Re-Wiring Enterprise IT With DevOps

Nico de Wet, MSc nico@nicodewet.com

October 13, 2016

Figure 1: *Mental Fortitude Required.*

DevOps `FROM scratch`[1] within siloed enterprise IT is anywhere from hard to exceptionally hard. As a practitioner your commitment to the underlying principles and practices must be total. With overhead political fire, a starting Joel Test score of 2 and enterprise IT silos you must succeed.

This article chronicles an enterprise experience of introducing DevOps and continuous deployment practices into, from a seasoned DevOps practitioners perspective, a virtual Ground Zero[2] with overhead tracer fire. In the spirit of *No More Consultants*[1] this was done in-house at the kiwi enterprise Jukt Micronics[5] by a committed team of veterans and DevOps newcomers. This article details the technical, on-boarding and political lessons learned during the first eight months of the evolution of the containerised continuous delivery pipeline at Jukt.

Although DevOps has multiple goals, our key measurable goal at Jukt was the ability to go from commit to production in the order of minutes while ensuring high quality. In this regard, in this article I steer clear of only focussing on the technical lessons learned in achieving the cited delivery goal. Rather, a key focus is on steps taken to cultivate a high performance culture and a total commitment to tried and tested practices - with container technology and open source in general as catalyst and accelerant respectively.

Moreover in terms of culture, in this article I propose that the cultural changes put forward in *No.8 Re-Wired*[2] directly apply to *Re-Wiring* Enterprise IT with DevOps. Furthermore, I propose using the **No.8 Re-Wired DevOps Scorecard™** as a means of incrementally measuring cultural progress.

1 Defining DevOps

In many ways the term *DevOps* is unfortunate in that to the uninitiated it may mean development and operations working together. The latter notion is true but also overly simplistic. DevOps refers to a range of mature software development and operational practices.

Some assume that DevOps refers to breaking down enterprise silos and that a startup implies DevOps due to low staff numbers. The latter is not true and also overoptimistic. With the perfect storm of

[1] In this context `FROM scratch` is a turn of phrase also referring to the *scratch* Docker base image.

[2] The World Trade Center site after the September 11 attacks.

personalities silos can and do exist with small staff numbers, even in startups. In terms of further confusion, some assume the scope of DevOps is limited to automatically standing up dev and test environments, this is also not necessarily true.

Although an in-depth coverage of DevOps is beyond the scope of this article, my coverage of the Jukt DevOps delivery pipeline and its constituent components will cover *some* foundational practices such as continuous integration.

In terms of defining DevOps, I borrow the SEI's definition sourced from the book *DevOps: A Software Architect's Perspective*[4]:

> *DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.*

Some key summary points that are made[4] to elaborate on the above:

- The quality of a deployed change to a system in terms of code, suitability of use by various stakeholders, various "ilities" (e.g. reliability) is important and the definition does not focus on *how* these are achieved.
- The definition requires the delivery mechanism to be of high quality. In other words, the quality of every part of the pipeline matters.
- Two time periods are important, the first starts when a developer makes a commit and the changes flows through the delivery pipeline to production. The second time period is the time between deployment into production and subsequently considering the change as a portion of normal production (this time period accounts for live testing and close monitoring of the change in production).
- The definition is goal oriented. The form of practices and tools are not specified.
- The goals in the definition does not restrict the scope of DevOps practices to testing and deployment. The operations perspective and monitoring practices are naturally also within scope.

2 No. 8 Re-Wired Enterprise IT with DevOps

In *No.8 Re-Wired*[2], Downs and Bridges celebrate the inventions and ingenuity of kiwis. They also put forward a set of cultural changes to truly take New Zealand innovation, both within and outside of enterprises, to global success.

I believe and propose that the cultural changes put forward in No. 8 Re-Wired[2] directly apply to *Re-wiring* Enterprise IT with DevOps. Moreover, without diving into technical matters the *FROM* and *TO* behaviours in Table 1 directly apply to enterprise IT silos.

The *No.8 Re-Wired DevOps Scorecard*^{TM} presented in Table 2 is my proposal for an internal self-assessment tool. When starting up DevOps in a given enterprise, or re-igniting DevOps for that matter, each member of what you believe to be your DevOps team fills in the form before kicking it off and at chosen intervals.

