devoteam

August 2020

The future of **DevOps**

Creative tech for Better Change

TABLE OF CONTENTS

EXECUTIVE SUMMARY

DevOps in the Enterprise, or DevOps at Scale, has a history as long as DevOps itself. This history is chequered with unfulfilled promises and gaps between the potential of DevOps for the Enterprise and what actually has been achieved.

While the benefits of DevOps are well understood and significant productivity and velocity gains have been made by businesses of all sizes, these benefits have not been widely realised when implemented at scale in large enterprises.

With so much to be gained, including the potential for a positive impact on costs that comes with improved productivity and speed - the need to pause, assess and re-think becomes apparent. The question becomes why has the full vision of DevOps not been realised? Building on this premise, are there any common wrong turns that have been made? What are the lessons to be learned? What areas need re-evaluating and what steps need to be taken to get DevOps working at scale and delivering it's true value for the enterprise?

This paper explores these questions and offers direction on how to pivot your DevOps strategy towards ongoing success. Starting with a brief history of how we got to where we are now, we then move on to an appraisal of some of the common legacy barriers to achieving DevOps at scale. We'll provide insight into what DevOps should look like for the Enterprise that's looking to reap the benefits of the productivity and velocity gains that are completely within reach for a large business, and yet despite talking about them for so long, have not been realized. We'll look at what aspects to consider on your journey to DevOps utopia, discussing the impact of the Cloud and predictions for the industry along the way.

DEVOPS BEGINNINGS

What's this DevOps stuff all about?

Well we can automate the way we build and test code. We can refactor much more quickly. We'll be more agile and save lots of time and money.

How much does it cost?
How much money will we save?

Umm. Not sure.

Ten years ago, DevOps was new, no one in the boardroom had heard the term, let alone understood what it meant.

The emergence of Cloud computing enabled the exponential growth of small, start-up businesses unencumbered with legacy infrastructure and software. These businesses had new designs, products and features - and the ability to deliver these features at an unprecedented speed. With agility built into their DNA, the startups were able to disrupt the decades old IT business models and delivery practices of the Enterprise, spawning new open source DevOps tools on a regular basis. The Enterprise could not keep up. Enterprise CIOs and CTOs realised the need to understand these new Agile and DevOps practices, and in many cases felt the need also to control these new practices using techniques that worked for them in the past - steering committees, budget reviews and CapEx funding models, centralised tools and teams.

Having tried to implement DevOps while relying on a legacy approach, the C-suite were frustrated that their changes did not seem to be making a significant difference. As hard as they tried to go faster, startups and other FinTech companies were going faster still. Flickr and Amazon were bragging about multiple deployments per day, yet the Enterprise was still trying to deploy more than once per quarter.

Many engineered lots of local optimisation to help individuals and teams develop, test and deploy code better and faster. But that meant that obstacles unique to the Enterprise were hit more quickly - lack of budget to obtain infrastructure, manual approval of the many changes required, understanding and meeting security and audit requirements of the business.

ENTERPRISE DEVOPS WHERE DID IT ALL GO WRONG?

The DevOps practice is up and running!

So things are happening faster?
We're saving on infrastructure costs right?

No. Not yet.

Oh.

Below are 6 key areas that have caused Enterprise DevOps to fail:

1. On-premise DevOps
2. Central DevOps
3. Shadow DevOps
4. (Lack of) Cost Transparency
5. Enterprise Silos
6. Control Tribes

ON-PREMISE DEVOPS

So, there are new versions of our build servers, think there's about 100 of them, so they'll each need upgrading

Um, yes, but, each instance takes 2 hours to upgrade, that's 200 hundred hours, so about five person weeks of maintenance work and there's only one of me and I'm on lunch in a bit...

On-premise DevOps requires on-premise infrastructure. In the pre-cloud world of Enterprise DevOps, Infrastructure was notoriously hard to obtain. So teams hoarded infrastructure they didn't necessarily need. And didn't always turn off infrastructure when it was no longer needed. This added cost and waste. On-premise managed infrastructure led to issues regarding maintenance, security, reliability and support.

CENTRAL DEVOPS

I'm still waiting on the Jira team to set up the new project...

I thought DevOps was supposed to speed all this up?!

