

Vom IT-Betrieb zu Platform Engineering. Ein Reisebericht (2/3)



Vor einer Woche habe ich berichtet, woher wir gekommen sind, welche Probleme wir hatten und wie wir mit Veränderungen wie z.B. der Einführung einer [SOA-Architektur](#) umgegangen sind.

Wir hatten schon einen wichtigen Schritt gemacht, indem wir unsere Arbeit sichtbar gemacht haben. Dadurch konnten wir priorisieren. Wir haben bewusst Themen abgemeldet bzw. erst für später eingeplant, um wiederum andere Themen mit einer höheren Priorität früher zu machen.

Wir hatten aber immer noch Probleme. Die Ticketanzahl stieg, wir waren immer noch zu langsam, und von Weiterentwicklung wollen wir gar nicht reden. Die Reise war also noch nicht zu Ende...

Kapitel 2: Basistechnologien, SelfServices & Automation

Wir wussten, dass wir nicht schnell genug waren. Die Anzahl der Tickets stieg

und acht Wochen Vorlauf für einen neuen Service in Produktion klingt irgendwie nicht zeitgemäß, oder? An welchen Schrauben sollten wir drehen? Die Teams, die wir unterstützen sollen, warteten zu oft auf uns. Immer wieder kam z.B. die Aussage, dass ein Server in der Cloud in wenigen Minuten verfügbar ist. Wir als Team waren mit operativer Arbeit überlastet und hatten nur wenig Zeit für Weiterentwicklung. Irgendwie mussten wir im Tagesgeschäft schneller werden, um mehr Zeit für die Weiterentwicklungen unserer eigenen Themen zu bekommen. Wenn wir einen Mehrwert bieten wollen, dann müssen wir uns mit neuen Technologien auseinandersetzen. Nicht zuletzt machen neue Technologien und Weiterentwicklung auch mehr Spaß als das hoch standardisierte und wiederholte Einrichten von neuen Servern, Monitoring, Datensicherung, Logging,

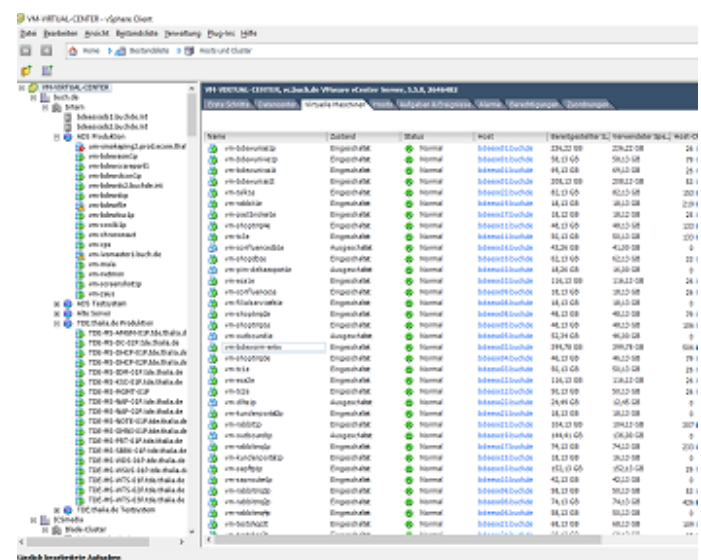


Was fehlte uns, um schneller zu werden? Wir hatten das Glück, dass wir die Chance hatten, Einblicke in die Arbeitsweise und das Mindset von Otto.de zu bekommen (vielen Dank und viele Grüße an die Kollegen von Otto.de!). Ein Satz ist mir besonders hängen geblieben, weil ich ihn anfangs erschreckend schlimm fand. Kurz zusammengefasst: Kommunikation macht langsam! Das meinten die Jungs jetzt nicht ernst, oder? Gute Kommunikation ist doch die Basis! Klar habe ich es erst missverstanden. Gute Kommunikation ist natürlich sehr wichtig. Gemeint war damit die Idee, dass **wenn zwei Teams miteinander reden müssen** bzw. Themen ein Team verlassen, im nächsten Team bearbeitet werden und dann wieder zurück müssen, dort geprüft werden, Fehler gefunden, wieder zurück an das andere Team zur Nachbesserung, zurück zum Test, Kommt das jemandem bekannt vor? Klingt das nach Topperformance? Solche Kommunikation an Schnittstellen macht einfach langsam. Versuche, Schnittstellen zu reduzieren. Versetze den Anforderer in die Lage, seine Anforderung selber umzusetzen, ohne Spezialisten-KnowHow zu haben. **Biete SelfServices an!** Ein SelfService basiert dabei auf einem Automaten mit einem einfachen Frontend. Über das Frontend ist

der Anforderer in der Lage, seine wiederkehrenden Aufgaben einzugeben. Der Automat erledigt dann die hoch standardisierte Umsetzung der Aufgabe in einer fehlerfreieren Qualität als ein Mensch es könnte.

Nur wo fange ich an? Leichter gesagt als getan. Wir haben uns daran orientiert, wo wir als IT-Betrieb die größten Schmerzen hatten und wo wir den neuen Produkt-Teams den größten Nutzen bringen konnten. Hier drei technische Beispiele, die uns nach vorne gebracht haben.

Beispiel 1: Automatisierte Service Bereitstellung (ASB)



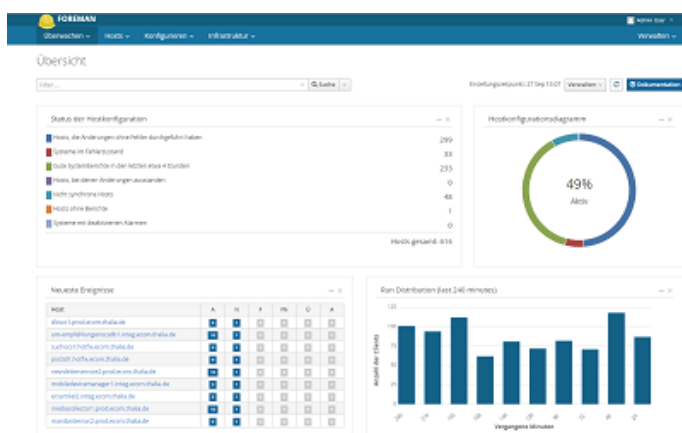
Name	Status	Datastore	Host	Power state	Memory
vms-1	Engaged	Normal	ESX-100	On	1024 MB
vms-2	Engaged	Normal	ESX-100	On	1024 MB
vms-3	Engaged	Normal	ESX-100	On	1024 MB
vms-4	Engaged	Normal	ESX-100	On	1024 MB
vms-5	Engaged	Normal	ESX-100	On	1024 MB
vms-6	Engaged	Normal	ESX-100	On	1024 MB
vms-7	Engaged	Normal	ESX-100	On	1024 MB
vms-8	Engaged	Normal	ESX-100	On	1024 MB
vms-9	Engaged	Normal	ESX-100	On	1024 MB
vms-10	Engaged	Normal	ESX-100	On	1024 MB
vms-11	Engaged	Normal	ESX-100	On	1024 MB
vms-12	Engaged	Normal	ESX-100	On	1024 MB
vms-13	Engaged	Normal	ESX-100	On	1024 MB
vms-14	Engaged	Normal	ESX-100	On	1024 MB
vms-15	Engaged	Normal	ESX-100	On	1024 MB
vms-16	Engaged	Normal	ESX-100	On	1024 MB
vms-17	Engaged	Normal	ESX-100	On	1024 MB
vms-18	Engaged	Normal	ESX-100	On	1024 MB
vms-19	Engaged	Normal	ESX-100	On	1024 MB
vms-20	Engaged	Normal	ESX-100	On	1024 MB
vms-21	Engaged	Normal	ESX-100	On	1024 MB
vms-22	Engaged	Normal	ESX-100	On	1024 MB
vms-23	Engaged	Normal	ESX-100	On	1024 MB
vms-24	Engaged	Normal	ESX-100	On	1024 MB
vms-25	Engaged	Normal	ESX-100	On	1024 MB
vms-26	Engaged	Normal	ESX-100	On	1024 MB
vms-27	Engaged	Normal	ESX-100	On	1024 MB
vms-28	Engaged	Normal	ESX-100	On	1024 MB
vms-29	Engaged	Normal	ESX-100	On	1024 MB
vms-30	Engaged	Normal	ESX-100	On	1024 MB
vms-31	Engaged	Normal	ESX-100	On	1024 MB
vms-32	Engaged	Normal	ESX-100	On	1024 MB
vms-33	Engaged	Normal	ESX-100	On	1024 MB
vms-34	Engaged	Normal	ESX-100	On	1024 MB
vms-35	Engaged	Normal	ESX-100	On	1024 MB
vms-36	Engaged	Normal	ESX-100	On	1024 MB
vms-37	Engaged	Normal	ESX-100	On	1024 MB
vms-38	Engaged	Normal	ESX-100	On	1024 MB
vms-39	Engaged	Normal	ESX-100	On	1024 MB
vms-40	Engaged	Normal	ESX-100	On	1024 MB
vms-41	Engaged	Normal	ESX-100	On	1024 MB
vms-42	Engaged	Normal	ESX-100	On	1024 MB
vms-43	Engaged	Normal	ESX-100	On	1024 MB
vms-44	Engaged	Normal	ESX-100	On	1024 MB
vms-45	Engaged	Normal	ESX-100	On	1024 MB
vms-46	Engaged	Normal	ESX-100	On	1024 MB
vms-47	Engaged	Normal	ESX-100	On	1024 MB
vms-48	Engaged	Normal	ESX-100	On	1024 MB
vms-49	Engaged	Normal	ESX-100	On	1024 MB
vms-50	Engaged	Normal	ESX-100	On	1024 MB
vms-51	Engaged	Normal	ESX-100	On	1024 MB
vms-52	Engaged	Normal	ESX-100	On	1024 MB
vms-53	Engaged	Normal	ESX-100	On	1024 MB
vms-54	Engaged	Normal	ESX-100	On	1024 MB
vms-55	Engaged	Normal	ESX-100	On	1024 MB
vms-56	Engaged	Normal	ESX-100	On	1024 MB
vms-57	Engaged	Normal	ESX-100	On	1024 MB
vms-58	Engaged	Normal	ESX-100	On	1024 MB
vms-59	Engaged	Normal	ESX-100	On	1024 MB
vms-60	Engaged	Normal	ESX-100	On	1024 MB
vms-61	Engaged	Normal	ESX-100	On	1024 MB
vms-62	Engaged	Normal	ESX-100	On	1024 MB
vms-63	Engaged	Normal	ESX-100	On	1024 MB
vms-64	Engaged	Normal	ESX-100	On	1024 MB
vms-65	Engaged	Normal	ESX-100	On	1024 MB
vms-66	Engaged	Normal	ESX-100	On	1024 MB
vms-67	Engaged	Normal	ESX-100	On	1024 MB
vms-68	Engaged	Normal	ESX-100	On	1024 MB
vms-69	Engaged	Normal	ESX-100	On	1024 MB
vms-70	Engaged	Normal	ESX-100	On	1024 MB
vms-71	Engaged	Normal	ESX-100	On	1024 MB
vms-72	Engaged	Normal	ESX-100	On	1024 MB
vms-73	Engaged	Normal	ESX-100	On	1024 MB
vms-74	Engaged	Normal	ESX-100	On	1024 MB
vms-75	Engaged	Normal	ESX-100	On	1024 MB
vms-76	Engaged	Normal	ESX-100	On	1024 MB
vms-77	Engaged	Normal	ESX-100	On	1024 MB
vms-78	Engaged	Normal	ESX-100	On	1024 MB
vms-79	Engaged	Normal	ESX-100	On	1024 MB
vms-80	Engaged	Normal	ESX-100	On	1024 MB
vms-81	Engaged	Normal	ESX-100	On	1024 MB
vms-82	Engaged	Normal	ESX-100	On	1024 MB
vms-83	Engaged	Normal	ESX-100	On	1024 MB
vms-84	Engaged	Normal	ESX-100	On	1024 MB
vms-85	Engaged	Normal	ESX-100	On	1024 MB
vms-86	Engaged	Normal	ESX-100	On	1024 MB
vms-87	Engaged	Normal	ESX-100	On	1024 MB
vms-88	Engaged	Normal	ESX-100	On	1024 MB
vms-89	Engaged	Normal	ESX-100	On	1024 MB
vms-90	Engaged	Normal	ESX-100	On	1024 MB
vms-91	Engaged	Normal	ESX-100	On	1024 MB
vms-92	Engaged	Normal	ESX-100	On	1024 MB
vms-93	Engaged	Normal	ESX-100	On	1024 MB
vms-94	Engaged	Normal	ESX-100	On	1024 MB
vms-95	Engaged	Normal	ESX-100	On	1024 MB
vms-96	Engaged	Normal	ESX-100	On	1024 MB
vms-97	Engaged	Normal	ESX-100	On	1024 MB
vms-98	Engaged	Normal	ESX-100	On	1024 MB
vms-99	Engaged	Normal	ESX-100	On	1024 MB
vms-100	Engaged	Normal	ESX-100	On	1024 MB