It is worth noting that it is entirely possible that your self-assessment exercise will in itself end up being flawed if a significant portion of your pipeline to production has been siloed off. Take a hard honest look at your pipeline. It may well be that in a given enterprise you will be limited to subset of mature engineering practices that formulated DevOps because silos cannot be broken down.

3 Ground Zero

In terms of the starting point, I use the Ground Zero analogy to contrast where one may have wanted to get to and where one may be at the start.

Informally speaking, as a software engineer Ground Zero represents rappelling down into Mogadishu in

Table 1: *The No. 8 Wire Paradigm[2]*

FROM	TO
Alone in a shed	Collaborating with peers
Making a solution to solve my own problem	Understanding the customer's problem
Making do with what's at hand	Using the best possible technology from around the world
Distrusting others	Sharing information with others and mutually benefitting
Doing it all myself	Working with specialists in design, marketing, production, etc.
Holding onto the invention	Letting the invention go out to be criticised, commented on and added to by others
Making it 'good enough'	Perfecting the design and functionality
Standing apart from the world	Being open to global cooperation

Table 2: *No.8 Re-Wired DevOps Scorecard*^{TM}

FROM	TO
Alone in an IT silo	Collaborating with DevOps peers
Making a solution to solve my own silo's problem	Understanding the customer's problem with DevOps practices as the means
Making do with what's at hand	Using the best possible technology from around the world
Distrusting other silos	Sharing information with others and mutually benefitting
Doing it all myself	Working with specialists in design, marketing, production, etc.
Holding onto the invention within my silo	Letting the invention go out to be criticised, commented on and added to by others
Making it 'good enough'	Perfecting the design and functionality
Standing apart from the business and other IT silos	Being open to enterprise-wide cooperation

Figure 2: *Rappel Down To Ground Zero*

the seminal film Blackhawk Down. The only difference is that the engineer would be rappelling down with a MacBook as your assault rifle.

Consider the following scenario, to set the scene, which may play itself out in IT in any given enterprise and in any given country.

- In-house developers not furnished with developer specced machines. Forced to bring in personal resources where unavailable in the enterprise.
- Continuous integration practices generally not followed or unknown.
- DevOps not understood top down in IT before launching it - siloes firmly entrenched even after launching DevOps.
- Open hostility to in-house developers commonplace within the enterprise and political drives to oust linux.
- In-house developers faced with an *enterprise IT developer* stigma in industry.
- Significant portion of development are contractors with less skin in the game by definition.
- IT in the midst of restructuring.
- Principle of least privilege applied universally, even within DMZ.
- Software architecture discipline poorly understood.
- Significant amounts of shadow IT when it comes to development.
- Agile projects represent cargo cult software engineering[3].
- Tertiary IT or Computer Science degrees rare, having absolutely no IT qualification commonplace.
- Near universal waterfall experience and no agile experience.
- Operations side of IT uncelebrated.
- Deployment to UAT and/or PROD taking days to weeks common.
- Development operating system and runtime never matches production.
- The ratio of developer to other IT roles may be in the region of 1 to 20.
- Automated regression testing suites generally unheard of.
- Active resistance to containerisation technology and in particular Docker in parts of the enterprise.
- Millions of dollars spent on projects but almost nothing allocated for maintenance nor training.
- The desire to hide and bury mistakes.

So, having read that list, you may be wondering, is this not a certain death scenario? Surely DevOps will never work no matter how talented the software engineers may be.

Well, in our case we had an influential leader with thirty years of in-the-trenches experience, still actively coding and a published, from the trenches, agile book to boot. This is a game changer, and having the backing of a respected and supportive veteran is why one would rappel down.

There is one more reason why you'd do it - because if you don't who will? You have to have skin in the game to truly be committed to fixing serious problems.

4 Starting Culture - No. 8 Re-wired Scorecard

I retrospectively completed the No.8 Re-Wired DevOps Scorecard^{TM} for Jukt and was able to do so given that I had lead the implementation of DevOps from inception. My retrospective score, illustrated in Table 3, shows a woeful score of 0.5 out of 8 which in itself represents the degree of opportunity for positive change.

