

DOUG SEVEN

Something can be learned in the course of observing things

[HOME](#)
[THINGLABS](#)
[AZURE IOT](#)
[IOT WORKSHOP](#)
[KNIGHTMARE](#)
[ABOUT](#)
[POSTS](#)
[COMMENTS](#)
You are here: [Home](#) / [DevOps](#) / [Knightmare: A DevOps Cautionary Tale](#)

Knightmare: A DevOps Cautionary Tale

APRIL 17, 2014 BY D7 64 COMMENTS



I was speaking at a conference last year on the topics of DevOps, Configuration as Code, and Continuous Delivery and used the following story to demonstrate the importance making deployments fully automated and repeatable as part of a DevOps/Continuous Delivery initiative. Since that conference I have been asked by several people to share the story through my blog. This story is true – this really happened. This is my telling of the story based on what I have read (I was not involved in this).


This is the story of how [a company with nearly \\$400 million in assets went bankrupt in 45-minutes because of a failed deployment](#).

Background

Knight Capital Group is an American global financial services firm engaging in [market making, electronic execution, and institutional sales and trading](#). In 2012 Knight was the [largest trader in US](#) equities with market share of around 17% on each the NYSE and NASDAQ. Knight's Electronic Trading Group (ETG) managed an average daily trading volume of more than 3.3 billion trades daily, [trading over 21 billion dollars.. daily](#). That's no joke!


On July 31, 2012 Knight had approximately \$365 million in cash and equivalents.

The NYSE was planning to launch a new [Retail Liquidity Program](#)  program meant to provide improved pricing to retail investors through retail brokers, like Knight on August 1, 2012. In preparation for this event Knight updated their automated, high-speed, algorithmic router that send orders into the market for execution known as SMARS. One of the core functions of SMARS is to receive orders from other components of Knight's trading platform ("parent" orders) and then send one or more "child" orders out for execution. In other words, [SMARS would receive large orders from the trading platform and break them up into multiple smaller orders in order to find a buyer/seller match for the volume of shares](#). The larger the parent order, the more child orders would be generated. 

The update to [SMARS was intended to replace old, unused code referred to as "Power Peg" – functionality that Knight hadn't used in 8-years](#) (why code that had been dead for 8-years was still present in the code base is a mystery, but that's not the point). [The code that that was updated repurposed an old flag](#)  was used to activate the Power Peg functionality. The code was thoroughly tested and proven to work correctly and reliably. What could possibly go wrong?

What Could Possibly Go Wrong? Indeed!

Between July 27, 2012 and July 31, 2012 [it manually deployed the new software to a limited number of servers per day – eight \(8\) servers in all](#). This is what the [SEC filing](#) says about the manual deployment process (BTW – if there is an SEC filing about your deployment something may have gone terribly wrong).

g the deployment of the new code, however, one of Knight's technicians did not copy the new code to one of the eight SMARS computer servers. Knight did not have a second technician review this deployment and no one at Knight realized that the Power Peg code had not been removed from the eighth server, nor the new RLP code added. Knight had no written procedures that required such a review.
SEC Filing | Release No. 70694 | October 16, 2013

At 9:30 AM Eastern Time on August 1, 2012 the markets opened and Knight began processing orders from broker-dealers on behalf of their customers for the new Retail Liquidity Program. The seven (7) servers that had the correct SMARS deployment began processing these orders correctly. Orders sent to the eighth server triggered the supposable repurposed flag and brought back from the dead the old Power Peg code.



Attack of the Killer Code Zombies

It's important to understand what the "dead" Power Peg code was meant to do. This functionality was meant to count the shares bought/sold against a parent order as child orders were executed. Power Peg would instruct the system to stop routing child orders once the parent order was fulfilled. Basically, Power Peg would keep track of the child orders and stop them once the parent order was completed. In 2005 Knight moved this cumulative tracking functionality to an earlier stage in the code execution (thus removing the count tracking from the Power Peg functionality).



When the Power Peg flag on the eighth server was activated the Power Peg functionality began routing child orders for execution, but wasn't tracking the amount of shares against the parent order – somewhat like an endless loop.

45 Minutes of Hell

Imagine what would happen if you had a system capable of sending automated, high-speed orders into the market without any tracking to see if enough orders had been executed. Yes, it was that bad.

