

# YOUR DEVOPS CHANGE JOURNEY

Where to start and what to do

---

“

Deciding on how to start a DevOps change journey can be a daunting prospect. This Executive Guide lays out the key areas, actors and activities that should align for successful change, along with some pragmatic approaches for change in the various roles that might collaborate.

**PETER-JOHN LIGHTFOOT** SOFTWARE DELIVERY ARCHITECT

assurity<sup>+</sup>

# CONTENTS

|   |          |  |           |
|---|----------|--|-----------|
| <b>INTRODUCTION</b>                                     | <b>3</b> |  |           |
| A note to the early adopters, champions and evangelists | 6        | Business Analyst   | 16        |
|   |          | Software Engineer  | 16        |
|   |          | Quality Assurance Engineer                                 | 16        |
|   |          | IT Operations Engineer                                     | 16        |
| <b>PART I – SEQUENCING</b>                              | <b>7</b> | <b>Start working towards continuously delivering value</b> | <b>19</b> |
| Before you start  | 8        | What are we changing?                                      | 19        |
| What are we changing?                                   | 8        | Who does what?   | 19        |
| Who does what?  | 8        | Leadership Team  | 19        |
| CEO   | 8        | Delivery Team  | 19        |
| Leadership Team   | 9        |  |           |
| Executive Committee                                     | 9        | <b>PART II – FUNCTIONS</b>                                 | <b>24</b> |
| Architecture Group                                      | 9        | IT Operations  | 25        |
| Start with people                                       | 11       | Quality Assurance  | 29        |
| What are we changing?                                   | 11       | Development  | 31        |
| Who does what?  | 11       | Architecture   | 33        |
| Leadership Team   | 11       | Team perspectives  | 34        |
| Team Manager  | 12       | Agile Team   | 34        |
| Project Manager   | 12       | DevOps Team  | 35        |
| Start building safety                                   | 15       | Agile roles  | 36        |
| What are we changing?                                   | 15       | Scrum Master   | 36        |
| Who does what?  | 15       | Product Owner  | 37        |
| CEO   | 15       |  |           |
| Project Manager   | 15       |  |           |
| Team Manager  | 15       |  |           |
| Architect   | 16       |  |           |
|   |          | <b>Traditional roles</b>                                   | <b>38</b> |
|   |          | Business Analyst   | 38        |
|   |          | Project Manager  | 39        |
|   |          | Change Manager   | 39        |
|   |          | <b>Leadership perspectives</b>                             | <b>40</b> |
|   |          | Team Manager   | 40        |
|   |          | Executive Committee  | 40        |
|   |          | <b>CONCLUSION</b>  | <b>41</b> |

# INTRODUCTION

---

# HOW TO APPROACH THIS EGUIDE

DevOps is a concert in which everyone has a part to play. It could be argued that some parts are more vital than others but, in the end, the best performance will be delivered when everyone is committed to the outcome.

This Executive Guide – or eGuide – considers some of these parts and their points of view. I will contrast some of what they know and experience today with what might await them in the ‘new world’ of software delivery in the digital business era.

I’ll describe this journey in its fullest form (i.e. as an organisational one), but it should be noted that success can be found in more localised scope. Of course, such success will have benefits that are similarly also more localised.

I hope this eGuide will help you see the overall landscape, provide an understanding of the basic building blocks for change and lead you to sequence them in a way that will maximise your chances for success.

The remainder of the eGuide is separated into two parts – the first part gives an ‘ideal world’ sequence of activities and actors that would contribute to success on a DevOps journey and the second dives into a somewhat deeper level of detail around these actors, their functional areas and some of the prominent challenges they might face.

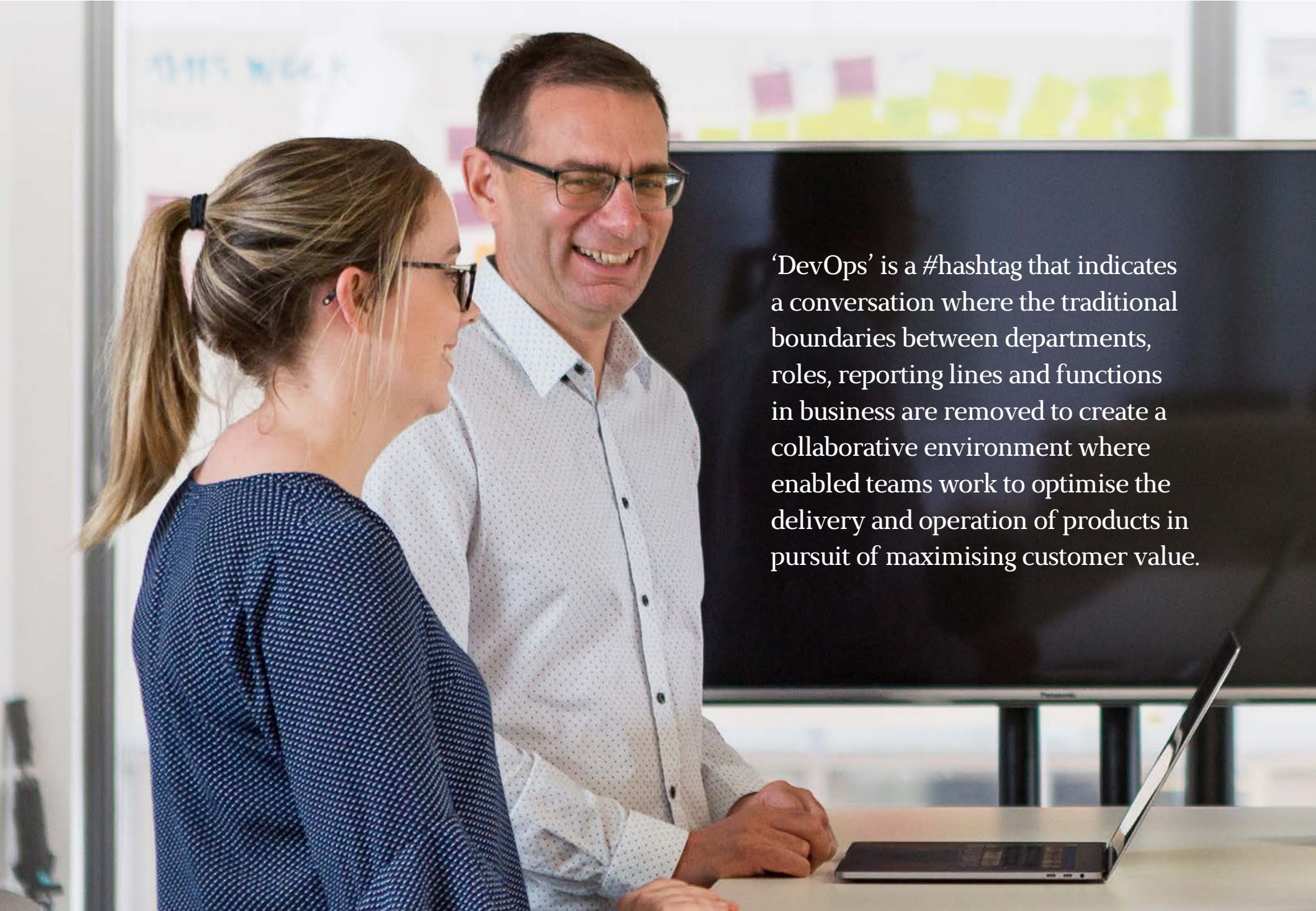
But first...