Central DevOps teams were put in place to manage the enterprise DevOps tooling. Management didn't always understand the value of these teams and they would struggle for funding and were often outsourced as a consequence. As a result, the teams that should have been the drivers of efficiency and speed became the cause of bottlenecks and delays. Fast moving development teams and departments resorted to running their own tooling.

This led to the creation of Shadow DevOps and the vicious cycle of tool sprawl. Integrating and managing tools from various and often competing vendors was time consuming and costly.

SHADOW DEVOPS

Good teams want autonomy - autonomy to work in the way they want, with the tools they want. At a team level this is generally a good idea. Team members will typically build expertise in the tools, if they get to choose them. Good teams, often in isolation, grew organically to become better at DevOps. Many of the tools were open-source, or based on freemium models, which meant costs were low from a team point of view. DevOps engineers created lots of local optimisations, and some teams did perform better.

But what is good for the team is not always good for the enterprise. The Enterprise DevOps tooling estate grew. Vendors proliferated and sprawled across the Enterprise, dozens of instances of these tools running, creating risk and inefficiency. Many teams had developed local or team-based vendor relationships. The cost of maintaining this fleet of tooling became significant, although as we'll discover - hard to quantify in monetary terms.

COST TRANSPARENCY

Cost transparency was non-existent. For most developers, architects and IT managers there was no understanding of the financial costs of the infrastructure they were consuming. The costs were opaque and rarely considered. Without transparency there was no clear view on how much a team or project was spending on their infrastructure and tooling.

Servers could run idle 24/7 Databases that had been forgotten about would be backed up daily, for years on end. Redundant data whose origins and purpose was long forgotten, was never deleted.

ENTERPRISE SILOS

Our code is ready to go live!

Great - are all your tasks approved?

I've chased the DBA three times but they never reply...

In the traditional Enterprise environment, not only did we have separate silos for Development and Operations, but there were also silos for:

- Service Management
- Networking
- Database and Storage
- Security
- Middleware
- Central DevOps

Mundane tasks that require the approvals or actions from these teams are typically bureaucratic and slow:

- Request approval to release a code change into production
- Create a DNS name (mynewapp.com)
- Create a database, backup/restore a database
- Request a penetration test of a website or mobile application
- Create a middleware queue
- Create a Jira project, Confluence space, Jenkins project, GitLab repo

DEV VS OPS

DevOps is about breaking down the Dev and Ops silos and aligning the incentives of these two different parts of the business. Developers are incentivised to go fast, even if that sometimes means breaking things. They get paid to write software and the more they write, arguably, the better they are.

On the other hand, Operations teams are incentivised for stability. And the most stable outcome they can have is nothing changing. They are not happy being woken up in the middle of the night trying to diagnose bugs resulting from yesterday's new release.

CHANGE MANAGEMENT

Have you been to the Change Approval Board?

It was postponed until next week because there was an urgent release.

There's a change freeze next week.

!@*&.

In large and regulated industries all (code) changes released into production had to be approved and/or audited by an independent internal party. Typically this was the Change Approval Board, or CAB. The CAB typically met once a week to approve and coordinate changes for the following week. This process was in direct conflict with teams that wanted or needed to deploy to production in small, frequent batches.

THE RISE OF CONTROL TRIBES

There's a growing understanding that DevOps is not just about Dev and Ops. Control Tribes – Audit, Compliance, Security, Architecture, Operations, Accessibility, etc - play a significant part in determining the velocity of feature development. Security is seen as a major aspect of DevOps, so DevSecOps has emerged as a further refinement of DevOps.

Audit and compliance need to make sure that the data being stored in the database conforms to GDPR rules. Marketing teams want to make sure that web UIs are on brand. An accessibility control function may ask if your product is built and accessible for people with disabilities.

While necessary to ensure security and regulatory compliance, Control Tribe engagements that are manual and time consuming significantly impact team velocity.

DEVOPS AS A SERVICE - THE EVOLUTION OF DEVOPS

We really need to look at this new approach to DevOps

I thought we already have a DevOps practice?

We have, we just didn't get it quite right

There is a new way of thinking about DevOps that was born in the cloud and employs cloud based tools and pipelines - DevOps as a Service. Pivoting towards DevOps as a Service addresses many of the issues experienced with legacy Enterprise DevOps and puts an organisation on the path to maximising the potential DevOps.

