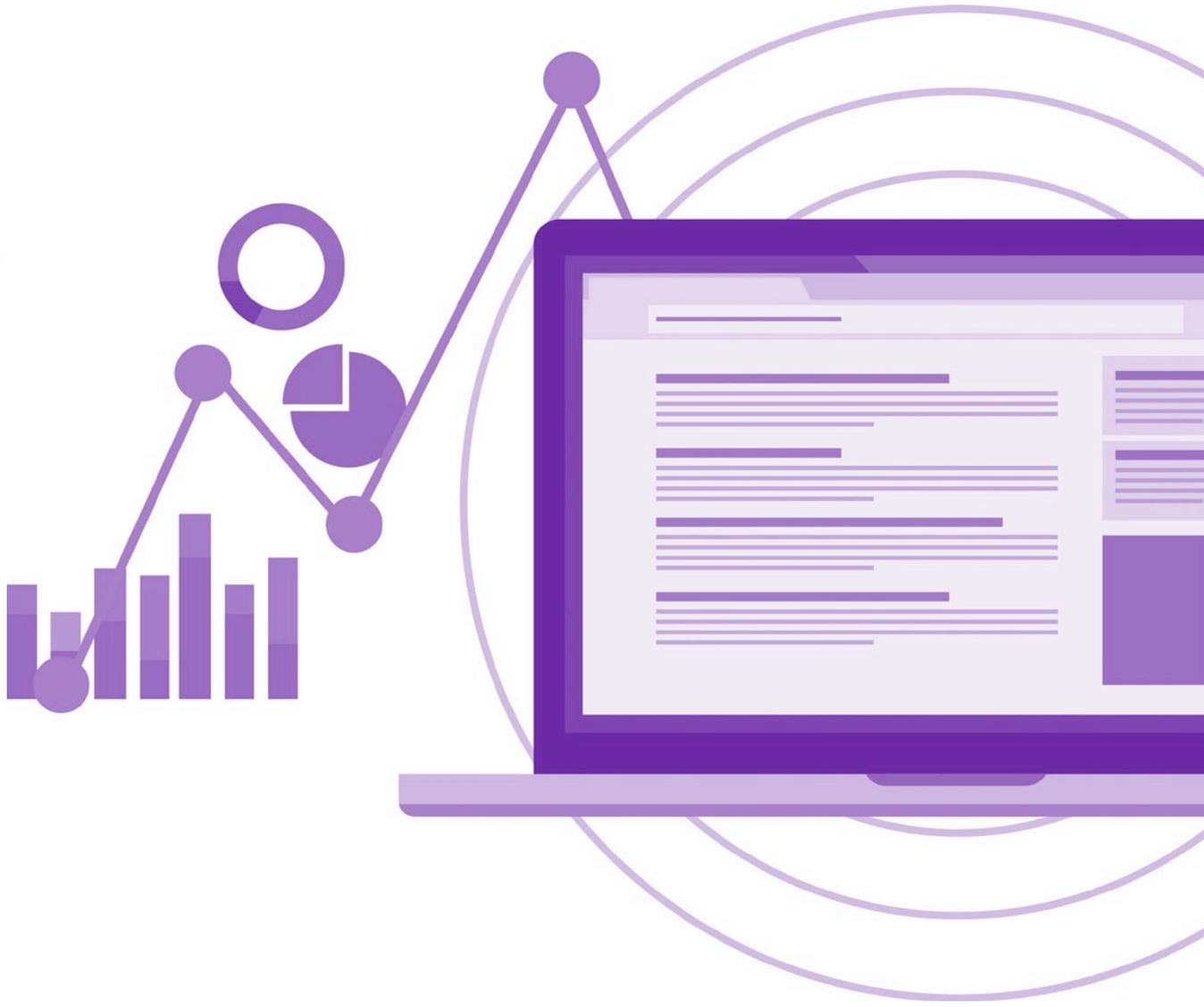
Introduction

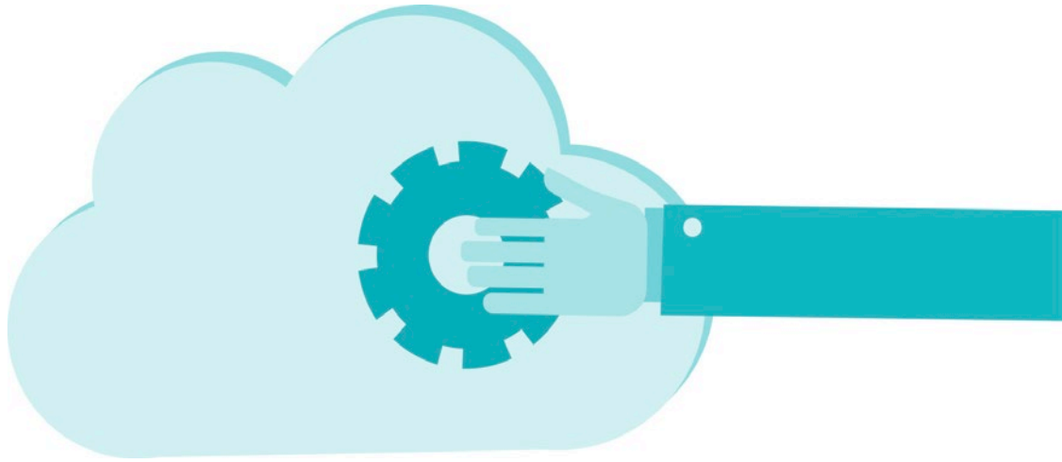
Today’s hyper-connected mobile consumers are driving a transformation across industries.

They have high expectations, and to compete you must provide a responsive, superior digital experience. The DevOps best practices of Continuous Delivery make this possible, but to do it right you will need a safety net. That is what metrics-driven application performance management is about. It allows you to deploy faster without failing faster — with key architecture, scalability, and performance metrics giving you full control over your software delivery pipeline.

In this eBook, we will tell you how to get started with performance metric monitoring. We will share real-life examples of application failures and how to avoid them by using 7 key metrics to find problems early on. Then we will show you how stuff really works in “the life of a metric” and how to integrate performance metrics into automation tools throughout your application delivery chain.

Happy monitoring!





Our authors:

Two guys that eat, drink, and breathe DevOps

Andreas Grabner

Performance Advocate,
Center of Excellence,
Dynatrace

Blog: blog.dynatrace.com
Twitter: @grabnerandi

Andreas Grabner has 15+ years' experience as an architect and developer in the Java and .NET space and is an advocate for high performing applications. He is a regular contributor to large performance communities and a frequent speaker at technology conferences.

Brett Hofer

Senior Solution Architect,
Dynatrace

Blog: blog.dynatrace.com
Twitter: @brett_solarch

Brett Hofer is passionate about DevOps and specializes in delivering complex mission critical software. With more than twenty years of experience—from product designer and solution architect to senior management—he has a unique 360° perspective of IT.

Back to Table of Contents

Section 1

DevOps: Building better software, faster

DevOps: Building better software, faster

A four-year-old with an iPad, the Chinese Uber, and a unicorn

What does a four-year-old with an iPad, the Chinese Uber (a.k.a. Didi Kuaidi), and a unicorn have in common? They are forcing us to build better software, faster.

There is a digital revolution happening across industries, driven by today's hyper-connected mobile consumers. They have high expectations — they want what they want, when they want it, and within their context (based upon location, task at hand, preferences, etc.).