The word ‘DevOps’ means different things to different people. In the interest of ensuring we’re all ‘on the same page’, I’ll provide a definition of the term for the context of this eGuide...

**Peter-John Lightfoot**  
Software Delivery Architect





A man and a woman are standing in an office, looking at a laptop on a desk. The man, wearing glasses and a light-colored patterned shirt, is smiling broadly. The woman, with her hair in a ponytail and wearing a dark blue patterned top, is also smiling and looking at the laptop. In the background, there is a large screen displaying a dark interface, and a whiteboard with colorful sticky notes is visible.

‘DevOps’ is a #hashtag that indicates a conversation where the traditional boundaries between departments, roles, reporting lines and functions in business are removed to create a collaborative environment where enabled teams work to optimise the delivery and operation of products in pursuit of maximising customer value.

# A NOTE TO THE EARLY ADOPTERS, CHAMPIONS AND EVANGELISTS

---

## TAKE THE RISK. MAKE IT SAFE

It's with you that the journey starts. As with the movement itself, every DevOps journey will have someone at the forefront, prodding, pushing, challenging, leading, breaking ground where others would rather avoid risk – and bringing those same people with them by making it safe enough.

You are quite likely faced with seemingly overwhelming resistance from various corners. Guard yourself against establishing an 'us and them' situation as a defence mechanism, but instead seek out the right influencers and do enough to deliver benefits to them on **their** terms. Solve **their** problems. Increase **their** value. Not your own.

## INVOLVE LEADERSHIP

To bring this to fruition, in the end you're going to need everyone. Including leadership. Especially leadership. Irrespective of how you plan on getting there, at some point you will need the CEO (or equivalent) to buy in on the idea. Luckily, most CEO's are rational when you can demonstrate tangible benefits and DevOps is full of opportunities for doing exactly that.

## IDENTIFY OTHER KEY STAKEHOLDERS

Using the subsequent sections, understand the landscape requirements for change and consider how you might best collaborate with downstream areas, upstream areas and leadership. Then set about creating awareness of the benefits in the appropriate places to evangelise for change. Finally, follow through with continuous improvement, being resilient, persistent and transparent.

## PART I

---

# SEQUENCING

### WHERE TO START. HOW TO PROGRESS

The number one stumbling block to companies embarking on a successful DevOps journey – by far – is the uncertainty of where to start and how to progress. This section speaks to some of this uncertainty by bringing into perspective a few important notions about getting started on this journey.

Firstly, it considers some things that are important to understand before you even begin. Then follow a few areas that can be sequentially addressed to enable your change journey to begin. Note that these are not completed sequentially, but merely started sequentially. Also, the word ‘journey’ is more apt for describing what lies ahead than ‘transformation’, as the latter implies that there will be an end.

# BEFORE YOU START

## INVOLVING LEADERSHIP EARLY

I won't pretend that everyone (or even most people) will start their DevOps journey with the full support of the CEO and Executive Committee/Steering Committee. Nevertheless, whether at the outset or later, the leadership team will at some point need to become intimately involved in changing the business. That is, at one point or another, the changes resulting from your DevOps initiative may begin to affect organisational structures, funding models, operational models and even strategy.

If you are part of the leadership team, then this section is important to you. If you're not, then it is your solemn duty to help bring the message of this section to the members of the leadership team, whether directly or indirectly.

## WHAT ARE WE CHANGING?

A company is not unlike a big ship – and changing it is not unlike steering that ship. To turn a big ship, you need a big rudder. The rudder used to steer a company is a shared vision, strategy or culture – or a combination of those – that guides the collective on their way. Start building that rudder by focusing collectively on how to shift the perception of the organisation away from a collection of silos (e.g. 'business', development, operations and support) and towards that of layers within a single value stream (e.g. leadership, discovery, product direction, design, delivery and organisational support).

## WHO DOES WHAT?

### CEO

If you're reading this, chances are that your leadership team has some legacy with respect to how they approach business management. Typically, software delivery would have historically been perceived as the responsibility of a separate department and not part of the core business. However, in a digital business world – even if your business is t-shirt manufacturing – you will be unable to deliver your core business if your IT systems should fail. This critical reliance on IT effectively makes IT part of your core business.

Read the remainder of this eGuide to get a feel for the kind of change that your teams will be going through as they'll need your full support. Then set about considering how to bring your leadership team to the same awareness and ensure that they're forging ahead as a unit.



### **Leadership Team**

Commit yourself to investing in upskilling as and when it's required – because it will invariably be required. Where appropriate, start with training in Lean Systems Thinking and Agile Mindset for yourself and management.

Gain a deep understanding of the principles and techniques of servant leadership and encourage those to propagate to all leadership positions in the organisation.

### **Executive Committee**

Ensure you have a thorough understanding of the impact and magnitude of the change required and then make the commitment. Once you set off on this journey, the question of commitment cannot stand in the way of support that may be required.

For example, in terms of the time investment required by delivery teams, the Pareto Principle (80/20 rule) may be used as a rough guide – i.e. initially as much as 80% and going forward never less than 20%, will be spent not on doing the work, but on improving how the work gets done. Keep in mind that improving something systemic will almost always create better long-term value than shipping a feature.

If you find it necessary to 'get the ball rolling', start a pilot initiative, anchored around a business capability or product that lends itself to a single autonomous team (cross-functional across business analysis, development, quality assurance, IT operations and support).