The move to the Cloud and infrastructure-as-code are significant changes, ones that most large companies are only beginning to come to grips with. There are pockets of excellence, but to make this transformation happen at scale means adapting to a new set of challenges.

By tackling each of the areas laid out below, the transformation to DevOps as a Service can start in earnest.

COST TRANSPARENCY

Moving from Enterprise DevOps to DevOps as a Service means moving from a CapEx (capital expenditure) spending model to an OpEx (operating expenditure) spending model. In traditional Enterprise DevOps, understanding IT costs at a product level was incredibly difficult for some near impossible for others.

Now, Cloud providers offer highly detailed and itemised monthly bills that allow forensic analysis of IT costs. Every server, database and gigabyte of data is accounted for, and importantly can be attributed back to the product team. With clear visibility and understanding of costs, the next step of cost optimisation can become a reality.

CLOUD DEVOPS

Cloud vendors' integrated DevOps pipelines mean users no longer need to deal with multiple vendors and integrate multiple tools. DevOps pipelines become a commodity, eliminating tool maintenance. The major cloud vendors provide integrated CI/CD pipelines and toolchains that require no integrations, are evergreen and are highly cost transparent. Costs are operational not capital and there is no tool or vendor sprawl. Creating a database involves clicking a button on a user console. Using a cloud platform to implement DevOps creates a symbiotic working environment to resolve issues.

INTEGRATED TOOLCHAINS

In the new vision CI/CD pipelines become commodity items running on cloud platforms. You check in code and the rest of the CI/CD process just works, with the deployment of the newly built code into a Kubernetes cluster.

Cloud and GitOps pipelines are entirely integrated. Jenkins will become obsolete. Having hundreds if not thousands of plugins to integrate with tens and hundreds of tools is a legacy Enterprise problem that Cloud-based DevOps pipelines eliminate.

AWS, Azure and GCP all have integrated CI/CD pipelines and GitLab CI has end to end 'GitOps' integration. Weaveworks have a different view of GitOps that even eliminates the need for traditional CI/CD. So when all these options are out there, why build your own pipeline? The days of multiple tools from multiple vendors requiring multiple integrations with multiple plugins are coming to an end. With this change, much of the work that was done by traditional DevOps engineers will also become redundant.

EVERGREEN SOFTWARE

The introduction of software-as-a-service (SaaS), where the software is provided and hosted by a vendor, and paid for in monthly bills, has been revolutionary. In the new world, evergreen software continuously improves DevOps tools behind the scenes without consumers knowing. It's a much simpler, much cleaner way of working.

CONTAINERISATION

At the heart of cloud native DevOps is containerisation. This is the building, testing and deploying applications in containers and typically running these containers on a Kubernetes platform.

The approach to containersations varies from vendor to vendor...

GitOps is a way of executing CI/CD that solely uses a cloud-based Git platform, such as GitLab CI, GitHub Actions or arguably Jenkins X. It is a developer focused way of working that concentrates on tools that the developer is used to using. A single platform provides end-to-end CI/CD hence all functionality is integrated.

AWS provides integrated DevOps pipelines using their CodeCommit Git-based repositories and CodeBuild and CodeDeploy services. **Google Cloud Platform** (GCP) provides integrated CI/CD and Kubernetes[1]. Most DevOps processes are automated, removing the need for traditional Enterprise ways of working. **Microsoft Azure.** The introduction of Azure DevOps[2] has allowed Microsoft-based DevOps teams to automate their DevOps pipelines and speed up the development cycle. Azure DevOps also integrates Agile tooling such as Azure Boards.

[1] https://cloud.google.com/docs/ci-cd
[2] https://azure.microsoft.com/en-us/services/devops/

METRICS, MEASURES AND MONITORING

Cloud Ops and Integrated Pipelines are practices that are maturing. The next step is metrics, measures and monitoring maturity.

The ability to measure is essential - measuring the value in delivering your product and the velocity at which you are delivering it. In order to go faster, you need to understand where your bottlenecks are. Four key DevOps metrics (Lead time for changes, Deployment frequency, Time to restore service and Change failure rate) help measure velocity, understand problems and indicate where velocity can be increased. Business value is the most important metric to measure, and possibly the hardest to capture and understand, but it is essential to figure out how to do this.