When the market opened at 9:30 AM people quickly knew something was wrong. By 9:31 AM it was evident to many people on Wall Street that something serious was happening. The market was being flooded with orders out of the ordinary for regular trading volumes on certain stocks. By 9:32 AM many people on Wall Street were wondering why it hadn't stopped. This was an eternity in high-speed trading terms. Why hadn't someone hit the kill-switch on whatever system was doing this? As it turns out there was no kill switch. During the first 45-minutes of trading Knight's executions constituted more than 50% of the trading volume, driving certain stocks up over 10% of their value. As a result other stocks decreased in value in response to the erroneous trades.



To make things worse, Knight's system began sending automated email messages earlier in the day – as early as 8:01 AM (when SMARS had processed orders eligible for pre-market trading). The email messages referenced SMARS and identified an error as "Power Peg disabled." Between 8:01 AM and 9:30 AM there were 97 of these emails sent to Knight personnel. Of course these emails were not designed as system alerts and therefore no one looked at them right away. Oops.



During the 45-minutes of Hell that Knight experienced they attempted several counter measures to try and stop the erroneous trades. There was no kill-switch (and no documented procedures for how to react) so they were left trying to diagnose the issue in a live trading environment where 8 billion shares were being traded every minute. Since they were unable to determine what was causing the erroneous orders they reacted by uninstalling the new code from the servers it was deployed to correctly. In other words, they removed the working code and left the broken code. This only amplified the issues causing additional parent orders to activate the Power Peg code on servers, not just the one that wasn't deployed to correctly. Eventually they were able to stop the system – after 45 minutes of trading.



In the first 45-minutes the market was open the Power Peg code received and processed 212 parent orders. As a result SMARS sent millions of child orders into the market resulting in 4 million transactions against 154 stocks for more than 397 million shares. For you stock market junkies this meant the Knight assumed approximately \$3.5 billion net long positions in 80 stocks and \$3.15 billion net short positions in 74 stocks. In laymen's terms, Knight Capital Group realized a \$460 million loss in 45-minutes. Remember, Knight only has \$365 million in cash and equivalents. In 45-minutes Knight went from being the largest trader in US equities and a major market maker in the NYSE and NASDAQ to bankrupt. They had 48-hours to raise the capital necessary to cover their losses (which they managed to do with a \$400 million investment from around a half-dozen investors). Knight Capital Group was eventually acquired by Getco LLC (December 2012) and the merged company is now called KCG Holdings.

A Lesson to Learn

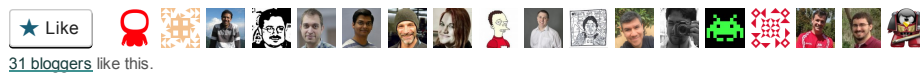
The events of August 1, 2012 should be a lesson to all development and operations teams. **It is not enough to build great software and test it; you also have to ensure it is delivered to market correctly so that your customers get the value you are delivering** (and so you don't bankrupt your company). **The engineer(s) who deployed SMARS are not solely to blame here – the process Knight had set up was not appropriate for the risk they were exposed to.** Additionally their process (or lack thereof) **was inherently prone to error. Any time your deployment process relies on humans reading and following instructions you are exposing yourself to risk.** Humans make mistakes. The mistakes could be in the instructions, in the interpretation of the instructions, or in the execution of the instructions.

Deployments need to be automated and repeatable and as free from potential human error as possible. Had Knight implemented an automated deployment system – complete with configuration, deployment and test automation – the error that cause the Knightmare would have been avoided.

A couple of the principles for Continuous Delivery apply here (even if you are not implementing a full Continuous Delivery process):

- **Releasing software should be a repeatable, reliable process.**
- **Automate as much as is reasonable.**

Share this:



31 bloggers like this.

FILED UNDER: DEVOPS

TAGGED WITH: CONTINUOUS DELIVERY, DEVOPS, RELEASE MANAGEMENT

« Strategy: The Art of the Product Manager

Scaling Agile Across the Enterprise »


Comments



Craig says:

April 27, 2014 at 5:46 AM

45 4 **Rate This**

Yes, DevOps ld have helped here, but sound coding practices would have prevented it entirely. Never re-purpose a variable.