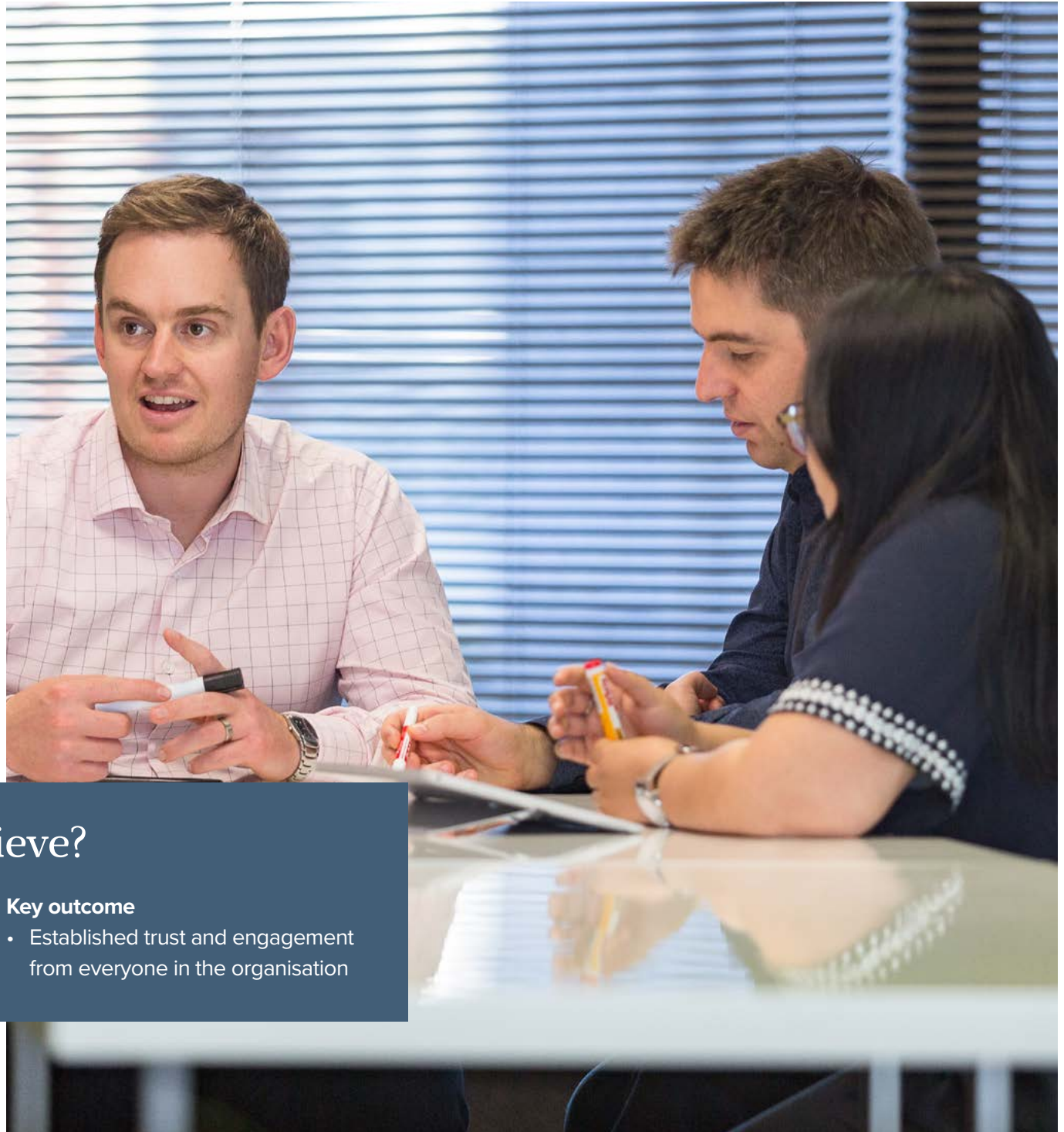
“

Initially as much as 80% and going forward never less than 20%, will be spent not on doing the work, but on improving how the work gets done

### Architecture Group

Perform a deep study of Lean Systems Thinking, Agile Mindset and the principles of Continuous Flow. Understand the technical requirements of Continuous Delivery, especially in the areas of Automated Infrastructure Provisioning, Configuration Management and Secrets Management. Use what you learn to prepare a clear vision of both the benefits and the kinds of changes that will likely be necessary to bring the organisation on this journey.

Your ability to articulate this vision will be instrumental in supporting critical leadership buy-in. Keep your architecture modular and Agile by describing the seams between functions on visual artefacts that support easy communication and collective understanding.



## What do we want to achieve?

### Key output

- Clear and unified messaging from leadership across all business areas

### Key outcome

- Established trust and engagement from everyone in the organisation

# START WITH PEOPLE

## **SERVANT LEADERSHIP. SELF-ORGANISING TEAMS**

Most companies today are becoming increasingly more aware of the importance of paying attention to their employees. We spend a great amount of time and money on getting the best people to work for us, but then depend on only a handful of them to tell the rest what to do and how to do it.

Combining principles from both servant leadership and self-organising teams is simple and profound, but very counter-intuitive. It will take time to eradicate the habits ingrained by years of working in command-and-control structures, but the key is consistent messaging and open communication.

## **WHAT ARE WE CHANGING?**

Encourage the modernisation of traditional role boundaries by shifting the collective focus away from job titles (e.g. Project Manager, Team Manager and even Product Owner or Scrum Master) towards function (e.g. governance, people management, product lead, craft lead, team captain). This is a fundamental keystone shift to improve the way that people interact and collaborate.

## **WHO DOES WHAT?**

### **Leadership Team**

Give the teams responsibility and authority to deliver business value and to use the tools, techniques and processes needed to do so.

Where appropriate, move to an integrated service model where teams are structured around business capabilities, products and services rather than IT projects or technologies.

The nature of self-organising teams and servant leadership makes this part of the recipe entirely subjective – a matter between teams and leaders. Be authentic and sincere and focus simply on creating an environment for teams to thrive in. Look to books like *The Five Dysfunctions of a Team* (Patrick Lencioni) for insights into what makes teams great. Another good place to start is with the first principle of Modern Agile (Joshua Kerievsky) – Make Safety a Pre-Requisite.

### Team Manager

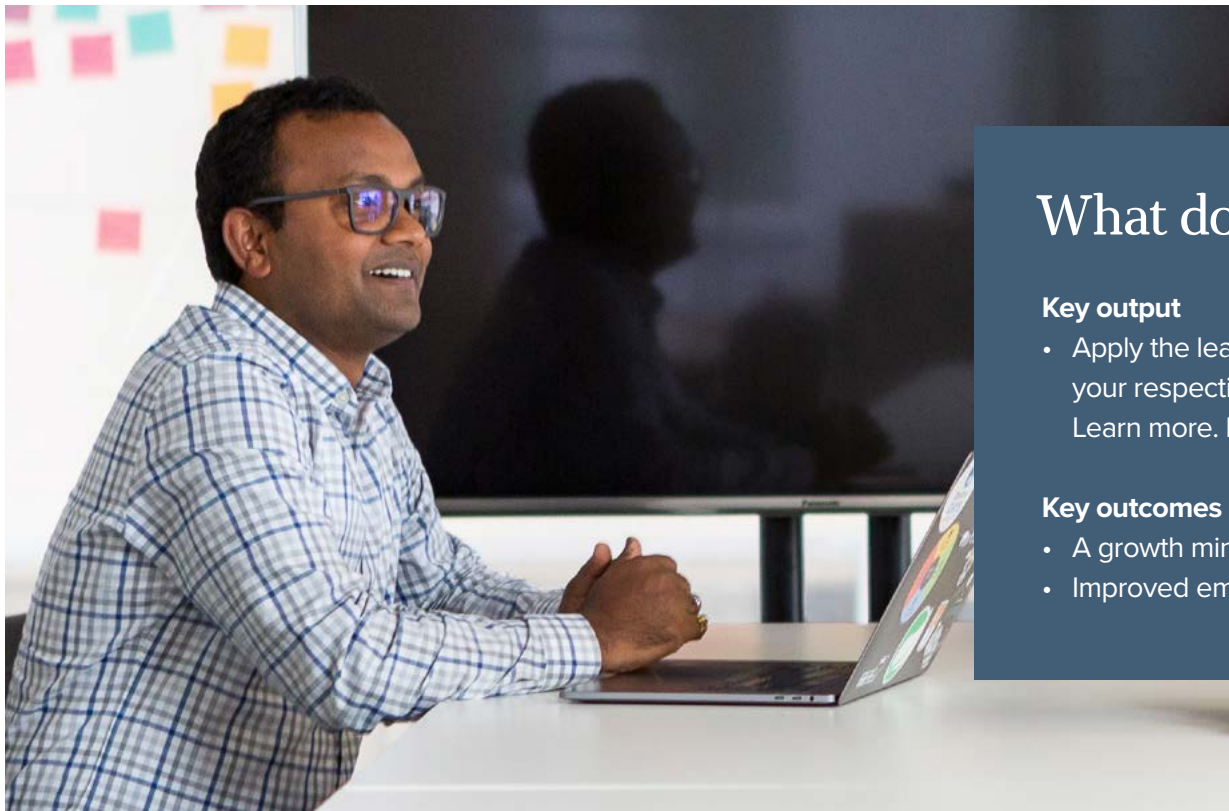
Support people to upskill where needed. As an example, in quality assurance, encourage and support movement from analyst (QA) to engineer (QE). Coding skills and automation capabilities will continue to become increasingly more needed and valuable, so encourage and support growth in those areas whenever it is appropriate. QA and IT operations will usually be among the first areas that could benefit from training and education in this area.