Their mobile devices are responsible for more traffic and revenue than ever before:

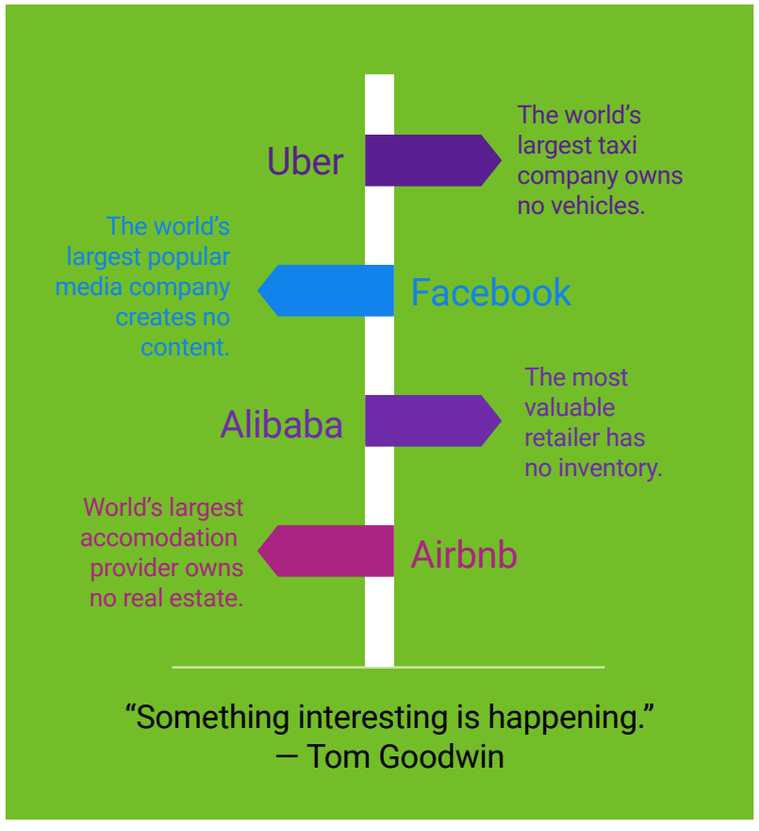
- **At Walmart on Cyber Monday 2014**, 70% of traffic was from Mobile.
- **Visa forecasts** that half of all payments will be mobile by 2020.
- In 2014, shopping apps sessions (iOS and Android) **grew 174% YOY.**[1]

The unicorn in all this? Technology giants like Facebook, Google, Etsy, and Pinterest. They are leading the way by providing responsive and superior digital experiences.

These unicorns are also answering the consumer with completely new business models. Uber, the world's largest taxi company, doesn't own a vehicle and Facebook, the world's most poplar media owner, creates no content.

In Asia, this digital revolution is amplified. Uber provides a very large number of taxi rides — 1 million. But the equivalent in China? It provides 5 million.[2]

How are you responding to these rapidly changing markets, requirements, and user expectations? Do you have the right processes and metrics in place to support continuous delivery and to build a better digital experience, faster — all the while keeping your customers happy?



1. flurrymobile.tumblr.com
2. KPCB, Uber. Date: May 2014 (Didi), Feb 2015 (Uber)