Vorher: Server klonen mit VMware

Wie ihr wisst, haben wir acht Wochen Vorlauf für einen Service in Produktion benötigt. Das muss natürlich auch schneller mit weniger Vorlauf gehen. Warum haben wir acht Wochen Vorlauf gebraucht? Primär aufgrund der Abstimmung zwischen IT-Betrieb und Entwicklung. Von den acht Wochen haben wir drei Wochen für die Abstimmung eingeplant. Eine Woche ist für den Bau der gesamten Infrastruktur eingeplant worden, so dass nach vier Wochen die Infrastruktur für den neuen Service bereitstand. Aber warum dann acht Wochen? Vier Wochen vor Produktion startet das erste Testdeployment. Hier muss die Infrastruktur fertig sein, damit die Tests starten können. Auch in der Testphase ist immer wieder aufgefallen, dass die bereitgestellte Infrastruktur in sich nicht immer konsistent war. So waren Server trotz Standard VMware Image gerne mal unterschiedlich konfiguriert, es fehlten Berechtigungen oder Benutzer waren nicht angelegt. Um die Fehler in der Massenbereitstellung zu reduzieren, hilft nur eines wirklich:

Automatisiere! Stelle Services automatisch und ohne Expertenwissen bereit.

Baue einen Automaten mit einem Webfrontend. Gebe dem Entwicklungsteam einen Knopf, auf den man drücken kann und aus dem dann nach wenigen Minuten ein Server rausfällt. Ein Server? Wer will den einen Server haben? Erweitere den Knopf, so dass nach der automatischen Server-Installation (das kann eine public Cloud eh besser) auch gleich noch das Betriebssystem passend für den **Service** konfiguriert wird, dann der Service installiert und konfiguriert wird und in Produktion gestellt wird. Das kann eine public Cloud für unsere Services „out of the box“ nicht -> Cool, Mehrwert generiert. Klingt das gut? ☐



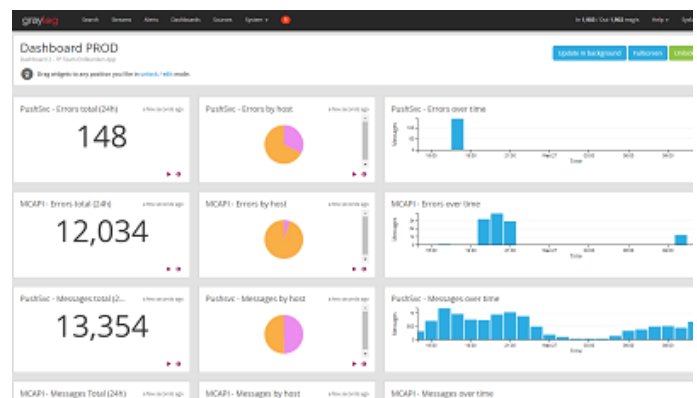
Nachher: Service Bereitstellung auf Foreman & Puppet Basis

Leute aus den Entwicklungsteams und dem IT-Betrieb haben also damit begonnen, basierend auf Foreman, Puppet und ein paar anderen Helferlein eine automatisierte Service Bereitstellung (Intern kurz: ASB) aufzubauen. Der Fokus lag im ersten Schritt auf dem Basis Betriebssystem und JAVA/Tomcat Services. Davon hatten wir die meisten, und es würde somit den größten Nutzen bringen. In dieser Zeit haben wir auch die Basis-Architektur für den Automatismus definiert und aufgebaut. Darauf basieren bis heute alle neu entwickelten Automatismen z.B. für DNS oder Apache Webserver. Wir haben einen starken Fokus auf den Aufbau der ASB-Infrastruktur gegeben, wodurch sicherlich auch andere Arbeit liegen geblieben ist. Jedoch konnten wir nur so den Entwicklungsteams die Möglichkeit geben, Server inkl. Services ohne Rückfrage beim IT-Betrieb aufzubauen. Egal in welcher Stage. Konsequenz: Der SelfService wird so gut genutzt, dass wir heute über alle Stages weit über 1.000 Server betreiben.

Beispiel 2: Zentrales Logdaten Management (ZLM)

Vorher: Logdaten Analyse auf der Console

gibt ja nicht nur Störungen. SOA und Microservice sei Dank gibt es ja fast täglich neue Services, die auch in das Log-Management möchten. Mal abgesehen von den Kapazitäten im Logdaten Management System beginnt das traditionelle Spiel im Ticketsystem: Neues Ticket: „Ich benötige Logging“, klar baue ich dir gerne, so fertig, hmm ich sehe nichts, ups jetzt aber, OK funktioniert, aber warum kann ich nicht mehr Daten ins System geben, aus den Büchern der menschlichen Schnittstellenkommunikation □ . Wie kann ich solche Kommunikation vermeiden? Baue einen **SelfService!** Plane und implementiere eine Logdaten Management Infrastruktur, die leistungsstark genug ist, um dein Volumen locker zu handhaben und wenig störanfällig ist. Sorge dafür, dass die Infrastruktur skaliert (nach oben und nach unten). Stelle eine gut dokumentierte Schnittstelle bereit, über die ein Entwicklungsteam selber seine Daten in das Logdaten Management schreiben kann und die Möglichkeit hat, die Daten selber auszuwerten.



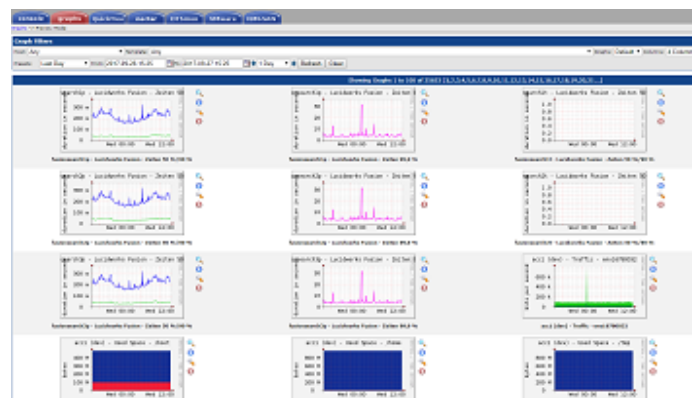
Nachher: Logdaten Management auf Graylog Basis

Auch den Aufbau eines neuen und leistungsfähigen Logdaten Managements konnten wir recht früh starten. Auch hier setzen wir auf das bewährte Prinzip, Entwicklerteams und IT-Betrieb zusammenzubringen. Schnell wurde aber klar, dass es nicht besonders schlau ist, alle Logdaten, die es gibt, ohne Sinn und Verstand in eine Datenbank zu pressen. Das ist zwar möglich, aber nicht schlau. So haben wir uns u.a. mit den Fragen auseinandergesetzt „Was und wie loggen wir überhaupt?“, „Wann brauchen wir eine Logausgabe?“ und „Was machen die Logdaten?“. Nach vielen Gesprächen über Anforderungen und Optionen haben wir einen Vorschlag erarbeitet, wie die Entwicklungsteams Logging in ihren Services nutzen können. Parallel dazu haben wir eine neue Logdaten Management Infrastruktur basierend auf Graylog aufgebaut. Graylog bietet

zusätzlich zum ELK-Stack noch ein paar nette Features wie z.B. ein User Management. Auch erschienen uns das Handling und der Betrieb im Vergleich zum ELK-Stack etwas einfacher. Gesized wurde die neue Infrastruktur für Spitzenlastzeiten. Bei uns ist das neben dem Schulbuchgeschäft natürlich das Weihnachtsgeschäft. So kamen wir auf eine Infrastruktur von u.a. zwei Elastic Search Master Nodes und acht Elastic Search Data Nodes in zwei Datacentern. Durch die Data Nodes ist die Infrastruktur jederzeit ohne viel Aufwand skalierbar. Im Peak kann die Infrastruktur bis zu 100.000 Messages die Sekunde verarbeiten. Um die künftigen Nutzer der Infrastruktur zu informieren und möglichst früh Feedback zu bekommen, haben wir zum Thema eine von uns so getaufte „Bier-Session“ angesetzt. Eine „Bier-Session“ ist in etwa vergleichbar mit einer Brownbag-Session, jedoch war es nicht in der Pause und es gab (alkoholfreies) Freibier. Das Feedback haben wir gerne aufgenommen und in die Infrastruktur eingebaut.