### Project Manager

Study the basic principles of Lean Systems Thinking, Lean Software Development, Agile and Modern Agile. Get to know the mechanisms and principles used by these disciplines to improve the working environment from both a people and a processes perspective. They will be key in creating a landscape that is equally navigable by everyone involved in your engagements.



[Refer to figures 1a and 1b](#)



## What do we want to achieve?

### Key output

- Apply the learnings from Lean and Agile to your respective immediate environments. Learn more. Rinse and repeat

### Key outcomes

- A growth mindset
- Improved employee engagement

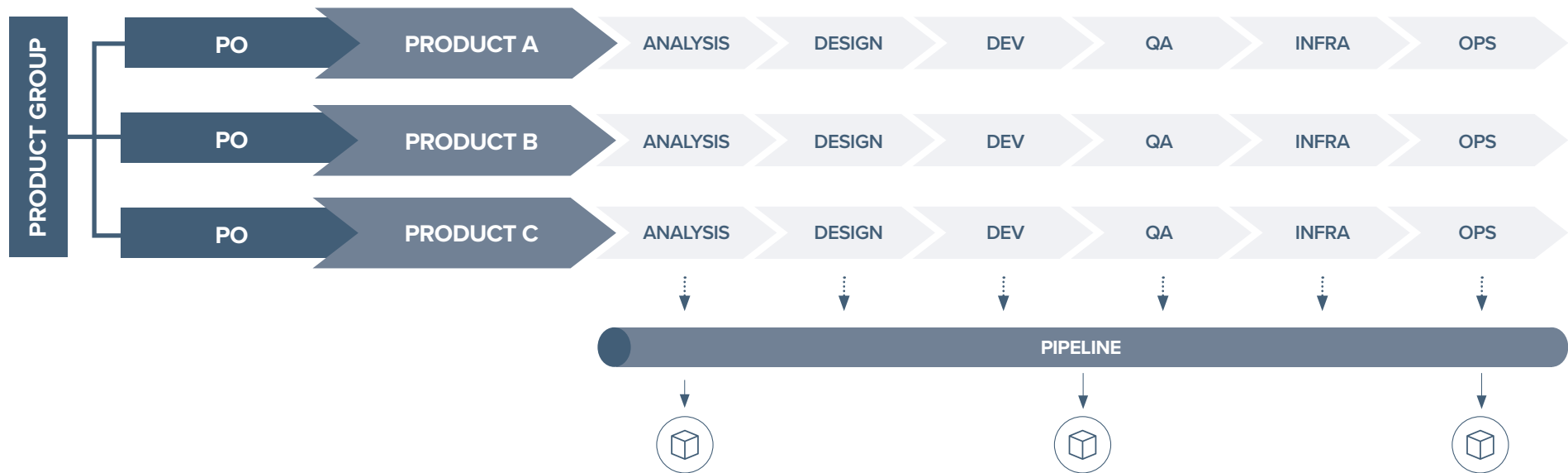
**FIGURES 1A (this page) AND 1B (next page)** show conceptually how DevOps can be used to improve the flow of work through stable, cross-functional, autonomous teams and delivery pipeline automation. Where traditional teams are kept disparate through often divergent management goals and KPI's, we instead look to create environments that help teams focus on finishing and shipping the prioritised work in progress.

**FIGURE 1A: TRADITIONAL – PROJECT-BASED**





**FIGURE 1B:** CONTEMPORARY – PRODUCT BASED



# START BUILDING SAFETY

## WHAT ARE WE CHANGING?

The primary mindset shift required for building safety is to move away from the notion of punishing failure towards accepting that failures are inevitable and instead celebrating when they're discovered – and rewarding ourselves when we learn from them.

In addition, this will encourage the creation of better safety mechanisms to deal with those failures. It is only when we don't have to ensure our own defence that we become free to address the true cause of a failure.

## WHO DOES WHAT?

### CEO

Read below about how Project Managers and/or Team Managers will be looking to institute mechanisms for ensuring that everyone has the authority to stop production when safety or quality concerns are raised. Consider providing teams with your phone number so that anyone can call to report if these measures aren't being honoured. That's what Paul O'Neill did at Alcoa.

### Project Manager

The '**Andon Cord**' is a mechanism (introduced by the Toyota Lean Manufacturing System) that can be used by anyone, whenever a defect is discovered, to bring the entire production line to a halt.

This aims to ensure that defects are not propagated to downstream work centres, but instead are resolved at the earliest possible opportunity. Understand this mechanism and consider how you might recreate it in your environment.

### Team Manager

Create a culture that principally values learning and growth and disapproves of blame. That is, support the implementation of the Andon Cord mechanism described above by ensuring that, first, everyone knows about it, knows how to use it and is encouraged to do so, secondly, anyone who uses it is commended for finding a defect and, finally, its use always results in learnings and improved quality. Feeling safe enough to stop the line when a defect is found is as important as finding the defect. If either of those are not present, defects will propagate down the line.

Get team members from downstream functions to start collaborating on work items early on in their lifecycle. The insights gained through such cross-functional collaboration enable everyone to build in better safety controls throughout the process.

### **Architect**

One of the biggest sources of resistance to a Continuous Flow way of working is the perception that risk must be mitigated by manual stage gates. Your ability to modernise the organisation's understanding of the requirements behind these legacy processes and translate them into automatable controls, checks and tests will greatly smooth the way for these principles to take hold.

Proactively ensuring that the right level of governance and compliance is built into automation requirements from the start and helping technology teams to communicate the effectiveness of those mechanisms will help others in the organisation to not fall into the trap of intuitively resisting the changes required to reduce the very risks they are fearful of.

### **Business Analyst**

Techniques such as ATDD (Acceptance Test Driven Development), BDD (Behaviour Driven Development) and SBE (Specification By Example) will help to make acceptance criteria relevant and improve product quality early on.



[Refer to figure 2](#)

### **Software Engineer**

Micro-testing and TDD (Test Driven Development) will help improve code testability, maintainability and overall quality.

### **Quality Assurance Engineer**

Automated deployment checks and automated testing (functional, UI, performance, compliance, security and acceptance) will help shift quality concerns 'to the left' through earlier discovery of defects and reduction of risks.