DevOps: Building better software, faster

How to deploy faster and not fail faster

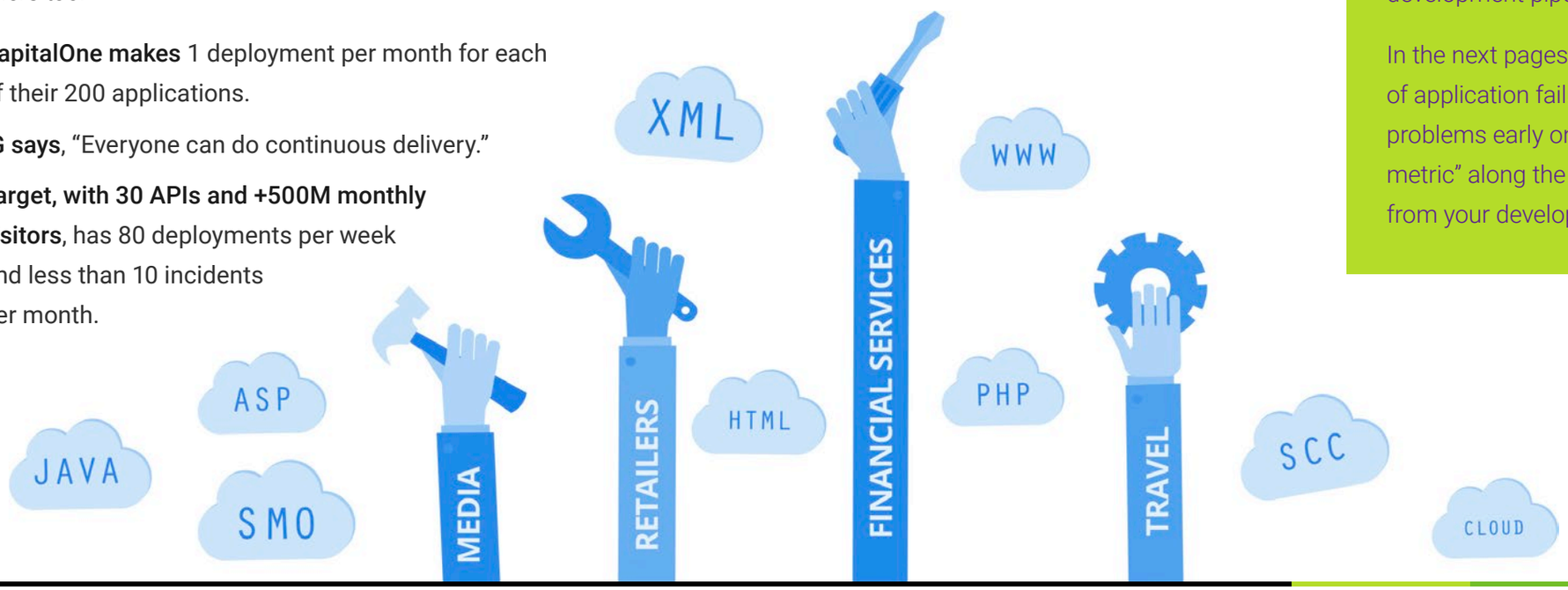
DevOps is about responding to changing market dynamics to get ideas to end customers as fast as possible, minimizing feature cycle time. The DevOps unicorns are not just Facebook and Google though.

They are financial services organizations and retailers too:[3]

- **CapitalOne makes** 1 deployment per month for each of their 200 applications.
- **IG says**, "Everyone can do continuous delivery."
- **Target, with 30 APIs and +500M monthly visitors**, has 80 deployments per week and less than 10 incidents per month.

You may be wondering about results beyond these individual companies. Does DevOps really pay off? Well, DevOps high performers are:[4]

- **More agile**, with 30x more frequent deployments and 8,000x faster lead times than their peers.
- **More reliable**, with 2x the change success rate and 12x faster mean time to recover.



What do you need to do DevOps like a unicorn?

You need a safety net. You need performance monitoring with complete visibility and the right metrics as quality gates throughout the application development pipeline.

In the next pages, we will use three real-life examples of application failures and the metrics you need to find problems early on. Then we will follow "the life of a metric" along the application development pipeline — from your developers' IDE through to production.

dynatrace

3. Velocity conference, 2015
4. PuppetLabs, 2013 (survey of 4,000 organizations)

Section 2

Metrics: I ❤ learning from others

Metrics: I ❤ learning from others

Don’t push without a plan

If your applications aren’t prepared and are slow to load (or they crash), your smartest marketing campaign is actually your worst marketing campaign.

Metrics

1. NUMBER OF RESOURCES
2. SIZE OF RESOURCES
3. PAGE SIZE

Case Study 1:
400 selfies

Here is an example of a great marketing campaign with poor implementation. One of the Superbowl advertisers asked consumers to take selfies, and then upload them to the mobile website. The plan was for an ad to run during the Superbowl, driving traffic to the site to see the last 400 selfies uploaded.

Unfortunately, it was implemented as a 20x20 image grid with individual image files served from the same image domain. When we tested it with an iPhone, we had to download a whopping **20 MB of content and 434 resources — completely ignoring web performance best practices.**

Case Study 2:
The favicon that was 150x too big

During another major sporting event, the largest element on the organizer’s mobile website landing page was a favicon the size of 370 kilobytes. This is the icon that is typically in the top left of the browser window and is usually 16x16 or 32x32 pixels.