Was haben wir mit der neuen Infrastruktur erreicht? Die Produkt-Teams können nun ihre eigenen Anwendungen durch eine einfache Konfiguration in der „logback.xml“ anbinden. Auch die Dashboards zur Auswertung der Logdaten können sich die Teams ohne Unterstützung des Platform Engineering Teams einrichten. Ein weiterer schöner SelfService, der die Aufwände im IT-Betrieb reduziert und die Arbeit in den Produkt Teams beschleunigt.

Beispiel 3: Automatisiertes Monitoring



Vorher: CACTI Monitoring hat seinen Dienst getan

Monitoring war auch so eine super Sache. Wir monitoren ja echt viel – mit einigen

unterschiedlichen Monitoring- & Alarmierungssystemen bekommen wir einen super Überblick über den Zustand unserer Services aus technischer Sicht und aus Kundensicht. Jedoch mit was für einem Aufwand! Sämtliche Sensoren wurden von einem Menschen (immerhin haben wir für das Monitoring einen dedizierten Menschen) manuell in die Monitoring Systeme eingetragen. Also erstellt das Entwicklungsteam ein Ticket für das Monitoring, da fehlen aber noch Information im Ticket, wieso fehlen da noch Information, ich erkläre es dir, jetzt habe ich ein Ticket mit allen Information, die Sensoren wurden eingebaut, da fehlt aber noch ein Sensor, OK, warum hast du das nicht gleich gesagt, nun ist der Sensor auch eingebaut, aber wo ist der Aktionsplan für die Alarmierung, der kommt später, Hatten wir das mit den Schnittstellen nicht schon mal? Gibt es eine Lösungsidee? Klar: Bau eine **SelfService** Schnittstelle! Im gleichen Zuge, in dem wir die SelfService Schnittstelle gebaut haben, haben wir auch unser solides, jedoch in die Tage gekommenes CACTI durch eine moderne Lösung abgelöst. Von der neuen Lösung versprochen wir uns mehr Möglichkeiten, Dashboards, Graphen, Informationen etc. bedarfsorientiert zu visualisieren, um dadurch schneller und einfacher entstören und planen zu können.



Nachher: Performance Monitoring mit InfluxDB & Grafana

Dank der frühen Planung und der Rahmenbedingungen, die geschaffen wurden, konnten wir auch damit beginnen, einen weiteren SelfService für das automatisierte Monitoring zu bauen. Möglich wurde das u.a. dadurch, dass wir uns externe Unterstützung ins Team geholt haben, die uns den Rücken im Tagesgeschäft frei gehalten hat, so dass die internen Mitarbeiter sich um die Entwicklung der neuen Technologie kümmern konnten. Das klingt alles super und hat uns sehr geholfen, ein Überschuss an Mitarbeitern war aber dennoch nicht zu erkennen. Wir haben einige Themen parallel bearbeitet. Doch wir haben das

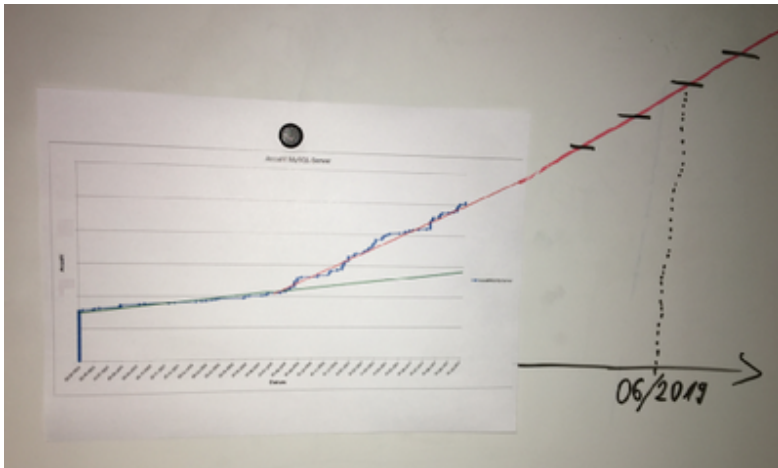
Beste daraus gemacht. Und so kam es z.B., dass wir u.a. einen unserer Datenbank Spezialisten überzeugen konnten, zusammen mit einem Monitoring-Spezialisten und Software-Entwicklern ein neues Performance Monitoring System basierend auf InfluxDB, Telegraph und Grafana zu bauen. Die Arbeiten wurden stark unterstützt von Produkt-Teams, die ihre Ideen und Anforderungen eingebracht haben. Nachdem der erste brauchbare Prototyp stand, haben wir vor der Pilotphase eine weitere „Bier-Session“ durchgeführt. Die Session hat allen Nutzern einen frühen Blick auf das künftige System gegeben und uns gutes Feedback gegeben, welches wir einarbeiten konnten.

Ein neues Performance Monitoring hilft aber nur in Teilen weiter. Es ist leistungsstark, voller Features und kann ohne Hilfe des IT-Betriebs genutzt werden. Jedoch fehlt noch eine Lösung für unser Alarmierungssystem und dessen Aktionspläne. Daher haben wir zusätzlich zusammen mit Software Entwicklern, Qualitätstestern und Administratoren einen komplett neuen Service gebaut, der es erlaubt, aus Informationen aus einer YAML-Datei die Sensoren in der Alarmierung sowie dem Performance Monitoring zu erstellen und zu konfigurieren. Dadurch haben wir den Entwicklungsteams eine Schnittstelle gegeben, die sie aus ihrem Tagesgeschäft kennen, haben die Monitoring Konfiguration zu Code gemacht und versioniert, die manuellen Aufwände reduziert und die Qualität erhöht. Klingt nach einem coolen SelfService? Ist er auch! ☐

Was haben wir mit den Änderungen an den Basistechnologien erreicht?

Wir haben eine Menge Arbeit in die Bereitstellung von SelfServices und Automaten investiert. Dadurch haben wir es geschafft, die aufwändige Abstimmung zwischen Entwicklungsteams und IT-Betrieb zu reduzieren. Wir haben durch Automation und technische Schnittstellen die Qualität in der Umsetzung von Anforderungen erhöht und den Bedarf für Nacharbeiten reduziert. Dadurch haben wir einen wichtigen Betrag zur Beschleunigung der Entwicklungsteams geleistet. Die Aufwände für wiederkehrende Arbeit wurden im IT-Betrieb reduziert, wodurch mehr Zeit für Weiterentwicklung entstanden ist.

Gibt es auch Schattenseiten?



Wachstum DB-Systeme der letzten 2 Jahre inkl.
„Prognose“

Natürlich ist nicht immer alles rosarot, und auch wir haben durch Schmerzen gelernt. So wird z.B. unser SelfService zur automatischen Service Bereitstellung sehr gerne und viel genutzt, was erst einmal sehr positiv ist. Hier ist ein Beispiel, wie sich die Anzahl unserer Datenbanksysteme über die letzten 2 Jahre geändert hat und welches Wachstum wir noch erwarten. Ratet einfach mal wann die automatische Service Bereitstellung verfügbar war □ Eines der verbundenen Probleme ist, dass die Anforderung an alle Komponenten der Plattform (Beispiel: Storage, CPU, RAM, Virtualisierung, ...) deutlich gestiegen sind. Auch die Anforderungen an die für den Betrieb notwendigen Systeme (Beispiel: Datensicherung, Monitoring, Logging, ...) sind davon nicht ausgenommen. Natürlich haben wir mit einem spürbaren Anstieg der Systeme und somit auch mit den Anforderungen gerechnet. Dennoch wurden wir ein Stück vom Erfolg überrascht. Fehlende Ressourcen in der Plattform haben zu Störungen in den Services geführt, was wir nachträglich im laufenden Betrieb mit Schmerzen in allen Teams korrigieren mussten. Auch die steigenden Kosten sowie die Kostenkontrolle für Plattform Ressourcen müssen bei der Einführung und dem Ausbau von SelfServices berücksichtigt werden. Aktuell verrechnen wir die Kosten nicht an die Teams, jedoch stellen wir über Reporting sicher, dass jedes Team weiß, was ein System kostet und welche Kosten für Plattform Ressourcen ein Team produziert.

Kapitel 3: Mindset, Methoden, Prozesse und mehr...

In einer Woche werde ich im dritten und vorerst letzten Kapitel unserer Reise darüber berichten, dass neben der Technologie auch kulturelle und methodische Änderungen notwendig waren. Z.B. war unser Mindset das eines klassischen IT-Betriebs. Das Mindset unserer wichtigsten Kunden, der Entwicklungsteams, war jedoch das einer agilen Software-Entwicklung. Und was ist eigentlich dieses DevOps? Hier gab es also noch etwas zu tun. Abgerundet wird das dritte Kapitel durch ein Fazit, was wir gelernt haben. To be continued ...