### **IT Operations Engineer**

Production monitoring, based on key performance metrics and health telemetry, will help to improve the collective awareness of product quality. Study, implement and improve the use of techniques like infrastructure as code, automated provisioning and automated deployment to create consistent, repeatable, automated processes for infrastructure provisioning and application deployments.

A close-up, profile view of a man with short brown hair, a beard, and glasses. He is looking down and to the right, presumably at a computer screen. The background is blurred, showing what appears to be an office or lab setting with some green and blue lights.

## What do we want to achieve?

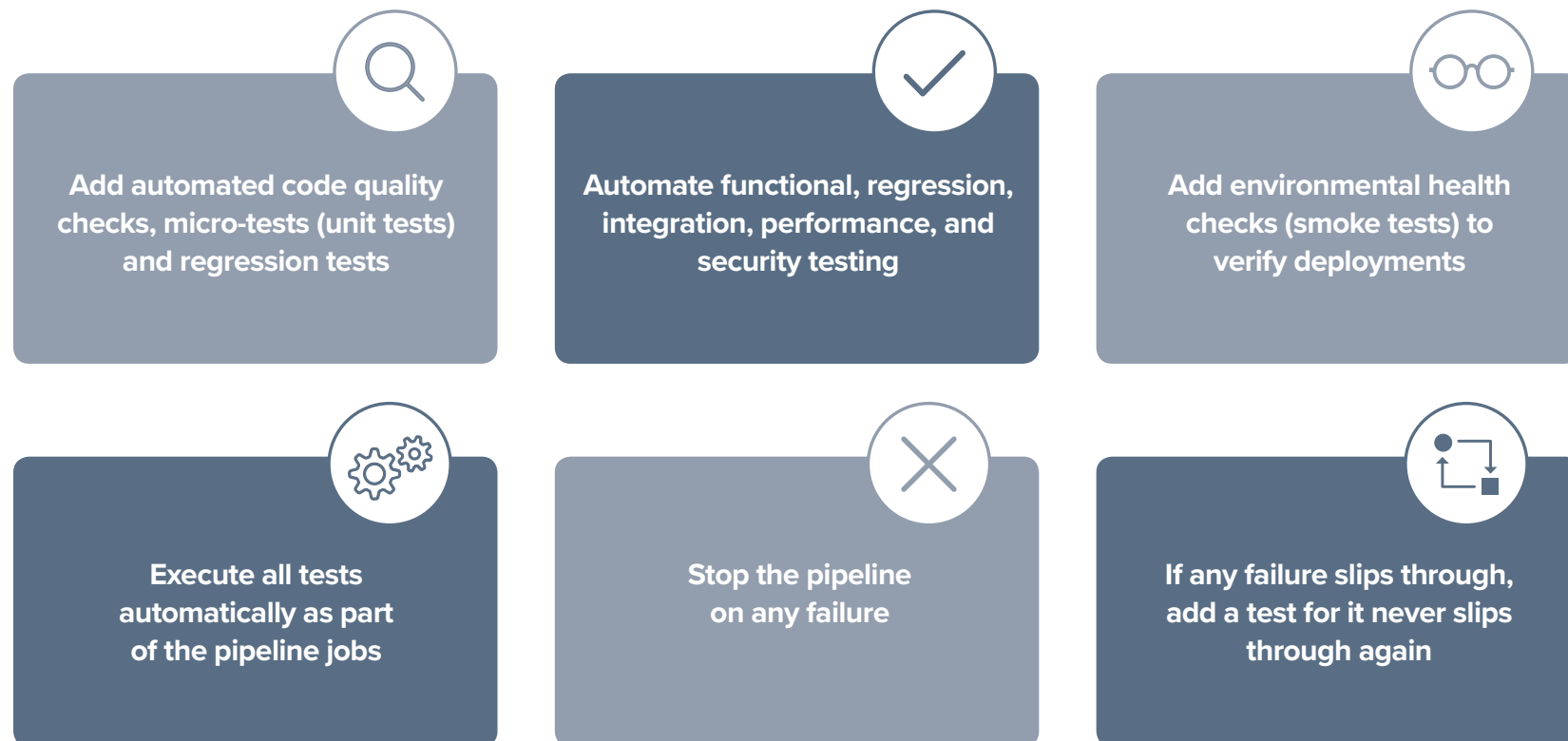
### Key output

- An environment of psychological safety – Modern Agile, Joshua Kerievsky
- Safety controls ‘baked into’ the delivery process
- Production monitoring

### Key outcomes

- Confidence that things can go wrong safely
- Confidence that every failure found will never be repeated
- Safety-driven culture

**FIGURE 2** highlights key considerations for delivering quality work consistently, reliably and safely.





# START BUILDING A LEARNING CAPABILITY

## WHAT ARE WE CHANGING?

As the culture of safety continues to grow, supported by a focus on rewarding efforts of learning from failures, encourage the creation of platforms and mechanisms for encoding those lessons within the organisation.

Use a cycle of continuous improvement to keep on strengthening your organisational learning capability by, first, discovering metrics that will improve your understanding, secondly, using those metrics to guide decisions on how to move forward and, finally, letting those decisions inform further refinement of metrics.

## WHO DOES WHAT?

### Leadership Team

Complement your customer research, needs analysis and/or design thinking work by creating a focus on understanding the customer experience by interpreting production monitoring data, business metrics, application metrics and health telemetry. Next, focus on ensuring that all learning is captured and codified.

Empower your teams to experiment and create a safe-to-fail environment to encourage a learning culture.

### Delivery Team

Use real-time production data – provided through production monitoring – to create information radiators that show system telemetry, deployment metrics, delivery metrics, business metrics and any other telemetry that will improve the collective awareness of system health, quality and customer satisfaction. For example, consider things like cycle time to production, customer use of features, wait time, defects, downtime, resource utilisation, mean time before failure, mean time to recovery etc.

Use source control. Use a single repository for application, test and infrastructure code. Use trunk-based development. Enforce coding standards. Use code review religiously. Experiment with mob programming.