Reply



Marc Seiler (@digitalfiz) says:

February 4, 2015 at 9:40 AM

9 1 **Rate This**

I was thinking the same thing the entire time i was reading this. **Its simple to create a new flag and it would have prevented something like this from happening.** It doesn't change the message this post is supposed to convey because it was mainly about bad deployments and the problems it causes but there is another message here: DON'T REPURPOSE FLAGS/VARIABLES! 😊

Reply



yesthatom says:

April 27, 2014 at 7:52 AM

2 1 **Rate This**

I can't believe they didn't deploy the code with the flag off, then wait a bit before flipping it on.

Reply



Martin Barry says:

April 30, 2014 at 1:14 AM

8 1 [Rate This](#)

Tom, I believe their use of the term “flag” is not in line with what we would normally think of in deployments (i.e. “feature flag” http://en.wikipedia.org/wiki/Feature_toggle) but instead it’s a variable in the data format of the orders being sent to the system. This means that “turning it off” was probably not so easy as you would need to change the order format being generated by the upstream system. Nor is it clear what would have happened to orders submitted without the RLP bit on.

[Reply](#)**Nav Marwaha says:**

May 5, 2014 at 4:54 AM

2 2 [Rate This](#)

Good post, regardless of development practices, level of experience or the number of quality gates, automation is a requirement for reliability.

[Reply](#)**VJ says:**

February 3, 2015 at 5:59 PM

3 1 [Rate This](#)

A scenario :

Lets assume they had very good DevOps. So all servers would be in sync. But – assume that the new code had a bug. So all servers are in synch, but have the same buggy code.

What if two versions of the code, i.e. the last 2 deployments had this bug.

So once they realize that something is wrong, they’d roll back the code, the bug still stays... Precious minutes have gone by. Maybe 20 minutes instead of the 45 minutes in your article.

So in short – their disaster / kill-switch is a code rollback + deployment in a live environment.

That would still be a defective design. What they’d need would be a big red switch (almost literally, somewhere in their dashboard) to immediately stop. Where is the business rule that says “first do no harm”.

[Reply](#)**Eric Minick says:**

February 4, 2015 at 10:48 AM

3 1 [Rate This](#)

VJ if the deployment to all servers had worked, they would have been ok. But in this case, 7 of 8 for one subsystem were deployed to correctly. Because the bad behavior, they rolled back the other 7 thinking the new code in that subsystem was the problem. That multiplied the problem until the eventual kill switch.

Disasters are almost always complex. In this case it was poor coding practices, plus questionable testing / code inspection practices, plus an error in deployment, plus a rollback at the granularity of the subsystem rather than the whole system. If you resolve any of those issues, you don’t get a disaster.

[Reply](#)**Jay Conrad says:**

July 26, 2016 at 11:12 AM

7 1 [Rate This](#)

One of the things I’ve seen in companies who don’t recognize the true importance and impact of their IT systems is that they don’t provide budget for legacy code updates. For example: I’ve seen situations where IT has no budget. It has to justify everything it does against a business expense. Which means constantly scrambling to line up new projects. Business rarely sees the need to update old software that is currently working, so they refuse to pay for it. The result is constant new code, made by the cheapest coders possible, while not investing in the technologies that would ultimately improve performance and mitigate risk. Why? Because these are seen as “IT problems” and not the purview of whatever project the IT people are working on, so nobody will pay for it. A great read regarding this practice is The Phoenix Project by Gene Kim, Kevin Behr, and George Spafford.

**darkfader says:**

March 16, 2015 at 12:51 PM

0 1 **Rate This**

Thank you for applying brain to the hype.

Probably one should ask why the techs involved did get to take the blame but didn't get authority to kill switch on their own.

Oh right, that's why you put Ops/SRE in place anyway. "R" is for responsible, aka flame bait.

Reply**allspaw says:**

February 3, 2015 at 6:11 PM

5 1 **Rate This**

I have written a bit about this event, and I would caution anyone to use the SEC report as anything at all other than for what the SEC needed it for. <http://www.kitchensoap.com/2013/10/29/counterfactuals-knight-capital/>

Reply**Marcus says:**

February 3, 2015 at 7:34 PM

1 1 **Rate This**

Fascinating read. I worked at a large auction house for fruits and vegetables once where a new software version was installed and failed, leading to large losses to the traders (although not as massive as these). This too was a case of improper deployment and no fall-back.