Alle drei Kapitel im Überblick



[Kapitel 1: Woher kommen wir? 6 Jahre im Schnelldurchlauf](#)



[Kapitel 2: Basistechnologien, SelfServices & Automation](#)



[Kapitel 3: Mindset, Methoden, Prozesse und mehr...](#)

Vom IT-Betrieb zu Platform Engineering. Ein Reisebericht (1/3)



Es gibt einige Artikel auf unserem Tech Blog, die über unseren Wandel zur Produkt-Organisation berichten. Auch wenn der IT-Betrieb bei solchen Aktivitäten gerne mal vergessen wird, so war es bei unserem Wandel zur Produkt-Organisation nicht der Fall. Als Learning aus anderen Umstellungen und Unternehmen wurde der IT-Betrieb sehr früh in den Prozess einbezogen. Nach gut einem Jahr kann man sagen: Das hat sich bezahlt gemacht!

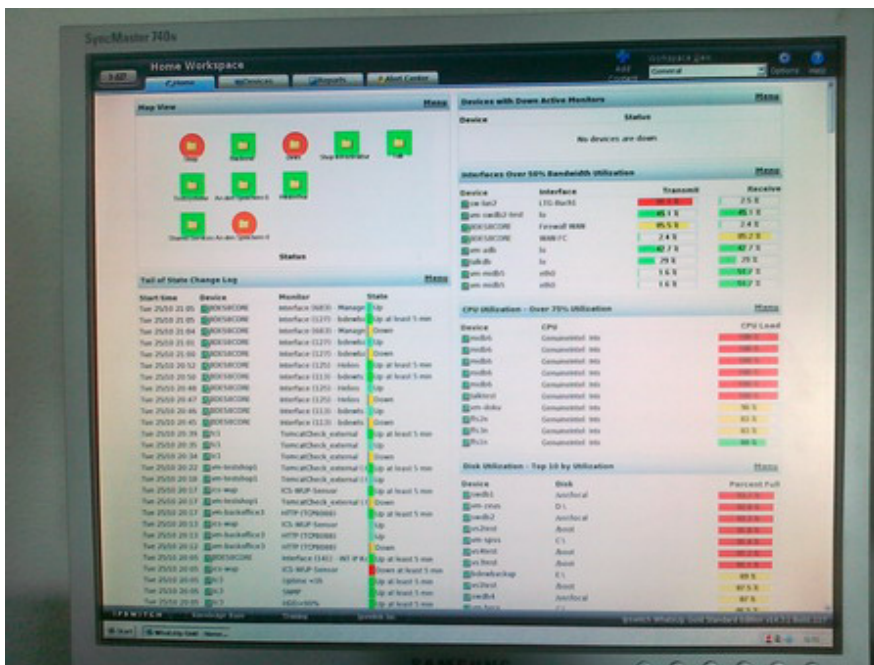
Die folgenden Zeilen sind ein Reisebericht aus dem Blickwinkel von IT-Operations. Woher kommen wir? Warum wollten wir uns verändern? Wie haben wir das gemacht? Was haben wir dabei gelernt? Diese kleine Artikelserien in drei Teilen gibt einen Überblick über die Themenblöcke, deren Hintergründe und Inhalte. Ich gehe dabei jedoch nicht auf jedes einzelne Detail ein. Eventuell folgen diese ebenfalls sehr spannenden Details in späteren Artikeln.

Kapitel 1: Woher kommen wir? 6 Jahre im

Schnelldurchlauf

Die Reise beginnt

Vor rund sieben Jahren haben wir nach und nach den IT-Betrieb aufgebaut. Dazu haben wir über die Zeit Datenbank-, Linux- und Windows-Spezialisten an Bord geholt und zu einem Team formiert. Eines war dabei immer klar: Bleib nah an der eigenen Entwicklung, die unsere eCommerce Software baut. Da die gesamte IT damals noch eine überschaubare Größe hatte und auf einem Flur saß, war das auch recht einfach. Genau wie heute war bereits damals die Stimmung sehr positiv, der Zusammenhalt und die gegenseitige Unterstützung sehr groß.



Monitoring & Alarmierung (2010)

Im Jahr 2010 hatten wir für thalia.de/ch/at lediglich rund 100 Server, die große Monolithen und einige wenige Services beherbergten. Eine vergleichsweise überschaubare Systemlandschaft mit Linux, Tomcat, JAVA, Apache, MySQL. Schon damals hatten wir kein eigenes Datacenter, sondern haben Datacenter as a Service (DCaaS) genutzt. Wir waren nicht darauf aus, unsere Zeit primär mit einem Schraubendreher im Rechenzentrum zu verbringen. Uns war immer klar, dass wir als Team den größten Mehrwert nah an den vom Kunden genutzten Services bringen können. Server in ein Rack einschrauben und Betriebssysteme installieren können viele Dienstleister und Cloudanbieter besser als wir.

Unsere Aufgabe war es, den Betrieb der Systeme und der Services rund um die Uhr sicherzustellen. In guter IT-Betriebs-Tradition haben wir durch Prozesse, Standardisierung und Professionalisierung einen guten Beitrag zur Verbesserung der Verfügbarkeit unseres Gesamtsystems und somit zum Umsatz geleistet. Auch wenn längst nicht alles perfekt war, so machten wir doch gute Fortschritte, und wir wurden stetig besser – was man u.a. auch an der Verfügbarkeit der Shop Systeme erkennen konnte.

Insourcing der Entwicklung & SOA-Architektur

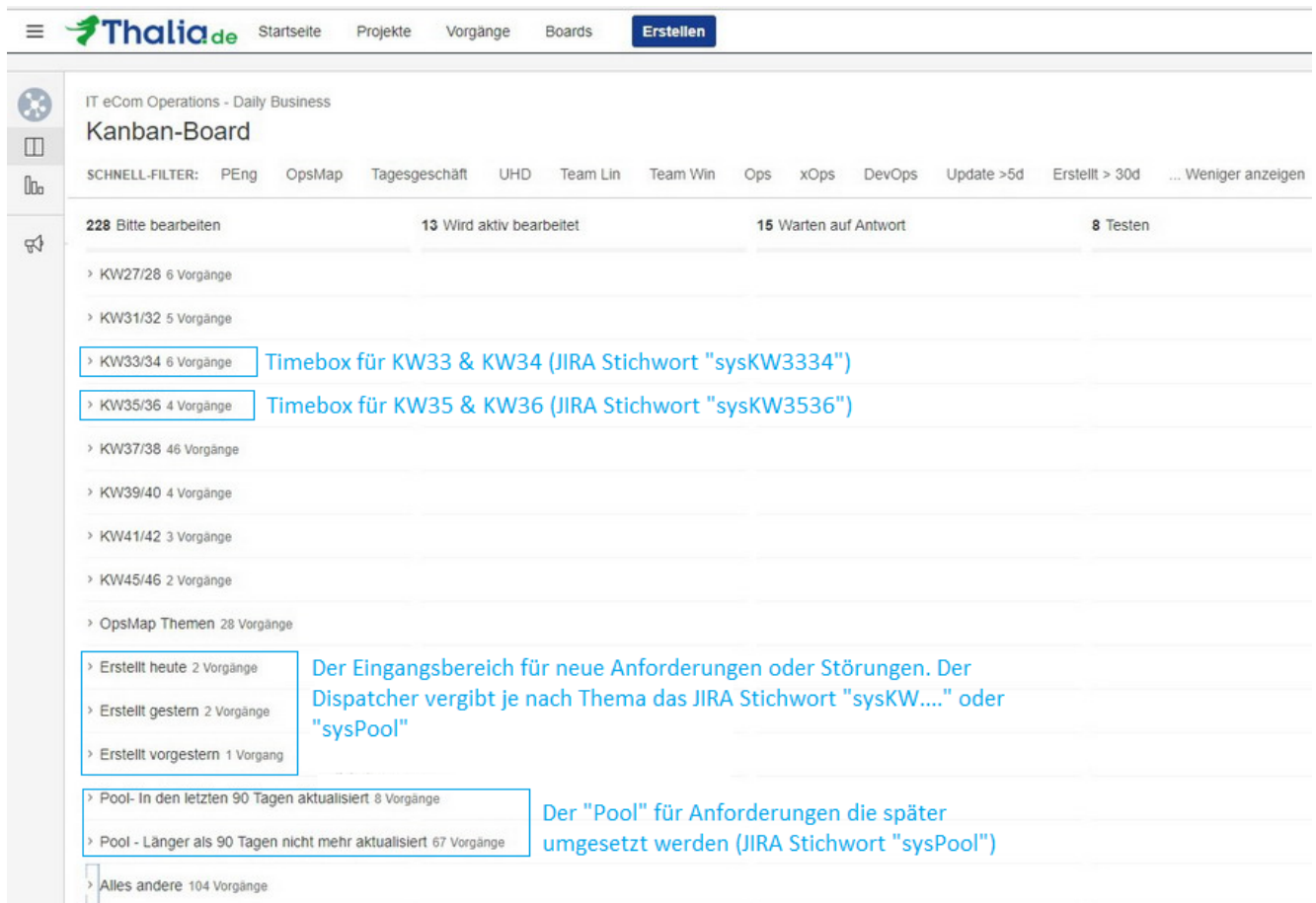
Doch nicht nur der Betrieb entwickelte sich weiter. Auch die Software-Entwicklung machte große und tolle Fortschritte. Damals wurde Software für unser eCommerce System zu einem großen Teil extern entwickelt. Diese Entwicklung wurde ins Haus geholt (hmm, das klingt gerade leichter als es wirklich ist – bestimmt schiebt einer der Kollegen mal einen Artikel dazu ☐). In diesem Zusammenhang wurde auch das Deployment, welches zuvor vom externen Dienstleister durchgeführt wurde, in den noch jungen IT-Betrieb übergeben. Ungefähr zur gleichen Zeit hatte die Software-Entwicklung eine echt gute Idee: [SOA-Architektur](#)! Tötet die Monolithen! Nach einem kurzen Termin mit unserem Software-Architekten, der die Idee vorstellte, sagten wir völlig ahnungslos dafür aber um so selbstbewusster: Klar, kein Problem! Und so kam was kommen musste. Der IT-Betrieb wurde von Anforderungen überrollt und konnte die Software-Entwicklung nicht bedarfsgerecht bedienen. Hinter Prozessen und dem Ticketsystem haben wir versucht, in Deckung zu gehen und die Anforderungen zu verstehen, zu priorisieren und sequenziell abzuarbeiten. Waren wir damals schnell? Nun ja, es gab da durchaus Verbesserungspotential.