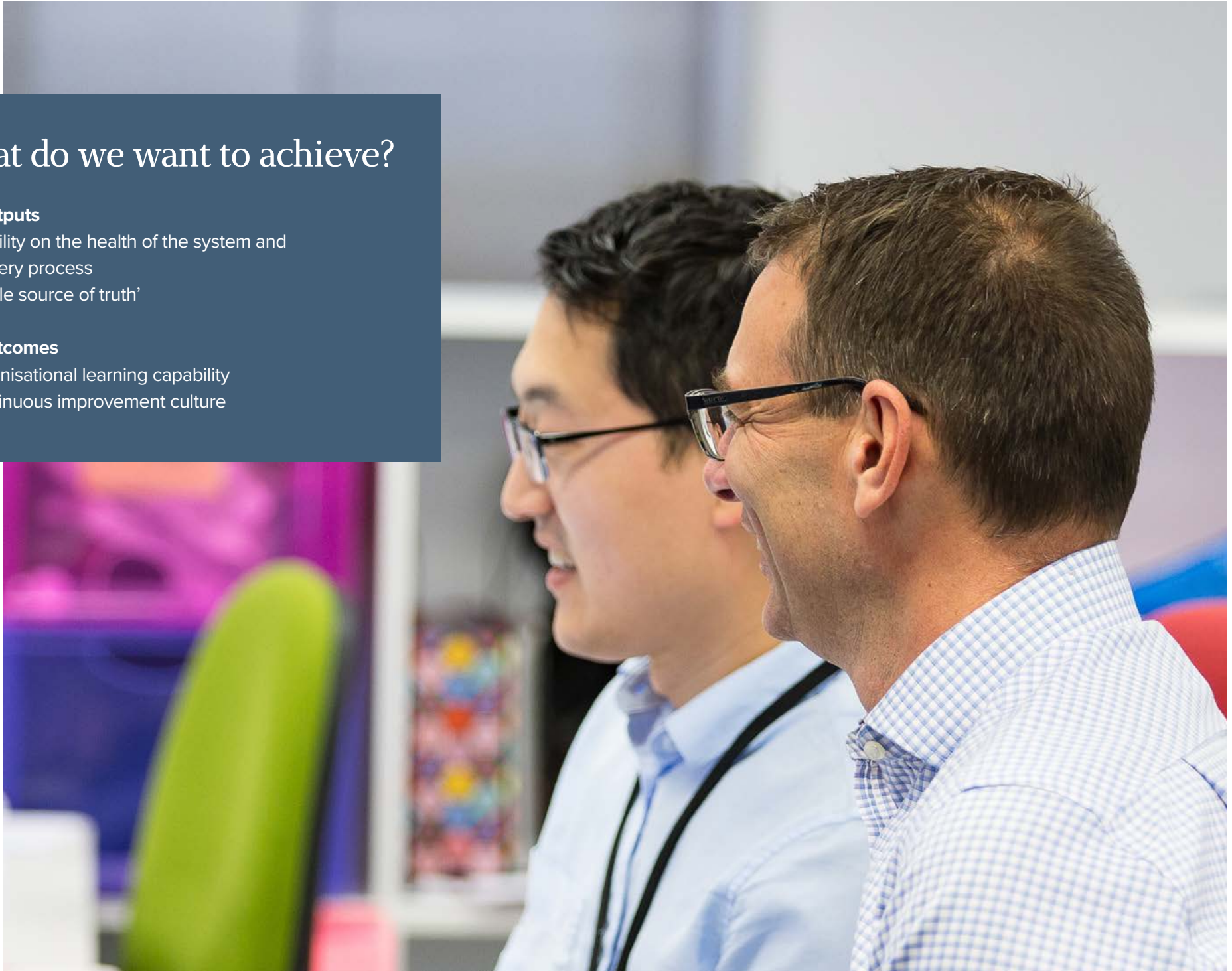
# What do we want to achieve?

## Key outputs

- Visibility on the health of the system and delivery process
- 'Single source of truth'

## Key outcomes

- Organisational learning capability
- Continuous improvement culture



# START WORKING TOWARDS CONTINUOUSLY DELIVERING VALUE

---

## WHAT ARE WE CHANGING?

WIP (Work in Progress) has been called ‘the silent killer’ because inactive WIP is inventory – and inventory attracts costs. In software development terms, these costs come in the form of context switching, stale information, technical debt or unrealised potential. Every time that processes create a handover point or a batch point (or worse, a batch handover point), that will create waste.

Creating a culture of continuous delivery is about eliminating handover points in favour of collaboration – and streamlining processes so that no work needs to be batched.

## WHO DOES WHAT?

### Leadership Team

Focus on anchoring every team around a single stream of work and tending towards a single work item WIP limit.

### Delivery Team

‘**Micro-development**’ is the technique of creating source code commits that are as small as possible (micro-changes), while being functionally appropriate and technically complete.



[Refer to figure 3](#)

CI (Continuous Integration) refers to both the technique of creating very frequent micro-change commits and the process of automatically integrating every such micro-change commit with all other preceding changes, then ensuring the integrity of the change by building and testing the application.

By inference, this implies that one uses a single (main trunk) branch on a single repository.

**CD (Continuous Delivery)** refers to the process of automatically producing a deployable artefact for every micro-change commit that is successfully integrated. These artefacts are cumulative so that every revision also contains all revisions that precede it.

These form the basis of further techniques for **production-based development** and can be extended in several ways. Most commonly, **deployment automation** can be used to automatically deploy the application to a target environment (e.g. TEST) and **continuous testing** will be used to run a suite of automated tests after such a deployment. Ultimately, **continuous deployment** would be employed to fully automate deployments to production for every commit that succeeds through the pipeline.





## What do we want to achieve?

### Key outputs

- Continuous flow mindset
- Continuous delivery process

### Key outcome

- Business agility

**FIGURES 3A AND 3B** illustrate the paradigm shift from an environment where work centres act as stage gates, to one where they contribute asynchronously to an continuous delivery ecosystem with automated compliance, safety and quality gates built in.





## PART II

---

# FUNCTIONS

In this second part of this eGuide, I'll be speaking to distinct groups of people separately, having a look at how you might be impacted and how you might best prepare yourself.

Feel free to skim over sections that aren't directed at your role, but I'd encourage you to circle back later to gain some insights into how your neighbours perceive and experience the journey that you are sharing.

# IT OPERATIONS

**Yours is arguably the most profound shift in your organisation's DevOps journey.**

Your status quo is typically based in an on-premise data centre where much of the legacy runs on dedicated tin and perhaps some of the modernised applications and services run on virtual machine infrastructure that is centrally managed. Some VM's have been allocated to development and testing as 'DEV', 'QA' and 'Acceptance' environments – and you might even have a 'Staging' or 'PRE-PROD' mirror of your 'PROD' environment. Changes to PRE-PROD and PROD are strictly controlled through CR's (Change Requests) and CAB (Change Approval Board) meetings, which also include requests for new VM's.

## **A MANUAL WORLD**

Risk is managed by batching changes into releases and allocating planned maintenance windows to deploy and patch system applications, but deploys are entirely manual. That is, scripts are run by hand to distribute and install application artefacts and the output is manually checked.

Most such batched deployments result in extreme hours and some incidents that must be resolved to get the system into a stable state again. Once stable, monitoring of the system comprises mostly of system resource metrics and perhaps periodic manual checks of application error logs.

“

Monitoring of the system comprises mostly of system resource metrics and perhaps periodic manual checks of application error logs

## MOVE TO AUTOMATION

Your ideal state is a place where you write and provide automation tools and scripts that enable self-service, instead of building and configuring systems on demand – where you provide detailed visibility of application and system telemetry to delivery teams instead of carrying a pager and where you guide and serve implementation needs during the design phase instead of putting out fires during deployment windows.

To get there, you'll need to pick up some coding skills and some knowledge of the software development discipline. Specifically, you'll need to learn how to use a SCM (Source Code Management) tool like Git effectively – and you'll need to come to terms with concepts of code quality (vs. code smells), peer review and test automation. Then, of course, you'll need the ability to script pretty much anything on your platform.

Chances are you already have this capability and just need the 'next level' of dev tools and techniques to support it.

You might also pick one or more tools to assist with the automation of infrastructure and application deployments, from the many that have in recent times been introduced into the market (e.g. Pivotal, Chef, Puppet, Ansible, TeamCity, VSTS, DSC, etc.).