Reply**vlad ill says:**

February 3, 2015 at 7:37 PM

4 4 **Rate This**

The lesson to be learned is that there are domains where computer should not take any decision without human validation. What about the people who lost their jobs because, oops, there was a bug? What about the other companies that maybe got into trouble because of sudden change of the stock value?

Automation of "high level decisions" is to be handled carefully...

Nice and educational post Btw.

Reply**joskid says:**

February 3, 2015 at 7:52 PM

0 1 **Rate This**

Reblogged this on josephdung.

Reply**Mark A Hart, NPDP (@OpLaunch) says:**

February 3, 2015 at 9:48 PM

2 4 **Rate This**

Using the #Cynefin framework provides a better characterization of this 'DevOps' failure

This post seems to have been written from a DevOps perspective. The suggested solutions are consistent with a DevOps perspective – examine the release process, automate more, and craft a kill switch with rollback capabilities.

Someone may read the post and put too much emphasis on the Knight technician that did not copy the old code to one of the eight servers. Someone may oversimplify a cause and effect relationship. Someone may insist on new rules to 'prevent this from ever happening again.'

A more powerful approach may invest to:

- Increase diversity to analyze the situation and synthesize better options
- Improve communication between specialties
- Improve implicit coordination between specialties
- Recruit individuals with more expertise to write and review code

A major factor that limited improving the capability of the team from nine years prior to the significant failure event was mis-characterizing the system. In a Cynefin framework, confining this failure to a DevOps problem is associating the system with the “Obvious” domain where there are simple cause and effect relationships that are recognizable by ‘professionals.’ The failure should not be associated with the Cynefin “Complicated” domain where a significant analysis by ‘specialists’ would have prevented the failure. The system should be associated with the Cynefin “Complex” domain – a complex adaptive system. The system is dispositional. The same initial conditions will not produce the same failure (except by accident). For more information about Cynefin, visit <http://en.wikipedia.org/wiki/Cynefin> and @CognitiveEdge.

Reply



Scott Mann says:

February 4, 2015 at 5:02 AM

0 1 **Rate This**

I appreciate your highlighting the tacit factors in such a catastrophe. Like the author, I also work in operations, and it's easy to fall into the same old thought patterns on causes and solutions. I particularly enjoy your point relating to diversity (Which comes in all forms: experience levels, cultural and educational backgrounds, skillsets, age, etc.), as I think this is a strong driver behind the success of DevOps itself. Having a variety of perspectives, both within and without your team, looking at your project has strong, demonstrable potential and can help curb oversights such as the one brought up in this article.

Reply



sushil10018 says:

February 4, 2015 at 2:07 AM

0 1 **Rate This**

Reblogged this on sushil10018.

Reply



malikest says:

February 4, 2015 at 2:17 AM

0 1 **Rate This**

Reblogged this on malikesaint and commented:

Deep

Reply



m50d says:

February 4, 2015 at 6:15 AM

3 1 **Rate This**

> why code that had been dead for 8-years was still present in the code base is a mystery, but that's not the point

On the contrary, that's exactly the point. Code with unused, and therefore untested, configuration possibilities is a disaster waiting to happen.

This is why I'm very sceptical about feature-flag-based approaches generally. Configuration is as much a part of your program as code is, and configuration changes should go through the same lifecycle – pull request, code review, release, deploy to staging – as any other change. If your release process is too heavy and you need to make fast config changes to production, fix your release process.

Reply



CFC says:

February 4, 2015 at 7:26 AM

2 1 **Rate This**

There were too many mistakes to attribute the epic failure to just DevOps (though I do fully agree that automation and testing is the only way):

- No teamwork and checklists while doing an update on *production* servers. Any update on production should require a team watching over each other, and going through a checklist.
- 8 years of unused old code in production. That tells you a lot about the lack of understanding about the risks of dangling “unused” code.
- Insufficient logging [from the code], and insufficient real-time log monitoring, correlation and analysis. That would have triggered enough clues early on to the engineers and ops folks.
- No hot-hot failover to a cluster with the previous version. That would have stopped all issues after 1 or 2 minutes. (That’s the bug red button that the article mentions)

If you also have been architecting software, systems and enterprises for a long time, you know disaster happens, you know some bugs are only caught in the wild and not during simulations, just like you know machines will go down. You need to prepare for the worst case in both scenarios.