Die gesamte Schnittstelle zwischen der Entwicklung und dem IT-Betrieb hatte einen massiven Overhead und verursachte hohe Reibungsverluste. Klar waren wir (Entwicklung & IT-Betrieb) immer noch gute Kollegen, haben uns geholfen wo es ging und haben freudig (inzwischen auf zwei Etagen) zusammengearbeitet. Events unterschiedlichster Art wie z.B. den Sysadminday haben wir gerne gefeiert und machen das immer noch. Der Stresspegel auf beiden Seiten stieg jedoch permanent an. Jeder im IT-Betrieb hatte im Ticket System seine 100 Vorgänge und mehr zu tun als er schaffen konnte. Transparenz für den Anforderer und für den Bearbeiter gab es trotz Ticket System jedoch schon lange nicht mehr. Klares Priorisieren der Masse war nahezu unmöglich. Zeit für eine Veränderung!

IT-Operations meets Kanban

Überrollt von Anforderungen aus der Entwicklung sowie aus dem IT-Betrieb mussten wir mehr Struktur und Transparenz in unsere Arbeit bringen. Wir waren ein klassischer IT-Betrieb und hörten zum ersten Mal etwas von „Kanban“. Hmm, was ist das? Was kann das? Sieht erst einmal spannend aus. Da wir damals schon JIRA mit dem AGILE Plugin im Einsatz hatten, haben wir einfach mal unsere Tickets in einem Kanban Board anzeigen lassen. Toll, 500 Tickets auf einem Board. Das soll jetzt helfen? Es hat viele Anläufe gebraucht, bis wir es tatsächlich geschafft haben, ein Kanban Board zu entwickeln, welches uns als IT-Betrieb den Fokus auf die „jetzt“ wirklich wichtigen Tickets gegeben und dem Anforderer ein Feedback gegeben hat, wann ein Ticket aller Wahrscheinlichkeit nach umgesetzt wird. Von der Grundidee haben wir unsere Themen sichtbar gemacht und Timeboxes von 2 Wochen eingeführt. Jedes Ticket wurde einer Timebox zugewiesen oder gezielt in den Pool gelegt, wenn es später bearbeitet werden sollte. Auch Tickets unbearbeitet jedoch kommentiert schließen kann sinnvoll sein, führt aber gerade am Anfang zu Diskussionen.



Damit das Board funktioniert, haben wir einen Filter auf alle offenen Tickets des IT-Betriebs gelegt. Das Board selber hat drei Kategorien von Swimlanes: (1) **Timeboxen** für die Bearbeitung, (2) den **Eingangsbereich** und (3) den **Pool**.

Eine **Timebox** dauert immer zwei Wochen. Tickets in einer Timebox werden mit dem JIRA Stichwort „sysKWxxxx“ getagged wobei „xxxx“ die jeweiligen zwei Kalenderwochen angeben (Beispiel sysKW0102, sysKW0304, sysKW0506, ...). Wird ein Ticket z.B. mit dem Stichwort „sysKW3334“ versehen, so erscheint das Ticket in der Swimlane der Timebox „KW33/34“. Diese Timebox gibt dem Bearbeiter die Chance sich nur auf Tickets in der Timebox zu fokussieren (und nicht auf alle 500 offenen Tickets). Der Anforderer kann am Stichwort erkennen, wann sein Ticket gemäß Planung bearbeitet werden soll.

Neue Anforderungen oder Störungen landen automatisch im **Eingangsbereich**. Ein Dispatcher entscheidet nun anhand von Prioritäten (ggf. sind hierfür Rückfragen beim Anforderer notwendig), in welcher Timebox ein Ticket planmäßig abgearbeitet werden soll. Um ein Ticket in eine Timebox zu packen, muss er lediglich das JIRA Stichwort „sysKWxxxx“ vergeben.

Ist die Priorität einer Anforderung oder eine Störung aktuell nicht besonders

hoch, so wird das zugehörige Ticket im **Pool** abgelegt. Dazu vergibt das Dispatcher das Stichwort „sysPool“. Auch ist es der Dispatcher, der immer wieder den Pool prüft, ob Tickets aufgrund veränderter Prioritäten in eine Timebox geschoben werden sollten. Der Anforderer wird automatisch per Mail über die Vergabe von JIRA Stichwörtern und somit den Planungsstatus des Tickets informiert. Ist er mit der Timebox oder dem Pool nicht einverstanden, so kann er beim Dispatcher die Priorität besprechen und ggf. verändern.

Was haben wir bis hier erreicht?

Wir hatten nun eine gute Übersicht über alle Themen und in welchem Status die Themen waren. Wir konnten Themen priorisieren, abmelden oder auf später verschieben. So konnten wir die wichtigen Themen früher machen und die vermeintlich unwichtigeren Themen später oder gar nicht. Im Team hatten wir mit einem Mal Struktur in den Themen. Die Team-Mitglieder hatten nicht mehr den Druck, alles auf einmal machen zu müssen. Sie hatten einen klaren Blick auf eine handhabbare Anzahl an Tickets. Dadurch konnten sie fokussierter an den wichtigen Themen arbeiten.

Ergebnis: Die Zufriedenheit bei Anforderer und Bearbeiter stieg. Die Zahl der Eskalationen reduzierte sich. Ein großer Schritt nach vorne!

Warum mussten wir uns dennoch weiterentwickeln?

Waren wir jetzt schnell genug? Nein! Auf keinen Fall!

Per Prozessdefinition musste ein virtueller Server, der in Produktion eingesetzt werden sollte, rund acht Wochen vor Deployment angemeldet werden (3 Wochen Planung, 1 Woche bauen, 4 Wochen die neue Software testen). Die echte Arbeitszeit in den acht Wochen lag jedoch nur bei ~3PT. Ein Großteil der Zeit während der acht Wochen ging für die Terminfindung, Warten, Abstimmung, Rückfragen, ... bei den beteiligten Teams drauf. Nicht jeder fand das super. Auch wurden die Aufgaben (insbesondere aus dem Tagesgeschäft) immer mehr. Wir hatten kaum Zeit für Innovation und Weiterentwicklung von Themen aus dem

Bereich IT-Betrieb. Wir waren zwar entspannter mit einem besseren Blick – jedoch immer noch Getriebene.

Die Reise geht also weiter ...

Kapitel 2: Basistechnologien, SelfServices & Automation

In einer Woche werde ich im zweiten Kapitel unserer Reise darüber berichten, was wir im Bereich der Technologien gemacht haben. Wie konnten wir Werkzeuge nutzen, um uns zu beschleunigen und die Schnittstellen zu den Produkt-Teams genauer definieren? To be continued ...

Alle drei Kapitel im Überblick



[Kapitel 1: Woher kommen wir? 6 Jahre im Schnelldurchlauf](#)



[Kapitel 2: Basistechnologien, SelfServices & Automation](#)



[Kapitel 3: Mindset, Methoden, Prozesse und mehr...](#)

Wissenskalender

Für die Vorweihnachtszeit hat sich Thalia das Thema „Wissensmanagement“ in den Fokus gelegt.

Die Idee: Jeder Mitarbeiter kann pro Tag ein Thema aus dem Alltag mitbringen und vorstellen. Egal, ob es ein Tool, Vorgehen, Pattern, Library, Metrik etc ist. Mein Thema für den 12. Dezember 2017 waren „Agile Werte“. Hierfür habe ich zunächst unsere 4 oberen Prinzipien aus dem Agilen Manifest vorgestellt. Anschließend habe ich daraus die Agilen Werten Fokus, Mut, Respekt, Offenheit und Commitment hergeleitet. Andere Themen wurden bisher aus den Bereichen DevOps, Qualitätsmanagement, Architektur und Agile Methoden vorgestellt.

Der Anreiz: Hinter jedem Thema / Türchen verbirgt sich auch ein Preis. In unserem Falle sind es Gutscheine für den Shop („Drink your own Champagne!“).

Habt ihr bereits Erfahrungen mit solchen Tools ? Was haltet ihr davon ?