Measuring usage is a good start and can serve as a good proxy for business value. How many people are using the system or platform? How many hits to the API? How many downloads of the app? How many transactions are processed? But the business may want to go beyond this and want answers to the questions of costs and savings.

For a platform team building an internal service, say IP Address Management or Database Admin it can be more challenging to prove value. Do you count the number of tickets you resolve? The number of Jiras you complete? The ability to measure the business value of what you are producing will help you understand where to spend your money and resources. We are still learning how to do that.

Measuring at scale, across a portfolio of products, requires consistency. Many organisations find it best to start small with a few DevOps-mature product teams and show how business value metrics can be collected and analysed. Find low-cost solutions to metric collection that work and that are scalable. Being able to analyse the data is key. Proving that the correct data gets captured must happen first. Automating the collection of that data can follow, once it's worth is understood.

The historical trends of these metrics can be revealing, so storing historical data is important. The question becomes - how should this data be captured and stored?

VALUE STREAM MAPPING

An antidote to local optimisation is to map the entire "value stream" - the journey a feature or product makes from "concept to cash". By mapping the journey or "flow", pain points and places where delays are occurring can be discovered. It can help reduce or eliminate waste, which often occurs during work handoffs.

THE SRE ROLE

Traditional DevOps roles are changing. Legacy Enterprise DevOps roles will diminish. Cloud Engineering and Site Reliability Engineering (SRE) will be key skills to making the move to DevOps as a Service.

The Site Reliability Engineer (SRE) is increasingly becoming a key role in the creation, delivery and operation of IT systems. Engineers who understand CI/CD, Networking, Security and Databases and can automate the infrastructure provisioning for them will be intrinsic to the profile of the roles you will need to create and fill in the future. The full scope of what an SRE does, understanding their role and responsibilities is something enterprises are still learning as DevOps as Service evolves.

PORTFOLIO MANAGEMENT

Portfolio management is another critical aspect of modern software product development that goes beyond what is strictly DevOps or Agile. Portfolio management considers what work should be undertaken by the team, department and company. Often there are many interests - e.g. products and features, competing for limited resources - people and money. Of all the interests of the company, which ones take priority? Which ones should be funded?

Their role is to make sure that the team, department or company is "doing the right work" - the work that will maximise the interests of the company. This can span profits, customer satisfaction or public good. They are the sentinels at the software factory door, making sure that the right materials are entering to produce the best outcomes. Once the work enters the software factory, DevOps is concerned with "doing the work right".

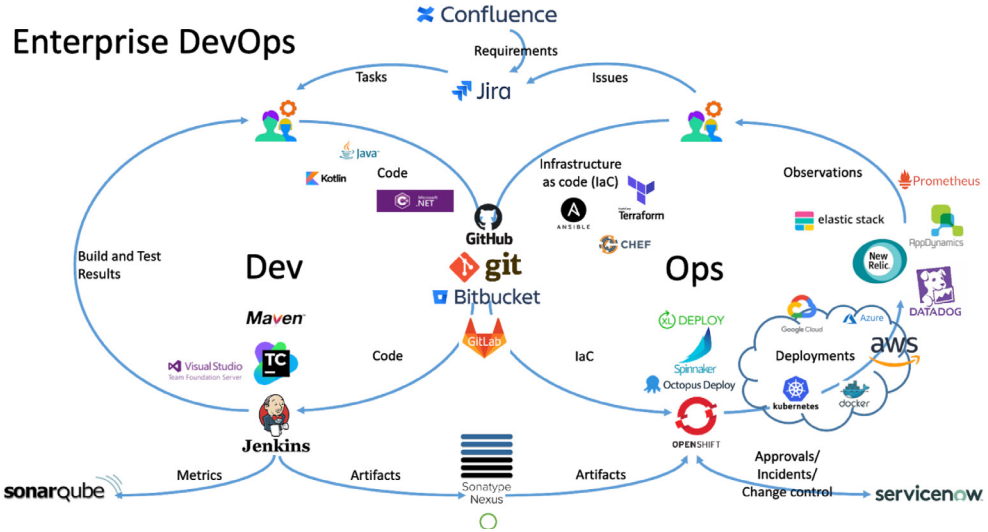
DELIVERING DEVOPS AS A SERVICE

THE SRE ROLE

In order to instigate real change and implement DevOps as a Service, legacy infrastructure and shadow DevOps need to be brought out into the light and cleaned up and cleared out.