They had taken the large resolution logo and had directly converted it into an icon and put it on the website.

It should have been a couple of bytes not 370 kilobytes. Someone wasn’t thinking.

These two instances required basic web performance optimization and performance testing from the developer’s machine throughout the development pipeline — prior to going live for big event.

Metrics: I ❤ learning from others

Don’t assume you know the environment

Your applications execute on the edge of the Internet, across an array of devices (e.g. smartphones, tablets, browsers) across service provider backbones and 3rd parties. There are many points of potential failure so don’t assume you know the environment without real user insight.

Metrics

④ NUMBER OF FUNCTIONAL ERRORS

⑤ 3RD PARTY CALLS

300 miles/hour — a world record!

Runtastic is a social mobile app that tracks your workout and allows you to put it on Facebook to brag to your friends. In this case, somebody biked for an hour and the app told him that he biked 480 km. That’s 300 miles/hour — a world record!

When this happens, people are less likely to use your app and will go on Twitter, Facebook, or the app store and say, “This is s%#&! Your app doesn’t work! Don’t buy it!!” The engineers in this case had a plan.

They knew they couldn’t test all combinations of the Android OS on all devices so:

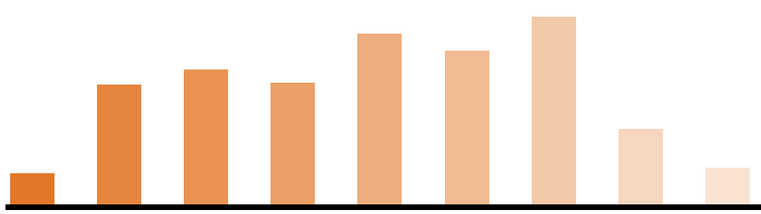
> They developed a unit test to check the GPS distance calculation every time the deployed app starts.

> Then, using real-user monitoring every single time the app ran on millions of installations, they found that the problem occurs only on a certain Android hardware configuration.

> A Google search told them that it was a well-known issue with the OS/hardware configuration, and out of Runtastic’s control.

> They told their users about the problem and suggested it was time to upgrade their Android version, using channels such as Twitter or Facebook and a built-in news feed in the app.

Runtastic users also have problems sometimes uploading their results to Facebook and Twitter. By actively monitoring, Runtastic points out the 3rd party source of the issue and tell users to try again later.



Metrics: I ❤ learning from others

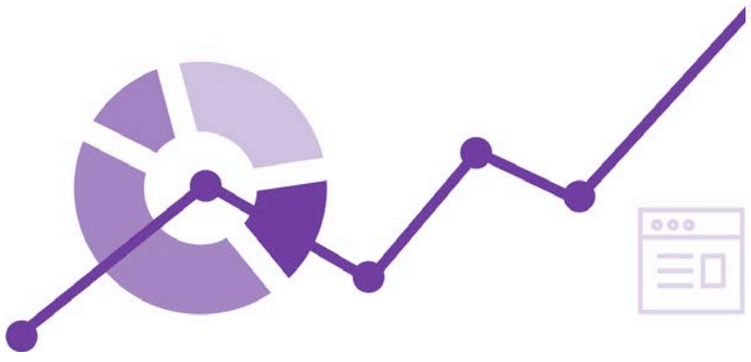
Don’t blindly (re) use components

Your engineering team is resourceful. Even if they don’t know how to do something immediately they will figure it out. This doesn’t always work in your favor, and that is why you need to monitor closely before things get to production.

Metrics

⑥ NUMBER OF SQL EXECUTIONS

⑦ NUMBER OF SAME SQL’S



Developers are always reusing existing components. The next example actually happened to us at Dynatrace. We needed a simple HTML report on our free trial registrations. He didn’t have much experience in creating HTML pages or client-facing reports so he did what developers do and Googled it. Our back-end developer found the sample code on GitHub and plugged it in. The report worked fine in the beginning but then it slowed over a few months to a crawl — taking minutes to render.