Wissens ADVENTSKALENDER



* GRÄUE ZEHL = WOCHENENDEN

www.kalender.com

POST-IT

KLEBEN

DRUCKEN

Verf. Nr. 1128

Bestell. Nr. 110 0000 00

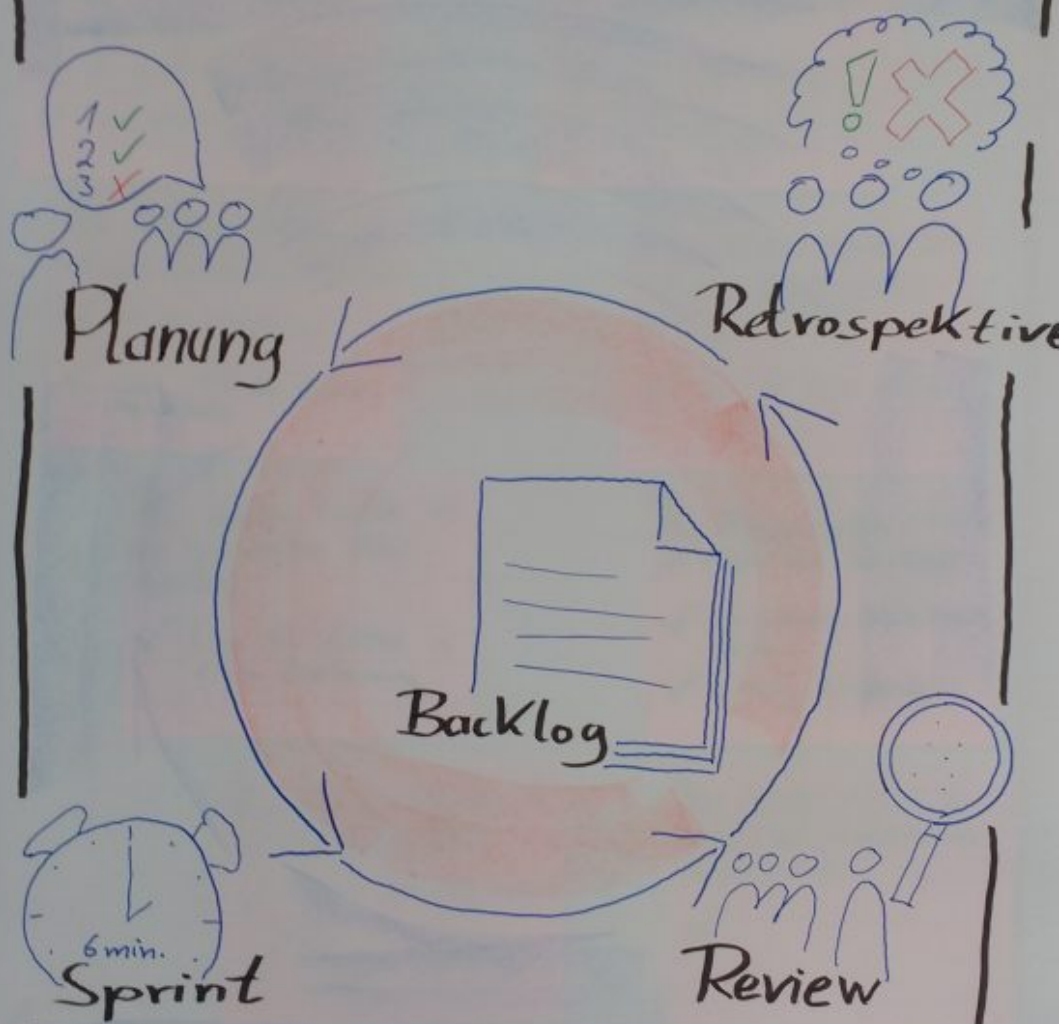
OVSoftware Scrum Workshop

Für die Mitarbeiter von Thalia gab es am 13.09.2017 ein besonderes Angebot:

„Scrum“ in einem Workshop ohne Software-Entwicklung kennenlernen

Das Unternehmen OVSoftware bietet ein Scrum - Spiel an, zu dem die Teilnehmer einfach ein gewisses handwerkliches Geschick mitbringen müssen.

Scrum Regeln



Die

Workshop-Leiter gaben zu Beginn eine kurze Einführung in die Entwicklungsmethode. Anschließend wurde die Vision für das zu entwickelnde Produkt vorgestellt. Für den Anreiz untereinander wurden die Teilnehmer in 5 Teams aufgeteilt.



Jedes Team

erhielt einen Arbeitsplatz. Nach Vorstellung der ersten „User Storys“ ging jedes Team mit einem Commitment in den ersten Sprint. Nach 6 Minuten wurde die Arbeit eingestellt.



Alle Teams traten zum ersten Review an und die Workshop Leiter bewerteten die Ergebnisse hinsichtlich zu erfüllender Abnahmekriterien. Nach einer kurzen Retrospektive wurden alle Teams wieder zu einem Commitment gebeten.



Getreu dem agilen Ansatz „Inspect & Adapt“ wurden weitere Sprints im

Workshop durchgeführt. Am Ende ging jedes Team mit einem vollendeten Prototyp und viel Spaß an einer neuen Arbeitsmethode aus dem Workshop.

Solltet ihr mir bis hierhin gefolgt sein und Euch fragen: „Worum geht`s hier eigentlich?“

...Ich habe bewusst hier das Produkt nicht näher vorgestellt, um noch Appetit für weitere Workshops anzuregen.



Münster macht Agil

Wow, was für ein schöner Start!

Ca. 50 Interessierte aus Münster und Umgebung sorgten bei regionalem Wetter für einen Speicher 6 gefüllt mit Gedanken, Austausch und interessanten Gesprächen rund um das Thema „Agile“, insbesondere „Scrum“.



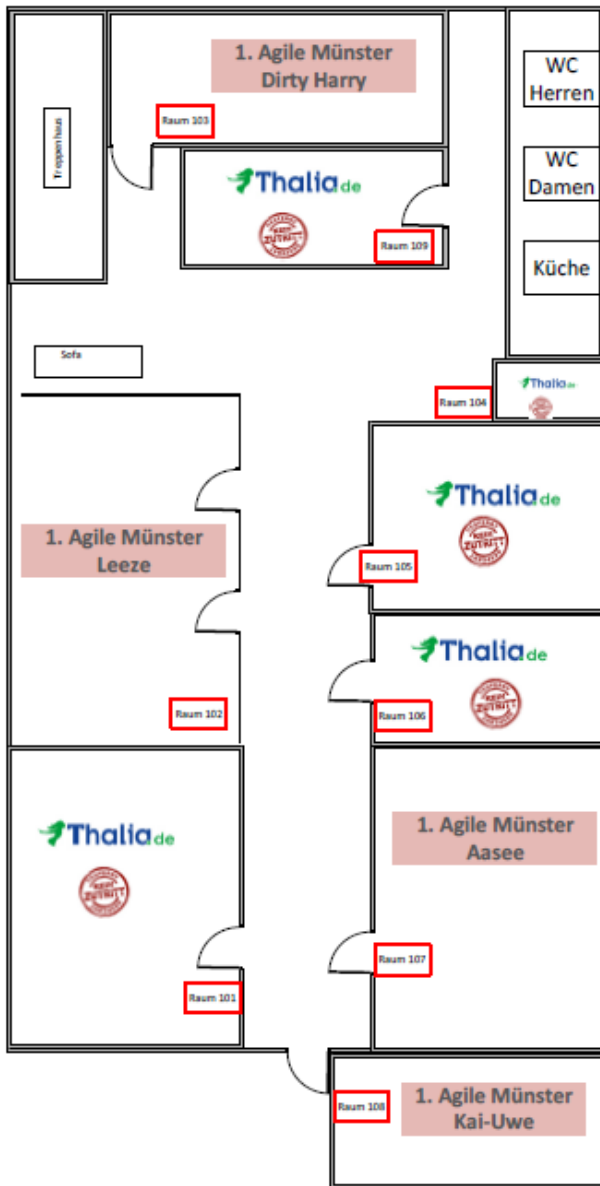
Um 19 Uhr gab es zunächst ein „social warmup“ bei italienischem Heißgebäck und Kaltgetränken.

Patrick und Matthias haben nach einer kurzen Begrüßung die Teilnehmer mit Kennenlernfragen rund um das Thema Agilität bunt gewürfelt und anschließend den Marketplace für die Themen und Fragen eröffnet.





1. Agile Münster Raumplan

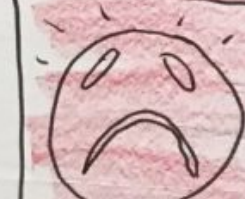


Von Fragen wie „**Scrum vs Kanban**“ beziehungsweise „**Was mache ich mit Daily Business in geschützten Sprints?**“ über **Best Practices für die Ausbildung von Product Ownern** gab es sogar eine DIY-Session mit Lego zum Thema „**Agile Transition - vorher, nachher**“.



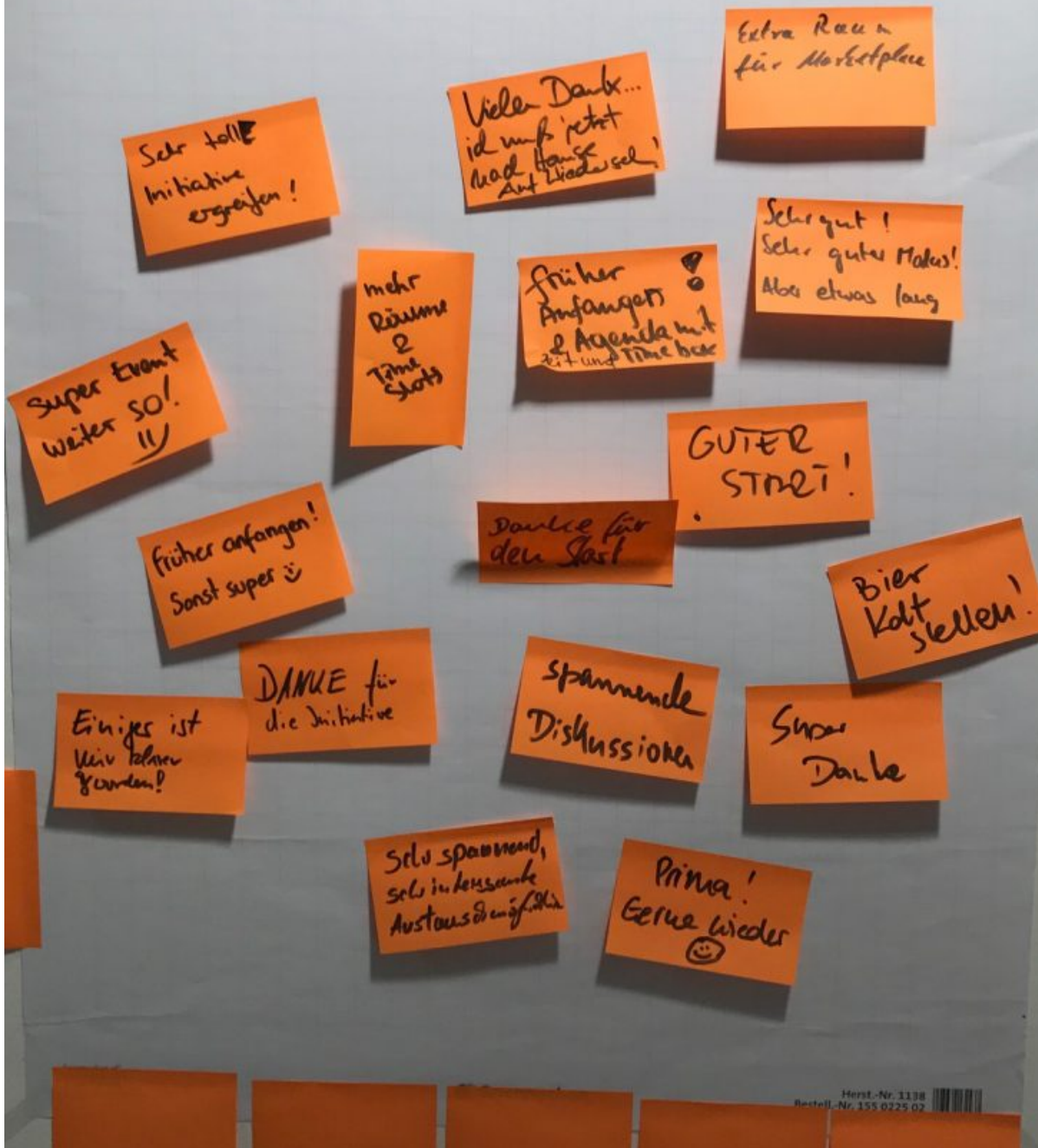
Um 21:45 Uhr schlossen Matthias und Patrick den Marketplace und damit auch den Open Space mit einer Retrospektive zum ersten Abend.