Table 3: *No.8 Re-Wired DevOps Scorecard^{TM} - Starting Culture At Jukt Micronics*

FROM	TO	SCORE
Alone in an IT silo	Collaborating with DevOps peers	0
Making a solution to solve my own silo's problem	Understanding the customer's problem with DevOps practices as the means	0
Making do with what's at hand	Using the best possible technology from around the world	0
Distrusting other silos	Sharing information with others and mutually benefitting	0
Doing it all myself	Working with specialists in design, marketing, production, etc.	0
Holding onto the invention within my silo	Letting the invention go out to be criticised, commented on and added to by others	0.25
Making it 'good enough'	Perfecting the design and functionality	0.25
Standing apart from the business and other IT silos	Being open to enterprise-wide cooperation	0

5 DevOps Pipeline

5.1 Pipeline Mark 1 - The Show And Tell Bridgehead

When we kicked off DevOps at Jukt the first objective was to establish the pipeline all the way to the show and tell (SNT) environment, and no further. This was our *Mark 1* pipeline which is a bridgehead in terms of our overall goal.

The reason for the limitation of scope when it came to the first pipeline was the priority of demonstrating capability internally within IT as well as to business stakeholders. Arguably by focussing on SNT, rather than taking a DevOps pipeline all the way to production, as per our definition of DevOps, this in itself does not constitute DevOps, but that is fine.

Another reason for the limitation of scope is because just establishing the bridgehead may be a fair challenge in particularly constrained IT environments. For example, at Jukt just getting a handful of linux based virtual machines provisioned along with root access to each was a significant challenge. In addition, not all concerned were familiar with key technologies and automation to the extent that we aimed to introduce and so we had to allow for onboarding time and effort.

In terms of further limitations of scope, we did not use an Artifact Repository (which happened to be Nexus 3) when focussed on SNT and only gradually increased the maturity of the pipeline. In the very first edition of the pipeline we copied snapshot Docker images onto the target environment, rather then standing up a private registry.

5.2 Pipeline Mark 2 - Onto Production Under Fire

The pathway onto production was the most difficult, not technically, rather in terms of dealing with constraints imposed by silos. The bulk of our pipeline was still siloed off and we had to gradually gain the access[3] required for automated deployment from our central continuous integration server.

In terms of production, our pipeline never went further than pulling, stopping and starting individual containers with no host level access other than managing images and containers. The complete pipeline is illustrated in Figure 3.

The gains made by using containers were without a doubt fundamental. In retrospect the restriction of only ever being able to deploy containers forced us to hone our core image management and Docker Engine skills. Moreover keeping the pipeline clean of garbage and nurturing our two Docker registries remained as challenges within our simple pipeline.

The above said in terms of fundamental gains, it

[3] These includes getting an operational/infrastructure team to allow specific sudoer commands (docker engine commands), opening specific firewall ports and more.