Next, if (like most) you don't typically work closely with the developers or testers, start making inroads to changing that. Get yourself invited to demo/review sessions, even planning sessions if you can. Help where you can by sharing your valuable insights of the production environment during early discussions of features or by proactively providing resources (like VM's, ports, NIC's, RAM or CPU etc.) early on. Specifically, try to help by finding ways to assist testers with better (i.e. more production-like) environments before they need to start testing.

Finally, armed with your new automation engineering skill set and a fresh perspective on some of the difficulties experienced by the delivery team(s), find the most painful processes and automate them. As a rule of thumb, if something is difficult or painful, do it more often. That will usually provide the motivation and priority to automate it.

## SHARE. PAIR

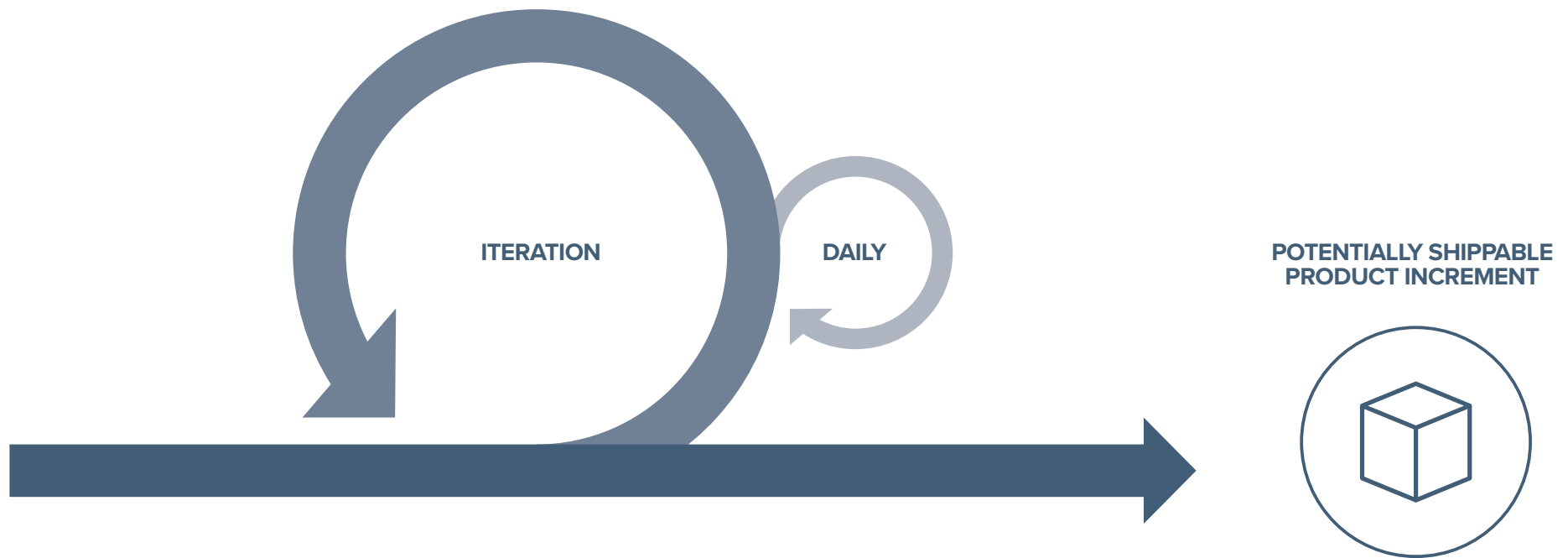
And don't forget to share everything you do. Not just with your team, but also with your upstream counterparts in development and test – and your leadership team too. Even better, pair with the developers to share and learn. Gaining a shared understanding of infrastructure automation will help the developers get to production quicker and help you learn development skills.

As a start for further reading, look at the James White Manifesto, with guidelines for infrastructure and automation. As an example, if something really is too hard or expensive to automate, it fails the IT purchasing rules. In other words, it should be replaced and/or factored out. Future purchases should be weighed against automatability to be fit-for-purpose.



[Refer to figures 4 & 5](#)

**FIGURE 4** illustrates the concept of iterative delivery. Each iteration should result in a ‘potentially shippable increment’.



**FIGURE 5** shows how contemporary Agile delivery falls short of its promise for fast response to market conditions because of a dependence on batched deployment processes that typically result from overly manual change processes.





# QUALITY ASSURANCE

I feel like the section for QA should have the word ‘lament’ in the title or – to paraphrase John – QA is patient, QA is kind. The unfortunate reality for you is that your patience and kindness usually end up being abused by the cowboy dev team and the marine corps ops team... and you tend to get squashed in the middle. And now, with the whole DevOps conversation, you harbour a silent fear of missing out, as if ‘DevOps’ means ‘no QA’.

In the worst-case world of traditional software engineering (that is, this may have improved for you if you’ve adopted some Agile practices), you would typically receive a specification written by an analyst who you probably haven’t met, from a developer on the floor above, along with the candidate binaries.

## LIVING IN HOPE, NOT CERTAINTY

You would try as best you can to make known the fact that the work item is already late in the hopes that, when it is eventually delivered after the deadline, someone will remember you pointed this out.

After pouring over the specification enough that you can author a test plan, you need to refresh, book or otherwise source an environment, install the candidate, make sure (as per the specification) that all the possibly affected integration points are fully functional. During this time, you might at irregular intervals be delayed while you wait for ops to provide or prepare some required resources. And then you test. You follow your carefully scripted test plan and you record results.

“

You might at irregular intervals be delayed while you wait for ops to provide or prepare some required resources

## **FORCED TO CUT CORNERS**

But you're working under this tremendous pressure caused by the compressed timeline and everyone else is chomping at the bit because marketing made promises and XYZ department still needs to do their UAT. So you cut some corners because there are still all the regression tests waiting. Oh well, maybe you'll skip some of those this time around.

It's a calculated risk and you feel you have your finger on the pulse of what this change is affecting. And so, in the end, you'll pass it along downstream, washing your hands of the consequences.

## **GET ALONGSIDE YOUR DEVS**

Now, imagine instead being involved in discussing the requirements and design with the rest of the team (including IT/Ops, Dev, Analysis and Business) at the outset. You gain intimate first-hand knowledge of all the details, influencing some of them right at the start to support better risk reduction later. You

work alongside Dev to write some automated checks for the new functionality, able to test it periodically as they deliver incremental pieces, using self-service production-like environments supplied by Ops.

With the basic checks in place and integrated in the code base (and therefore by extension also included in the deployment pipeline), you embark on some exploratory test sessions even as some final cosmetic changes are being made. Your exploration exposes a bug and, even while (or before) it gets fixed, you add an automated functional test for it.

To transition to this brave new world, you also (like Ops) will need to pick up some coding skills. There are many test frameworks and platforms to choose from so it would be best to approach it with the help of the other members of the delivery team, getting input from Dev and Ops. Beyond coding however, your focus will very much be on automation. That is, you'll need to come to terms with configuration management, test data management and, in some cases, service virtualisation.

## **AUTOMATE TESTING AS MUCH AS POSSIBLE**

Many of the test tools today will offer strong support for these, but you'll need to know what they are and how to use them. You will also need to be familiar with the principles of Agile and Lean Software Development and how these apply to testing (i.e. Lean Testing). Learn to identify and minimise waste in the testing process as this is an important part of maximising the value you add.