Murphy’s Law is so true in our world...

Reply



dwntwn1973 says:

February 4, 2015 at 9:03 AM

8 1 **Rate This**

I’ve been in what is now called the “DevOps” space for nearly 20 years, over half of that in the financial world. Knight was both a vendor to and a competitor of the company I currently work for.

Deployment automation *might* have helped. Maybe. But few companies can afford exactly duplicate environments, and this was essentially caused by environmental differences. Even automated validation of deployment might not have helped in this case if the automation didn’t know about the environmental difference. Automation is only as good as the knowledge of the people setting it up. If a manual install wasn’t aware of the old system, then there’s a good chance the automated system wouldn’t have known to check for it either.

Automation of a rollback is also only as good as the decision-making on whether to make the roll-back. And if the automation inadvertently started the old system, there’s also no guarantee that switching the contemporary system back would have stopped the old system – you could have ended up with the same problem even after an automated rollback of the contemporary system.

Which brings me to a final point: Automation is a requirement in large, modern, environments. But over-reliance on it can lead to the people operating the system being unaware of what it’s doing. Automation is most useful for validations, because validating things are done correctly is tedious and easy to skimp on when done manually. Even when automating, having human-involved breakpoints or human-driven steps helps insure that those operating the system know the system and how it operates, greatly improving their ability to troubleshoot issues, diagnose problems, and make appropriate suggestions on what steps to take to stop or mitigate a problem. Automation is a tool, but it is only one tool and it still requires a craftsman to wield it appropriately. Expertise is what makes and keeps great systems great.

Reply



GarrettHampton says:

February 4, 2015 at 9:06 AM

0 1 **Rate This**

Reblogged this on Garrett S. Y. Hampton and commented:
Incredible. DevOps always watch, document, and review your deployments!

Reply



Satan says:

February 4, 2015 at 1:48 PM

3 1 **Rate This**

The exact same thing could have happened with an automated deployment. What they should have done is kill Power Peg with fire, deploy, verify, and THEN deploy the new functionality.

Reply



Bobby says:

February 4, 2015 at 2:50 PM

0 1 **Rate This**

There were a few problems... the First problem was they had no Disaster Recovery Plan, i.e. a kill switch. Leaving the whole system running and while a problem was apparent was a bad idea. Included in the lack of a Disaster Recovery Plan was knowing the disaster was going on. There were no indicators showing that they were making trades without the parent order, nor something showing how much money was payable and how much was received. They also didn't have a plan to implement a kill switch. The Second problem was a function of being incomplete and not keeping code clean, as the "Power Peg" code was still in the original code and never removed after going unused. Along with that, they didn't test the prior code against the new order format, so they didn't have a reliable rollback plan. Lastly, they deployed without a mechanism for ensuring that all installed code was identical on on servers, and they didn't have a monitor on each node so they could find the problem node faster.

[Reply](#)**asdf says:**

February 5, 2015 at 12:53 AM

1 2 Rate This

I don't see any DEV in the OPS story here, but thanks, it was very interesting to read technical details about the incident.

[Reply](#)**thenrio says:**

February 5, 2015 at 1:18 PM

1 1 Rate This

Looks more like a product failure...

Failure to understand what your you are really doing

I know of a quite similar tale, and only similar because it was only a deployment issue

SG warrant market making in HK, approx 15 years ago

(as a dis, I CAN be wrong on some figures as it was a tale related by SG mates, still I deeply trust them because we worked together on this market making for more than 2 years, and I left before it happened)

Bad refill and limit configuration, bad underlying for one instrument => bad price, arbitrated by automaton of doetsche boerse

Loss was 4M€ in 4 minutes

And the operator stopped the market making

.

4 minutes, not 45

Could and should have been less

My point is that the operator had a deep understanding of how market making worked...and how to stop it

Electronic trading that you can not halt is a failure to understand electronic trading

It is not a deployment issue per se

[Reply](#)**AirFrank says:**

February 5, 2015 at 2:19 PM

2 1 Rate This

Never leave old code behind and don't reuse old flags. Had this happen on International Space Station. Guy overloaded a flag and in a future upgrade forgot that it was overloaded. I asked why he did that and he replied that he didn't want yet another flag in the system.