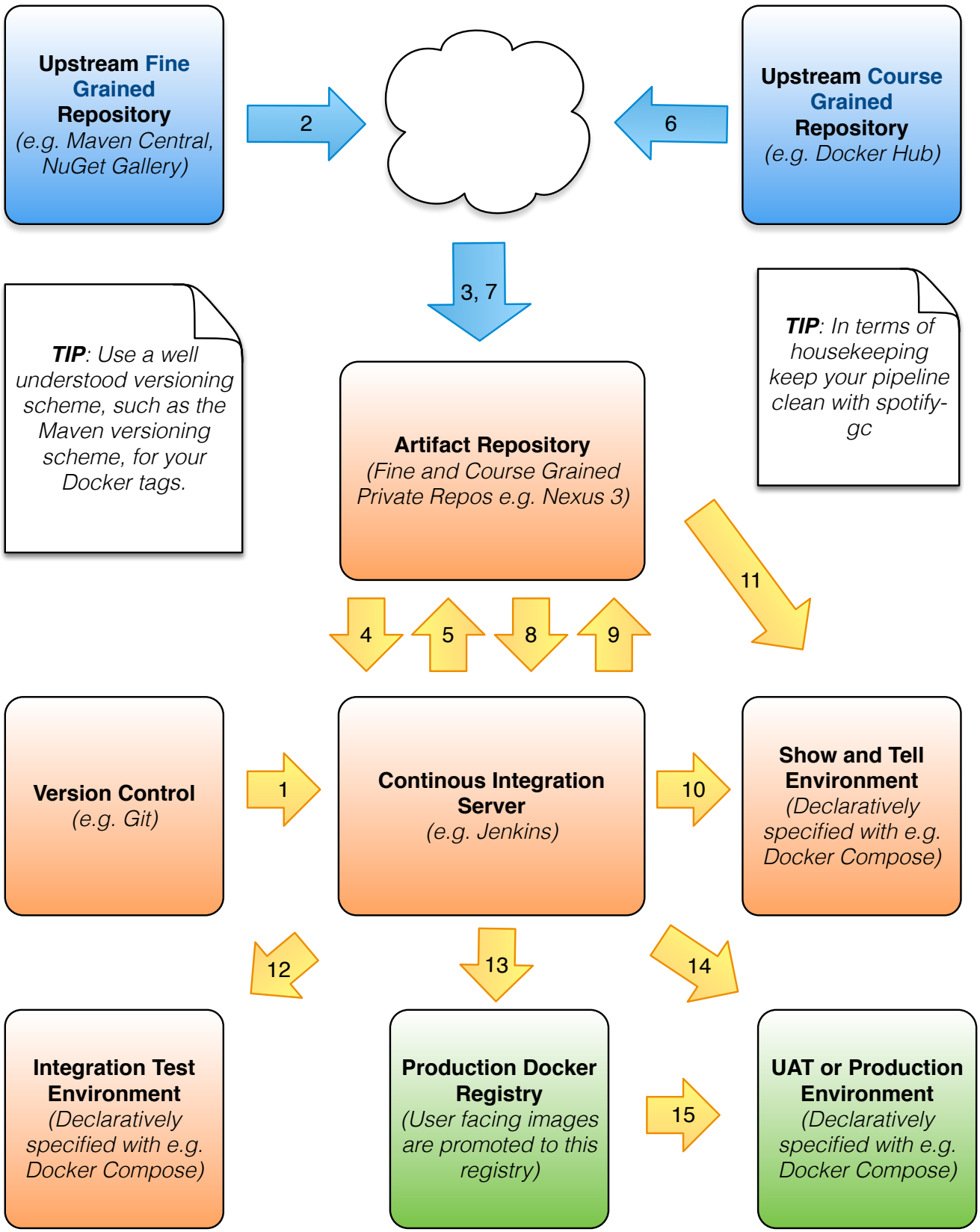

Figure 3: *The Docker Pipeline At Jukt Micronics After Eight Months*

Figure 4: *Establishing The Show And Tell Bridgehead*

Figure 5: *Moving Onto To Production Under Fire*

would be incorrect to say that we had successfully implemented DevOps at this stage. This is because a fair portion of the infrastructure in-between the end user and our containers were not managed using DevOps principles nor the DevOps team. In particular practices such as infrastructure-as-code and also the notion of temporary security credentials, as made possible by for example the AWS Security Token Service, were missing entirely. In short, we had no less than three IT silos that remained.

Our approach to mitigating the impact of the remaining silos was one of engaging individuals to facilitate collaboration, rather than engaging the corporate structure which was not something we could realistically attend to in the short or long term. We used base imaging as our engagement approach.

A key lesson learned when building this stage of the pipeline was one of patience and gradually taking team members who happened to be in another silo along for the journey. For example, although it was suboptimal, having access to a specific set of Docker Engine commands on upstream hosts and a specific set of firewall rules being put in to place was all we needed to auto-deploy from our continuous integration server. We effectively learned to accept the limitations in the short term and also build confidence as we went along.

In terms of where we could have done things better, as a blessing in disguise in the early production days we did experience a significant outage. It was a blessing in the sense that it exposed a number of cultural issues as well as highlighting the fact that not all had the basic Docker Engine experience and skill that they needed.

5.3 Pipeline Mark 3 - On-board In-house Expertise With Base Imaging

The final enhancement that we made to our basic image propagation pipeline was both cultural and technical.