We looked behind the scenes and found the sample he re-used was based upon Hibernate — a very popular Object Relational (OR) Mapping Framework to access data in the database. The code wasn’t using any of the advanced Hibernate optimization features for better database access. In this case, Hibernate was executing 4K database statements to report three lines in the output HTML report.

More metrics

These 7 metrics are just the beginning. An easy way to get started is to track them manually — throughout your application development pipeline. Once you have a handle on what you need to know, it’s time to start looking at how to automate performance monitoring. This is the goal of continuous delivery: automate your development pipeline with quality gates based upon metrics in each stage.

Here are some more metrics to consider:

- **Time spent in API**
- **Calls into API**
- **# of domains**
- **Total size**
- **# of items per page**
- **# AJAX per page**

For more metrics check out our blog series on software quality metrics.

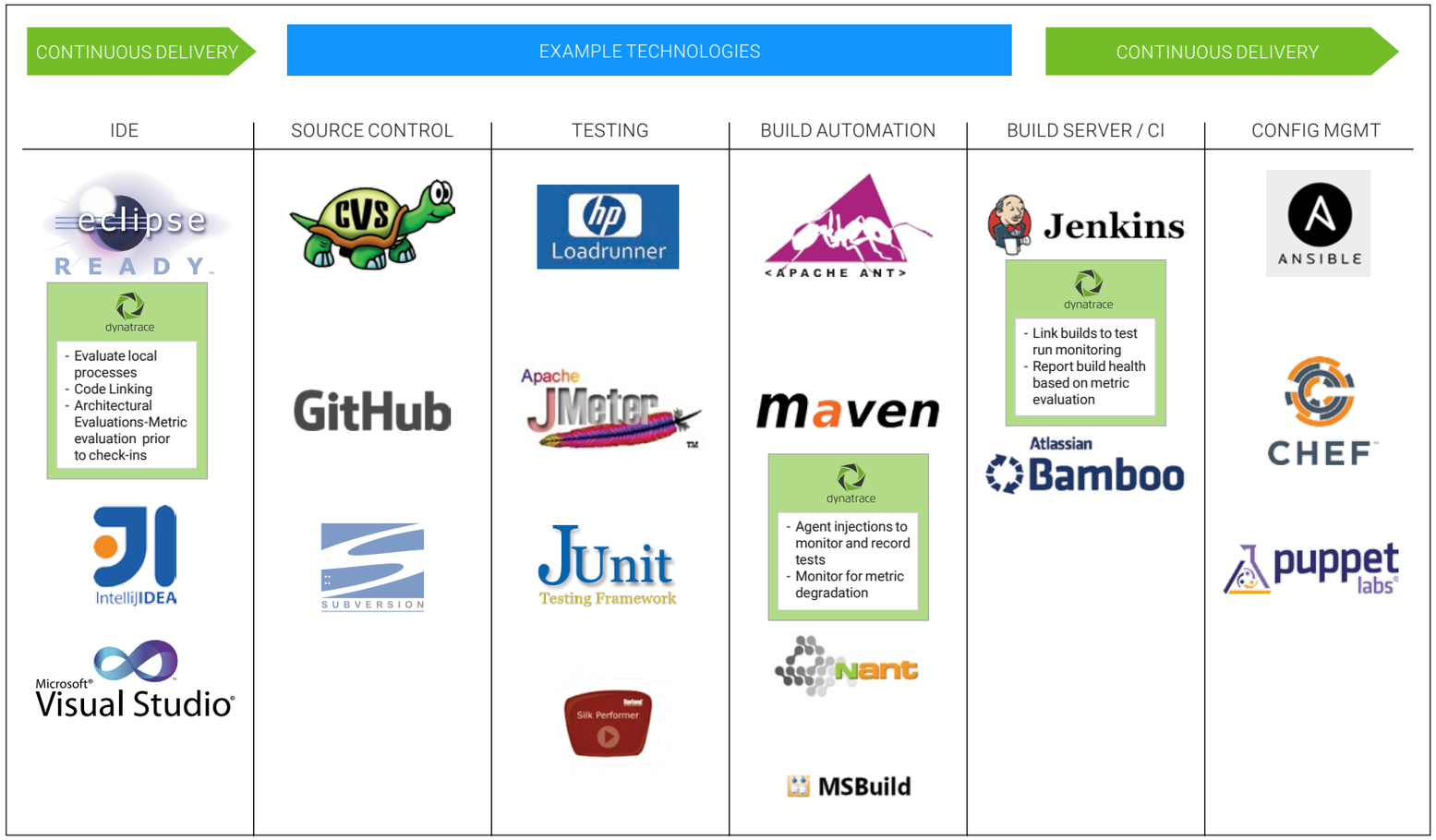
Section 3

The life of a metric

The life of a metric

You may be wondering: Now, how does this actually work?

How and when do I collect these performance metrics? To answer this question, we will take you through the life of a metric — in this case, # of SQL executions — with a few steps in the continuous delivery lifecycle.



1. **Developer's IDE/Eclipse:** The developer's workstation is the most underused place for performance monitoring. It's pretty simple. If your developers are able to do the analysis before checking in code, they can stop issues earlier in the lifecycle and reduce overall costs.

   In this example, performance monitoring tools will be integrated into Eclipse. Maximums are set for performance metrics, including our metric — # of SQL executions. Then, every time the developer executes a SQL call, performance monitoring will be tracking it.

The life of a metric

You may be wondering: Now, how does this actually work?

2. **Unit test/JUnit:** In the next step, the developer authors the JUnit test case class. Performance monitoring is configured to watch test cases and to roll up each pre-defined metric per test case. This is where the developer will evaluate the # of SQL executions per test.

3. **Build automation/Maven:** The developer will then run a MAVEN test with local code and uses performance monitoring to evaluate the call stacks and the # of SQL executions. At this point, the architect can do a quick analysis of metrics and architectural validation, instead of a long drawn out code review. Then the developer decides whether the code is ready to be promoted to the configuration management system.