Wie fühlst Du Dich?



Feedback

Wie war es für Euch?



Ein Dank gilt neben dem Sponsor Thalia auch den freiwilligen Helfern, die auch nach Abreise der Gäste noch halfen, die Büros wieder herzustellen.

Wir freuen uns schon auf das nächste Mal, wenn es heißt:

Münster macht Agil

Dieser Beitrag entstand mit:



Matthias Hochschulz Agile Coach /
Scrum Master

Happy SysAdmin Day!

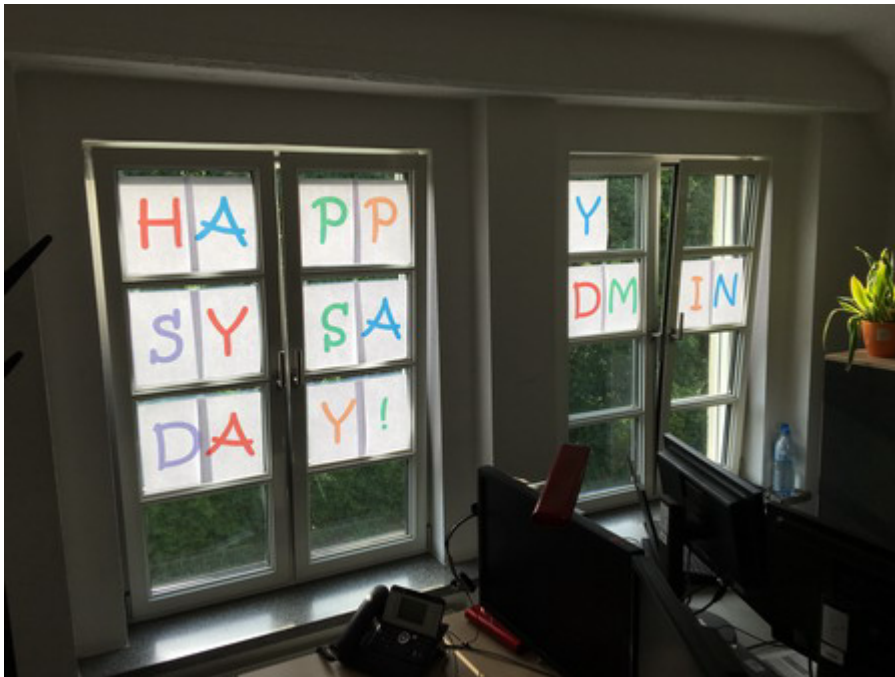
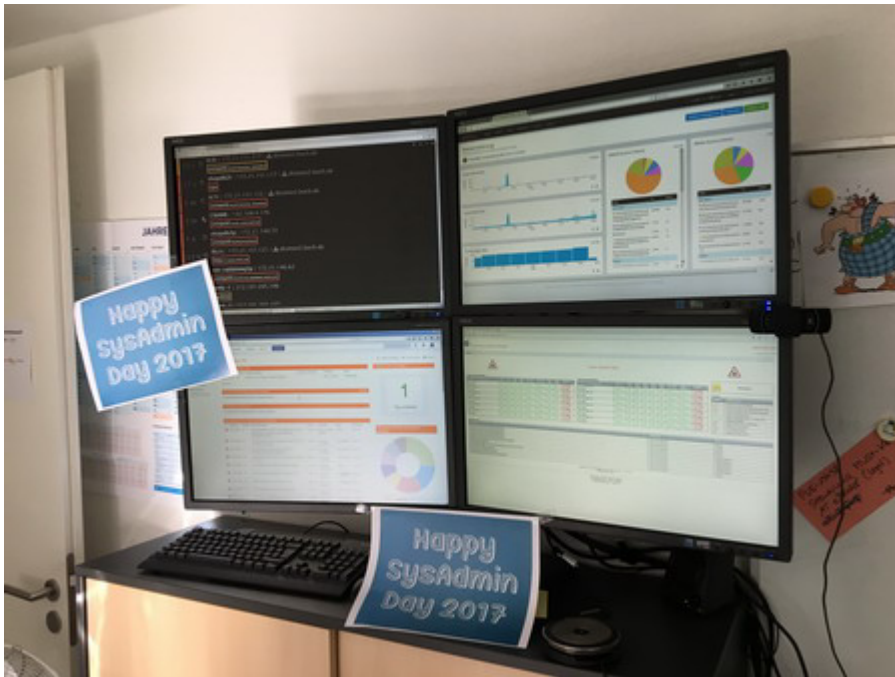
Es ist der letzte Freitag im Juli. Die Flure sind mit Danksagungen tapeziert. Die Leute sind noch freundlicher zu den System-Administratoren. In der Kaffee-Küche stehen Süßigkeiten.

Das kann nur eines bedeuten.... **Es ist SysAdmin Day!**

Vielen Dank allen System-Administratoren für die unermüdliche Arbeit in und für unsere Crossfunctional Teams.







Btw: Liebe Entwickler, habt noch ein wenig Geduld. Am 256 Tag des Jahres ist der „[Day of the Programmer](#)„. Auch die QA kommt nicht zu kurz. Am 9.11. ist „[World Quality Day](#)„

Berliner Team lernt fliegen.



Wir in Berlin wollen hoch hinaus und suchen ständig nach neuen Ideen die hochgesteckten Ziele zu erreichen. Eine vollkommen neue Variante hat das Team bei einem gemeinsamen Besuch des [Berliner Jump House](#) ausprobiert. Wir hatten alle extrem viel Spaß und konnten völlig neue Höhenflüge erreichen.

Ich@Thalia

Am Anfang

Im Frühjahr des letzten Jahres (2016) habe ich meine Tätigkeiten als *System Engineer* bei Thalia am Standort Münster begonnen. Jeder — wirklich jeder in meinem näheren Umfeld — kannte bereits den Namen bzw. die Marke Thalia und konnte damit zumindest eine Verbindung zum (lokalen) Buchhandel herstellen. *„Aber wie in aller Welt soll denn meine Tätigkeit bei einem Buchhändler aussehen ... ?!“* „, habe ich mich — sowie mein Umfeld sicherlich auch — zurecht gefragt. Zum damaligen Zeitpunkt konnte ich diese Frage nicht wirklich beantworten, ließ mich aber gern auf dieses Abenteuer und neue Erfahrungen ein.

Ich gestehe, dass ich keine Ahnung davon hatte, was mich in den kommenden

Wochen und Monaten erwarten würde, aber hey, ist das nicht immer so, wenn man(n) etwas Neues beginnt ?! Gleichzeitig konnte (und ein wenig auch „wollte“) ich nicht wahrhaben, dass Thalia bereits zu diesem Zeitpunkt ein durchaus gewachsenes eCommerce Business — neben dem klassischen Buchhandel — erschaffen hatte und dieses erfolgreich betreibt.

Plötzlich mittendrin, statt nur dabei

Niemals auf der Stelle zu treten liegt mir quasi schon von Berufs wegen (als sog. Externer Kollege) im Blut. Diesen ständigen Fortschritt sowie die Weiterentwicklung bereits existierender Produkte sind auch nach mehr als einem Jahr mein ständiger Begleiter innerhalb unseres/unserer Teams.

[Wikipedia](#) bezeichnet *„ein Team als einen Zusammenschluss mehrerer Personen zur Lösung einer bestimmten Aufgabe oder zur Erreichung eines bestimmten Ziels ...“* (Quelle: [Wikipedia: Team](#)). Soviel zur Definition, doch mitten in ein bereits existierendes Team integriert zu werden, wahrgenommen zu werden, Aufgaben zu erledigen, Kritisch zu hinterfragen, „Tekki-Witze“ auf eine Pinnwand zu kritzeln, um sich sowie alle anderen zum Lachen zu bringen, Fehler zu machen, Spaß daran zu haben Neues auszuprobieren, etc. sind meine Gründe, warum ich meine Tätigkeit hier gern ausübe. Neben den sog. harten Fakten, die ich eben beschrieben habe, sind es jedoch immer die Menschen selbst, die für mich den deutlichsten Unterschied machen. Ich zitiere an dieser Stelle gern einen Satz, der mir selbst schon häufig um die Ohren geflattert ist und den ich nur doppelt und dreifach unterstreichen kann: „... Es sind die Kollegen und Kolleginnen, die für mich den Unterschied machen“. Bingo, voll ins Schwarze ☐

Spielplatz auf 12 Uhr

Achtung Spoiler-Alarm: Wer glaubt, dass wir für unsere eCommerce Plattform nur längst überfällige Hardware und Low-Cost Komponenten einsetzen, dem kann ich hier nur folgende Buzzwords anbieten: [F5](#), [Palo Alto](#), [VMWare](#), [HP](#), [IBM](#), [Graylog](#), [Puppet](#) & Co sind nur ein paar Komponenten, die bei uns im täglichen Einsatz sind.

Das Thema „Datenverarbeitung in Echtzeit“ steht in unserer Liste der ToDo's schon länger ganz oben. Daraus erwachsen für unsere IT neue

Herausforderungen, um die automatische Speicherung von Daten, ihre Verarbeitung und Analyse schnell und übersichtlich zu ermöglichen. [Graylog](#) ist ein Open-Source-Projekt für das Log-Management von Daten jeglichen Formats. Auf der Graylog-Plattform werden sowohl strukturierte als auch unstrukturierte Log-Daten von verschiedenen Servern und Anwendungen gesammelt, indiziert und analysiert. Ebenso möglich ist die Verwendung von Komponenten für Metadaten wie MongoDB oder Elasticsearch als Log-Daten-Speicher und Text-Suche. Auch hier stehen Dashboards zur Verfügung, denen Analyse- und Suchresultate als Widgets zugefügt werden können. Die Suche sowie die grafische Visualisierung sind somit ohne Vorkenntnisse oder lange Schulungen möglich.