Mapping your current DevOps model and assessing the tooling in place will give you the full picture of where your Enterprise is right now, in order to plan out your DevOps as a Service journey.

Below is an example of a DevOps Model, overlaid with a typical Enterprise DevOps toolset. At every point in the model there are several, often dozens, of tools in use, sometimes officially sanctioned sometimes not.



START SMALL

Scaling can be hard to manage, so large enterprises have a tendency to wrap lots of controls around scale. Mass migration to the cloud requires controls, which can easily mutate into bottlenecks. It's easy to end up slowing down, when you are trying to speed up.

Discover the working states, and the waiting states. When you get good at this at small scale, you are then in a much better position to begin scaling. Begin to build expertise in cloud native development and transforming teams and products.

BREAK DOWN THE SILOS

One of the biggest changes to instigate is breaking down the Dev and Ops silos and defining the SRE Role within your business. The aim being to move away from traditional Dev and Ops departments to a "you build it, you run it" engineering philosophy backed by the SRE.

ALIGN CULTURE AND TOOLS

> Tools do change the way you think and change the way you work.
>
> Gene Kim

The desire to improve tools has its roots in a desire for cultural advancement and consistency, and improved software quality. In DevOps, tools and culture are deeply intertwined. To have a successful DevOps transformation, both good culture and good tools are vital.

IMPLEMENT LEAN CONTROL

It's important that Control Tribes are aligned. They need to understand what their requirements are, ensure these requirements are visible to all and prioritised along with new features. The non-functional requirements should be in the same backlog as all the other work. Control engagements with product teams should be meaningful and minimal. They should provide enough engagement to ensure requirements are satisfied without becoming a burden or bottleneck.

AUTOMATE THE MUNDANE

The increase in automation that DevOps as a Service will deliver will mean that 80% of current roles will disappear. Many people who work in centralised operational and control roles such as service management, middleware, or database management will need to think about re-skilling.

In particular, much of the work undertaken by network, database and middleware admins will be automated. Not all network, database and middleware admin jobs will become obsolete, but many of their tasks are already being automated as the move to the cloud gathers pace. The remaining work will be focused on the more difficult but more creative 20% of tasks that can't be automated. The key to survival for those in legacy roles will be to pivot towards these more creative tasks.

RESOURCE FOR THE NEW WORLD

Finding the resources to help achieve these transformations at scale isn't easy. Those with cloud and SRE skills are in high demand. The education system is struggling to train people in these new skills in line with the speed of change. Expensive coding bootcamps for software development retraining are ten times over-subscribed.

CREATE COMMUNITIES OF PRACTICE

Finally creating Communities of Practice is challenging but fundamental (and fun!). Find the people in the organisation that are passionate about DevOps and bring them together. They will be leaders in the transformation.

CONCLUSION

For true DevOps success, the Enterprise attempt at DevOps needs to be consigned to history. Developers and Operations people need to align on the same incentives: delighting the customer and delivering features to them in a fast, secure and cost effective way.

In the future, DevOps will consist of managed, cloud-based services that, like the promise of the silent butler called Jenkins, will work quietly in the background and will fulfil their duties without asking.

The key determinant of the success of DevOps will be in the velocity achieved in delivering value to customers. Measuring and improving on this velocity requires the collaboration of not just colleagues in development and operations, but all parties engaged in making clients happy and businesses successful.

In the current climate, every business will experience pressures and challenges of some kind. Arguably Covid-19 gives us opportunities to accelerate and innovate more quickly. A thriving DevOps practice can tip the balance between lagging further behind the competition or moving at pace towards a successful future.

ABOUT DEVOTEAM

Devoteam is a leading consulting firm focused on digital strategy, tech platforms and cybersecurity. By combining creativity, tech and data insights, we empower our customers to transform their business and unlock the future.

With 25 years' experience and 8,000 employees across Europe and the Middle East, Devoteam promotes responsible tech for people and works to create better change.

Creative Tech for Better Change

CONTACT

Graham Zabel
Head of DevOps
Devoteam UK
graham.zabel@devoteam.com

Creative tech for Better Change