4. **CI build server/Jenkins:** If all is good in the above steps, the developer checks in the code. The Jenkins build server brings down the change, and runs the test cases. Performance monitoring alerts may be fired. Jenkins goes to the performance monitoring server and collects data about what happened, including the # of SQL executions. Deployment decisions are made “Go” or “No go” with exception reporting.

The result? Functional and architectural confidence prior to auto deployment.

Your next steps

We know you are facing some serious changes in markets, requirements, and user expectations. Your users want a responsive digital experience and they want it now. Continuous delivery is the answer — if you build in a performance safety net. We hope these metrics give you a good starting point. Go forward to automate as much as possible, and keep focusing on faster and better!

Below are our some additional resources that will help you on your journey.

DevOps tools we love

Here is a collection of tools that foster collaboration amongst Product Management, Development, IT Operations, and Technical Support teams to allow them you build more quality your their products, and support you in establishing better feedback loops.

- Change Controls — JIRA Agile
- Development — Eclipse
- Source Control — Git/GitHub
- Build Automation — Ant and Maven
- Configuration Management — Ansible, Chef, and Puppet
- Test Automation — LoadRunner and Selenium
- VM, Cloud and Container Technologies: — Vagrant, Packer, VeeWee, Docker, and Cloud Foundry

Webinar replays

- 5 Key Metrics to Release Better Software Faster
- DevOps: From Adoption to Performance

The blogroll

- DevOps reactions — Enjoy some DevOps humor!
- Dynatrace APM on DevOps
- IT Revolution's DevOps
- Software Quality Metrics for Your Continuous Delivery

Recommended reading

- *Continuous Delivery* by Jez Humble and David Farley

Learn more at dynatrace.com

Dynatrace is the innovator behind the industry's premier Digital Performance Platform, making real-time information about digital performance visible and actionable for everyone across business and IT. We help customers of all sizes see their applications and digital channels through the lens of their end users. Over 8,000 organizations use these insights to master complexity, gain operational agility and grow revenue by delivering amazing customer experiences.

6.30.16 417_EBK_7PerformanceMetrics_jg

dynatrace