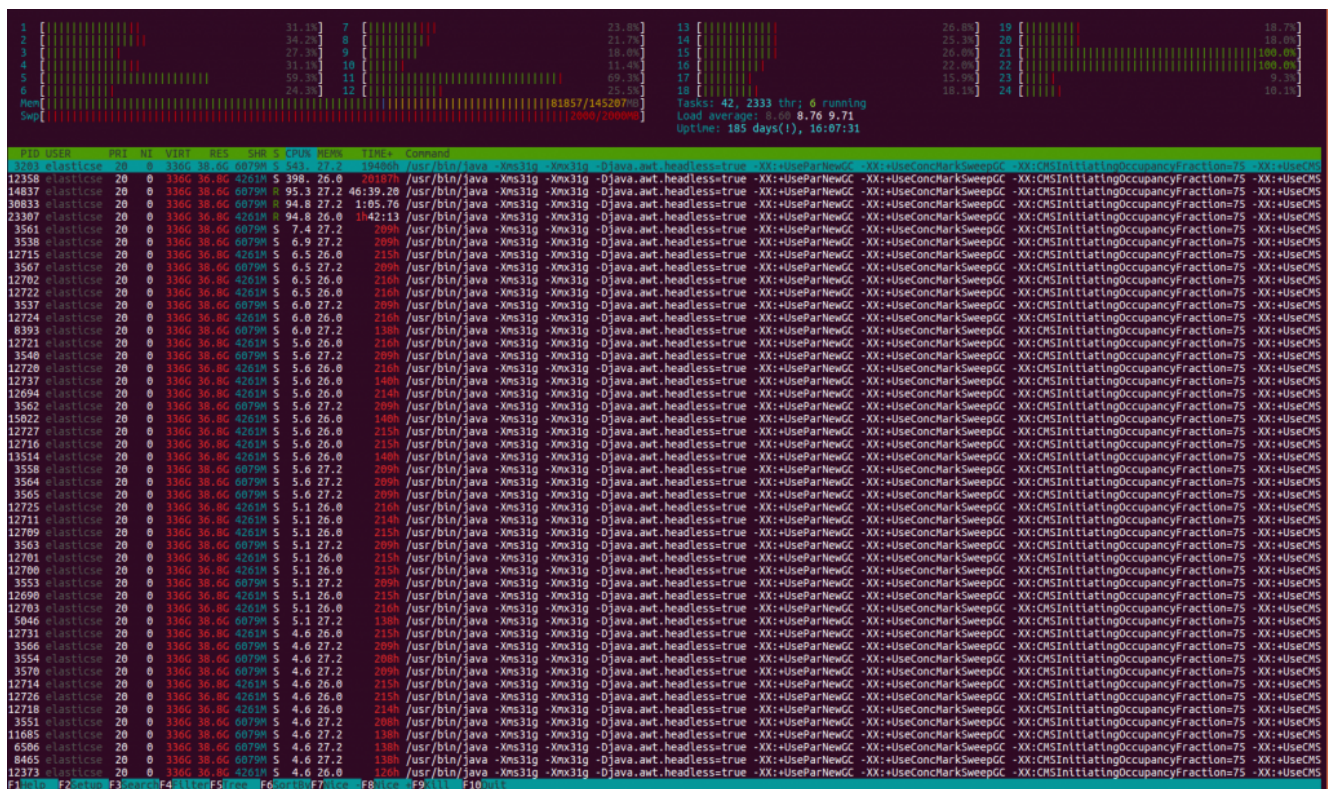


Bild 1: Einer der neuen LogServer auf Basis von Graylog kurz vor dem GoLive

Hand auf Herz: Wer hat sich nicht schon mal wie ein Kleinkind an Weihnachten gefreut, wenn plötzlich klar wird, dass die nagelneue, vor CPU und RAM strotzende Hardware betriebsbereit dasteht und direkt einem kleinen aber feinen Lasttest unterzogen wird. Jeder (in dieser Branche). Und ja, wir sind große Kinder auf einem Spielplatz, aber genau dafür ist dieser ja auch gedacht. Zum Lernen.

Bin ich richtig hier ?

Kurz und schmerzlos: Ja, absolut.

Meetup@Thalia: Vortrag Johannes Mainusch: „Otto.de - wie die Titanic den Eisberg verfehlte“

Am Donnerstag war es endlich so weit, wir traten aus dem Schatten der internen Fachvorträge ins Rampenlicht der offenen MeetUp Kultur.

Die offene Kultur war für diese erste Veranstaltung auch prägend. Nicht nur, dass sie inmitten unserer Räumlichkeiten stattfand, sie war auch ein Beispiel für die enge Zusammenarbeit und Vernetzung der Branche.

So konnten wir Johannes Mainusch dafür gewinnen seinen Vortrag „Warum die Titanic den Eisberg verfehlte“ in unserem Haus erneut zu präsentieren. Der Vortrag konnte bereits im Rahmen der Keynote der Continuous Lifecycle Konferenz begeistern.

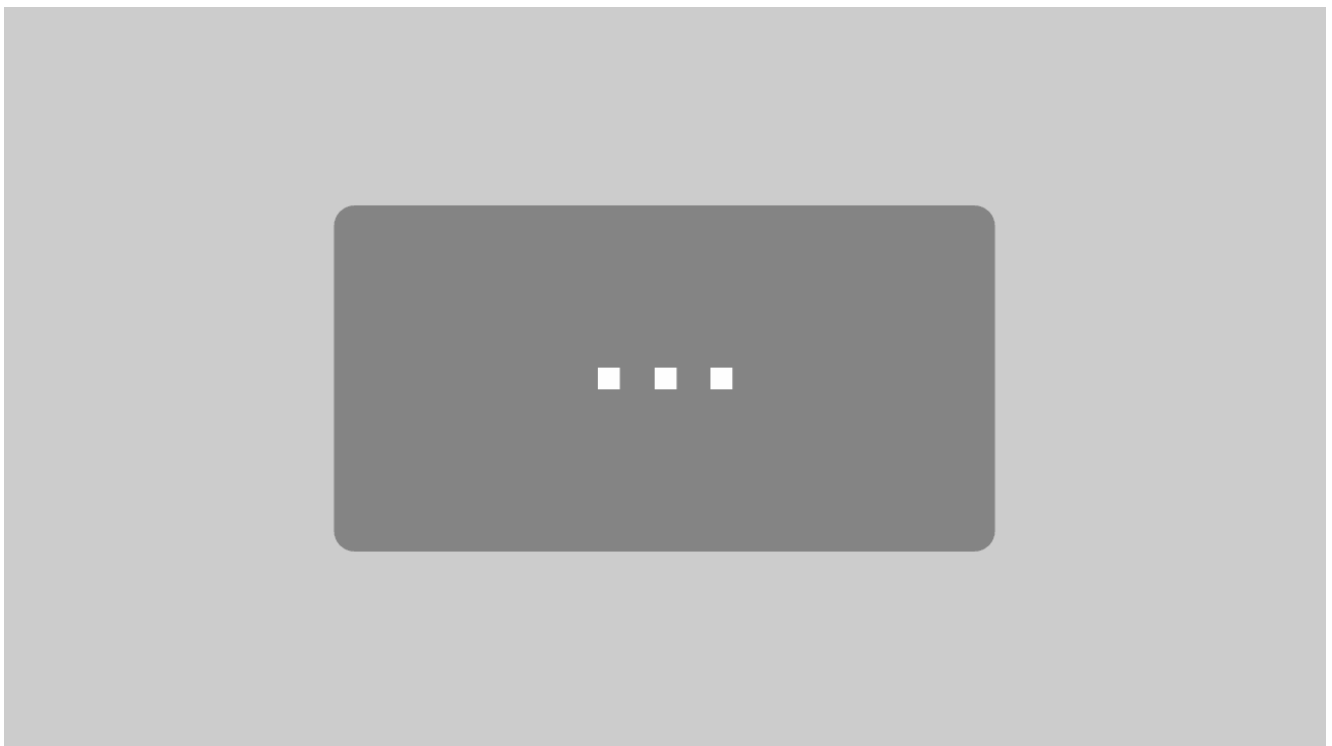


Vortrag durch Johannes Mainusch

Der Vortrag wurde neben den Berliner Kollegen, auch von Kollegen aus Münster

und Hagen, sowie einigen externen Gästen besucht. In rund zwei Stunden bekamen wir so einen spannenden Einblick, wie otto.de die langsame und schwer veränderbare Plattform modernisieren konnte. Aus einer 10 Jahre alten Intershop Installation mit mehr als 750.000 modifizierten Code Zeilen wurde ein Musterbeispiel für agile Entwicklung. Firmen wie Quelle und Neckermann waren unter ähnlichen Bedingungen in der Insolvenz gelandet. Otto.de konnte dagegen mit dem 2012 gestarteten Lhotse- Projekt zu einer der größten E-Commerce Plattformen in Europa wachsen. Welche Entscheidungen letztendlich dafür notwendig waren und was ein Unternehmen befähigt mehr als 340 Releases pro Woche Live zu bringen, schilderte Johannes Mainusch anschaulich und untermalt von vielen Beispielen.

Wer Interesse hat, kann sich den vollen Keynote Beitrag auch bei Youtube ansehen:



Mit dem Laden des Videos akzeptieren Sie die Datenschutzerklärung von YouTube.

[Mehr erfahren](#)

[Video laden](#)

☐ YouTube immer entsperren



Johannes Mainusch

ist seit April 2016 selbstständig bei der kommitment GmbH & Co. KG tätig. Vorherige Stationen waren unter anderem Lufthansa, Xing, Otto.de bis hin zur E-Post. Author des Titels „[Scrum mit User Stories](#)“



Start mit einem Pizza Buffet



Vortrag durch Johannes Mainusch



Diskussionsrunde nach dem Vortrag