[Reply](#)**fubar says:**

February 5, 2015 at 8:56 PM

4 2 Rate This

Shouldn't the real lesson learned here be why stockmarkets can allow something like this to happen in the first place?

[Reply](#)

**glass says:**

February 6, 2015 at 12:46 AM

3 1 Rate This

must be hell of a release engineer, took Knight to leading trader just by doing manual deployment.

Reply**mrpjscott says:**

February 17, 2015 at 10:51 PM

0 1 Rate This

Reblogged this on [refresh.me](#) and commented:

This article is an awesome example of why automation is so important for businesses. I know that I personally will do my best to introduce more automation and configuration management tools into the environments I work on. This article is a reminder that without it, terrible things can happen!

Until next time!

Reply**PieceDigital says:**

June 19, 2015 at 7:04 PM

0 1 Rate This

How do you reactivate old code that's laid dormant for ages? Reusing a variable or route?

Reply**Ron Barak says:**

August 17, 2015 at 7:57 AM

0 1 Rate This

Excellent headup for new (??) DevOps.

Errata:

an average daily trading volume of more than 3.3 billion trades [daily]

The code that [that] was updated

Power Peg would instruct the [the] system

Reply**Joseph Kamal says:**

July 26, 2016 at 2:46 PM

0 2 Rate This

Awesome writeup, that was a great read and an excellent cautionary tale.

Thanks!!

Reply**quikedcode says:**

July 27, 2016 at 12:54 PM

0 1 Rate This

Did the IT office commit massive suicide when daily loss report arrived? 😊

Reply**discreetsecuritysolutions says:**

July 31, 2016 at 4:48 AM

0 1 Rate This

No killswitch? That's foolish.

Reply

Trackbacks

Feature Toggles from a Continuous Delivery Perspective » Programming Bytes says:

September 25, 2014 at 4:11 AM

0 1 **Rate This**

[...] support dangers of using – or misusing – feature flags was illustrated by a recent high-profile business failure at a major financial institution. The team used feature flags to contain operational risk when they introduced a new application [...]

Reply

Knightmare: A DevOps Cautionary Tale | Andymatic says:

October 12, 2014 at 11:15 PM

0 0 **Rate This**

[...] Nightmare: A DevOps Cautionary Tale [...]

Reply

4p – How a ~\$400M company went bankrupt in 45m because of a failed deployment – blog.offeryour.com says:

February 3, 2015 at 5:22 PM

0 0 **Rate This**

[...] <https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/> [...]

Reply

HACKERNYTT.se - dagliga nyheter för dig som bygger framtiden says:

February 4, 2015 at 3:28 AM

0 0 **Rate This**

Failade med devops – företaget kraschade

– användaren erikstarck postade en länk hit från Hackernytt.se.

Reply

Readings: Knightmare, Ulbricht Convicted, Franklin on Vaccinations | Neurovagrant says:

February 4, 2015 at 4:12 PM

0 0 **Rate This**

[...] Doug Seven: Knightmare: A DevOps Cautionary Tale – “This is the story of how a company with nearly \$400 million in assets went bankrupt in 45-minutes because of a failed deployment.” – I remember watching Knight explode in real time, had never heard the actual story. This is amazing. A great, and relatively short, post. [...]

Reply

1p – Knightmare: A DevOps Cautionary Tale – Exploding Ads says:

February 5, 2015 at 3:48 AM

0 0 **Rate This**

[...] <https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale?nb=1> [...]

Reply

Highlights KW 06/15 | drikkes.com/ says:

February 6, 2015 at 9:21 AM

0 0 **Rate This**

[...] Knightmare: A DevOps Cautionary Tale – Doug Seven – Wie man mit durch Softwarepfusch in 45 Minuten eine Firma plattmachen kann. [...]

Reply



Developer Catchup: New Node, Profanity, Oh-My-Git, Knightmares, Bad Docs and Go Tracing |

Codescaling says:

February 8, 2015 at 4:54 AM

0 0 **Rate This**

[...] to hear a horror story? Read Knightmare: A DevOps Cautionary Tale and wince at the pain that took a company down in 45 minutes. Yeah, dodgy dossier [...]