We used Docker Engine base imaging as a means of working with our database administrators to build on popular open source database engine images published to Docker Hub. This helped in terms of leveraging their experience and also serving as a means of not only stimulating discussion but also encoding best practices either in Dockerfiles or as version controlled Markdown documentation.

Furthermore we worked with the operationally oriented team members to build on Apache httpd base images to route internal traffic.

Final enhancements made during this *Mark 3* phase of our pipeline was proxying Docker Hub, with Nexus 3, and also keeping our core build and dev infrastructure clean with *docker-gc*[6].

In terms of on-boarding lessons learned, the main lesson here was to not forget about our original, measurable goal which was to go from commit to production in the order of minutes while ensuring high quality. The reason why this is so important is because it helps challenge all concerned in terms of their ways of working, which might not be compatible with DevOps as we have defined it and are measuring it. Never lose sight of your overall goal and never compromise on doing what needs to be done to achieve it.

It is worth noting that during this phase we stuck to our *No More Consultants* way of working and believe that this helped with getting buy-in and also building a team with skin in the game.

6 Concluding Cultural Checkpoint - Revisiting Our No. 8 Re-wired Scorecard

Although, after eight months, we had achieved a fair amount up to the point in time where we had built what we like to call pipeline *Mark 3*, its important to look back and subjectively give ourselves a score.

It is also important not to forget the definition of DevOps before scoring because otherwise we might end up scoring higher than what we should be doing.

> *DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.*

In Table 4 we ended up, in my mind, with a score of 2 out of 8 which is a significant improvement on 0.5 out of 8. The scores remained low overall since I believed we still had a long way to go in each category.

In terms of the cited definition of DevOps, this is where we made the greatest gain in that deployment to production now took in the order of tens of seconds, whereas previously it could have taken days. We had not quite started to measure the reduction in time between committing a change and the deployment of the change in to normal production. We had also not started to distinguish between closely monitored production and normal production.

To conclude, looking back, although our journey was far from complete, in terms of subjective cultural measures and *some* objective measures we were light years ahead of where we were at the starting point. We had also introduced containerisation gradually, with baby steps, which looking back was a blessing. From here orchestration and scheduling are likely to be next ports of call.

References

[1] Parcell, Geoff. and Collison, Chris. (2014). No More Consultants: We Know More Than We Think *Wiley*, October 2009.

[2] Bridges, Jon. and Downs, David. (2014). No. 8 Re-wired - 202 New Zealand Inventions That Changed the World *PENGUIN BOOKS*, pg 9.

[3] Steve McConnell *Cargo Cult Software Engineering* 2000: IEEE Software. http://sunnyday.mit.edu/16.355/cargo-cult.pdf

[4] Len Bass, Ingo Weber, Liming Zhu *DevOps : a software architect's perspective*, First edition, May 2015, Pearson Education Inc, ISBN 978-0-13-404984-7

[5] Glass, Stephen. Former associate-editor at magazine *The New Republic* that reported on the non-existent software firm Jukt Micronics[4]. https://en.wikipedia.org/wiki/Stephen_Glass

[6] Spotify. *Docker garbage collection of containers and images*. https://github.com/spotify/docker-gc

[4] Similarly the kiwi enterprise Jukt Micronics is non-existent. I use this name as a generic replacement for an anonymous enterprise.

Table 4: *No.8 Re-Wired DevOps Scorecard^{TM} - Culture At Jukt Micronics After Eight Months*

FROM	TO	SCORE
Alone in an IT silo	Collaborating with DevOps peers	0.25
Making a solution to solve my own silo's problem	Understanding the customer's problem with DevOps practices as the means	0.25
Making do with what's at hand	Using the best possible technology from around the world	0.25
Distrusting other silos	Sharing information with others and mutually benefitting	0.25
Doing it all myself	Working with specialists in design, marketing, production, etc.	0.25
Holding onto the invention within my silo	Letting the invention go out to be criticised, commented on and added to by others	0.25
Making it 'good enough'	Perfecting the design and functionality	0.25
Standing apart from the business and other IT silos	Being open to enterprise-wide cooperation	0.25