Reply

QCon London 2015–Takeaways from “Small is Beautiful” | theburningmonk.com says:

March 5, 2015 at 6:20 PM

0 0 **Rate This**

[...] then retold the well-known tale of how Knight Capital Group lost \$460 million in 45 minutes from a different perspective to highlight the potential danger of leaving old code [...]

Reply

Knight Capital Group - Seite 4 says:

April 2, 2015 at 7:26 AM

0 0 **Rate This**

[...] Knight Capital Group Hier ein Blogeintrag was (technisch) schief lief:
<https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/> (Eigentlich das selbe wie Lancelot bereits vor 2 1/2 Jahren geschrieben hat. Ich hab's trotzdem [...])

Reply



Why Driverless Cars? | Survival of the Craziest says:

April 11, 2015 at 7:56 AM

0 0 **Rate This**

[...] long as software is written by humans, there can and will be disastrous and potentially fatal mistakes. Not too long ago, the good folks who write software that puts [...]

Reply

CodeMotion 15–Takeaways from “Measuring micro-services” | theburningmonk.com says:

June 4, 2015 at 3:00 AM

0 0 **Rate This**

[...] Because it helps us deal with deployment risks and move away from the big-bang deployments associated with monolithic systems (a case in point being the Knight Capital Group tragedy). [...]

Reply

What I've been reading - June - Tyler Crammond says:

June 15, 2015 at 5:30 AM

0 0 **Rate This**

[...] Dev ops gone wrong – bankrupt in 45 minutes The story of how a company with nearly \$400 million in assets went bankrupt in 45-minutes because of a failed deployment.
<https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/> [...]

Reply

Professional Release Engineering | Just Digital People says:

June 16, 2015 at 10:29 PM

0 0 **Rate This**

[...] I chat with some dev afterwards about release engineering practices at their companies, and the struggles they have to craft professional software with best practices. As a former manager I know what it's like when a software professional wants to over engineer the hell out of something, and as a software engineer I'm also aware of the evils of premature optimization. And, an engineering slip-up in release management and deployment can bankrupt your company – just ask Knight Capital Group. [...]

Reply

Devops - what is it and why should you care? - Steiniche says:

June 29, 2015 at 1:23 PM

0 0 **Rate This**

[...] are yet to see that it will bring value to them and their customers businesses. The story about how Knight's Electronic Trading Group (ETG) lost nearly \$400 million in about 45 minutes, where the fault was placed with their deployment [...]

Reply

Extended Goals for Developing | AOD Coding says:

July 29, 2015 at 11:59 AM

0 0 **Rate This**

[...] – a company that lost over \$400 million dollars in one night because of a technical failure. <https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/>. Bloomberg reported the same [...]

Reply



Se7en Deadly Sins to Do in Python code | A Techie's Journey Through Life ... says:

August 10, 2015 at 4:06 AM

0 0 **Rate This**

[...] your company can lose millions and can be out of business. Maybe after some sleep-less night of the development team, the “bug” would have found [...]

Reply

Confluence: Engineering says:

January 14, 2016 at 7:47 AM

0 0 **Rate This**

Introduction to Cloud Orchestration

Why Orchestration ? It is worth following the story of Knight Capital to realize the need of Orchestration. This high speed trading company lost nearly \$400 million dollars within 45 minutes. This was caused by a human error in manual deployment....

Reply

The Power Equation and the Financial Rise of the UK | Ramen IR says:

February 23, 2016 at 7:39 PM

0 0 **Rate This**

[...] was “45 minutes of hell”. On August 1st 2012, Knight Capital went from having \$365 million in available cash to \$460 million [...]

Reply

The Power Equation and the Financial Rise of the UK | Ramen IR says:

February 23, 2016 at 7:39 PM

0 0 **Rate This**

[...] was “45 minutes of hell”. On August 1st 2012, Knight Capital went from having \$365 million in available cash to \$460 million [...]

[Reply](#)**Feature Toggles Revisited – Feature Flags, Toggles, Controls** says:

February 24, 2016 at 1:35 AM

0 0 [Rate This](#)

[...] is that bringing untested code into production that might be exposed accidentally is a bad idea. He cites a business failure at a financial institution as an example for such a situation. “Feature toggles require a robust engineering process, [...]

[Reply](#)**The Tao of Microservices | Richard Rodger** says:

February 24, 2016 at 8:39 AM

0 0 [Rate This](#)

[...] that can be solved with deployment automation and management (you should be doing this anyway, even for monoliths!). They are not fundamental weaknesses, no more than the need to compile a high-level language [...]

[Reply](#)**The secret step to save your software from the subtle death of inmaintainability | The Coding Flow** says:

April 30, 2016 at 10:50 AM

0 0 [Rate This](#)

[...] the importance of this last step, let me tell you a story that I found on Doug Seven’s blog about a high-volume daytrading firm called Knight Capital [...]

[Reply](#)**A Buggy Release – Voice of the DBA** says:

July 26, 2016 at 7:47 AM

0 0 [Rate This](#)

[...] DevOps isn’t a panacea for building better software. Witness the issues at Knight Capital, where they went from having \$364mm in assets to losing \$460mm in 45 minutes. Mostly because of a [...]

[Reply](#)**Node.js Module Replacements, HTTPS Adoption, and Much More - Intertech Blog** says:

July 29, 2016 at 3:01 AM

0 0 [Rate This](#)

[...] In 45 minutes, a \$400 million company went bankrupt – all because of one failed deployment. [...]

[Reply](#)**A computer tech forgot to install some new code... – Crazy Facts** says:

August 9, 2016 at 8:51 AM

0 0 [Rate This](#)

[...] computer tech forgot to install some new code in a server. This resulted in a high speed stock trading company that did \$21 billion [...]

[Reply](#)**Quick Fact: A computer tech forgot to install some new code in... - Quick Facts** says:

August 11, 2016 at 12:02 AM

0 0 [Rate This](#)

[...] A computer tech forgot to install some new code in a server. This resulted in a high speed stock trading company that did \$21 billion in daily trades to go bankrupt in 45 minutes. Ref: dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/ [...]

[Reply](#)**Knightmare – Where were the testing and development practices? – Recurring Theme says:**

August 23, 2016 at 6:46 AM

0 0 Rate This

[...] Read the entire article here: <https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/> [...]

[Reply](#)**Automatizar cambios con la ayuda de Fibonacci – EACHANG says:**

September 19, 2016 at 3:21 PM

0 0 Rate This

[...] La automatización puede ser beneficiosa para simplificar el trabajo de configuración de los equipos (por ejemplo) y puede ir mas allá debido a paradigmas como SDN, el cual centraliza el plano de control de un sistema de red (Espero hacer una entrada de esto luego) y esto nos lleva a aplicaciones mas avanzadas como el tener un mayor control de hacia donde vamos a reenviar los paquetes (encima de los protocolos de enrutamiento que tengamos), QoS, y la posible interacción de APIs dentro de un sistema de red. También puede eliminar la posibilidad de error humano en la implementación de un sistema. (leer: Como un negocio puede ir en bancarrota en 45 minutos por un error humano) [...]

[Reply](#)**Powershell: Backup & Copy – Cultivating Software says:**

November 3, 2016 at 9:02 PM

0 0 Rate This

[...] deployments can save you a great deal of headaches and even trauma in some cases. However I found it disappointing when I actually tried to put the best practice of [...]

[Reply](#)

Leave a Reply

TAG CLOUD

.NET 4.5 90-days Agile **Android** Apache Cordova **Arduino** Azure IoT Build Cloud css **Device** DevOps Google **HTML5** Hybrid **Icenium** Intel Edison **iOS** **IoT** **IoT** Workshop iPad iPhone JavaScript Johnny Five LiveSync MacOS X Marketing **Metro** style **Microsoft** Mobile Native Node.js PaaS PhoneGap Positioning Random Raspberry Pi 2 **Release** Silverlight SW8DPD **Telerik** **ThingLabs** UI Visual Studio Visual Studio LightSwitch 2011 **Windows** 8 Windows 10 IoT Windows Phone WinRT XAML

LATEST TWEETS



dseven
@dseven

A new konektd is out at konektd.com/?edition_id=59... #IoT #ConnectedCar Stories via @AllAbout5G @StreamNowCEO @Smart_Technical

Will Indian startups help build something li...
yourstory.com In the factories of Tata Motors, ...
konektd.com

ABOUT.ME

about.me/dseven

21h

[Return to top of page](#)