Again, like Ops, get involved in the delivery process if you aren't already. Simply helping where you can add value will in most cases get you noticed enough that you could get yourself invited to design or planning sessions.

Your aim with all this is to automate as much of the menial testing as possible so that you free yourself up to do more relevant exploratory testing. But remember, once you've automated something, it needs to become part of a build or deployment pipeline. If you don't have anything of the sort, start a conversation with Dev.

# DEVELOPMENT



You might think that DevOps basically just means that Dev needs to get more automation going so that they can deploy to production. You'd largely be wrong. I'm sad to say that, if you're really excited about DevOps, you're either going to be doing it wrong or be very disappointed by the reality.

The truth is that while there are major changes underway in the downstream areas – and, of course, those changes will influence you in some ways – yours is probably the least disrupted area. Still, your active participation will make the world of difference to the rest of delivery.

## **GAIN VALUABLE NEW INSIGHTS**

Yours is the skill that can empower QA and Ops in their respective journeys and you have insights to offer with regards to automation and how architectural design changes might enable big gains in testability and deployability for the team.

Working closely with the rest of the team across all these areas of the delivery stream will not only give a tremendous benefit to others, but will also reciprocate by letting you gain valuable new insights into how your system holds up in downstream stages of its lifecycle.



“

Yours is the skill that can empower QA and Ops in their respective journeys

If you apply principles of organisational learning and continuous improvement, these insights will take you a long way towards improving your applications for downstream consumption.

But not all your focus will be directed externally. There are three key areas of focus that will be critical for you as a Developer or Development Team to get exactly right. Developing them will be an iterative process – they all support each other so there's no way to just fix one in isolation. First, you need to be doing continuous integration properly. 'Integration' means that all code, from all developers, gets integrated into a single code base, compiles successfully and passes static code analysis checks as well as code-level unit tests.

## **COMBINING CONTINUOUS FLOW WITH CONTINUOUS INTEGRATION**

Conversely, 'continuous integration' means that every change made by every developer is automatically integrated and tested.

Secondly, you need to get to working on a single stable trunk without branching. Some would argue that you're not doing CI if you're not working on trunk, but I've seen plenty of CI servers pulling from various branches using complex rule sets and criteria.

Thirdly, you need to do TDD. And that doesn't simply mean 'write a test first'. Using a TDD approach, coupled with micro-testing techniques, will bring you ever closer to continuous flow. Since you have already ensured that test criteria are satisfied and you're contributing the automated tests at the same time as the production code, work items flow in one direction only, for the most part never failing QA. If for any reason you feel quite strongly that TDD is not for you, consider at least adopting SBE (Specification By Example) and/or BDD (Behaviour Driven Development) to introduce a measure of automated regression testing capability into your source code.

Combining continuous flow with continuous integration on a stable trunk will get you perfectly positioned for implementation of continuous delivery. Furthermore, supporting QA and Ops from that position will help the team to successfully move to continuous deployment.

# ARCHITECTURE

DevOps is nothing more (and indeed nothing less) than Software Delivery Architecture, with the primary intent of ensuring the Continuous Flow of work. It requires an alignment of changes across the areas of people (organisational structures, teams, mindset and culture), process (business processes, process controls, operating models, work methodologies and feedback loops) and technology (applications, data, platforms, infrastructure and analytics). This of course also means that it can be an incredibly complex problem space. In most organisations, the Architect is uniquely placed to have a deep understanding of all these areas.

Using standard architectural approaches, define the current and future state of your Software Delivery Architecture. Identify and construct a roadmap for change that considers all those areas (people, process and technology), and create simple visual artefacts that describe the areas where change will likely be needed first, taking a just-in-time, just-enough and continuous approach to the change journey.





# TEAM PERSPECTIVES

## **AGILE TEAM**

The Agile team as a unit is already well positioned along the journey of limiting WIP and tending towards a single item workflow. That is, if you truly have an Agile mindset and aren't just going through the motions of some Agile framework. Assuming it's the former (if not then this section just doesn't really apply), then it now falls to you to open your ranks to accommodate and support

some folks from Ops in the team. If you're using Scrum, you would probably want to think about coming up with a custom workflow model based on Kanban.

Come to terms with the notions (e.g. continuous flow) and techniques (e.g. continuous integration) described earlier, take a product focus for driving towards relevant customer value and build out your capability for rapid experimentation and rapid learning with fast feedback loops.

“

It now falls to you to open your ranks to accommodate and support some folks from Ops in the team

## DEVOPS TEAM

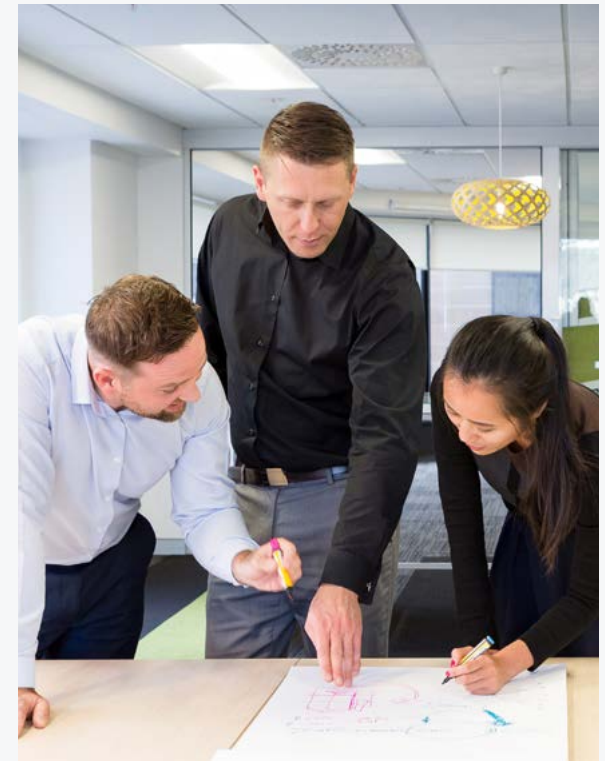
If the word ‘DevOps’ is anywhere in your job title or designation, you’re going about it all wrong. You should really get this eGuide through to your leadership team and start having some tough conversations.

That said, being part of an interim ‘tiger team’ initiative designed to generate momentum is both an honour and super exciting. What you desperately need to remember though is that your entire focus should be on making your ‘DevOps’ team obsolete.

To do so, you will set about creating patterns, processes and principles (as described in earlier parts of this text) that others in the more ‘traditional’ areas will learn to follow.

For example, one technique that I’ve successfully employed to promote the DevOps way of working – while still making sure that the group has a built-in ‘expiry date’, so they don’t simply become another silo – is using a ‘DevOps Working Group’ (made up of folks from all areas of IT and Delivery) to collaborate on a short-term initiative, to build out their Continuous Delivery capability as a product.

To better understand the do’s and don’ts of a DevOps team, have a look at the list of team topologies (both Types and Anti-Types) that the folks at DevOps Topologies have put together.



# AGILE ROLES

## SCRUM MASTER

You're focused on creating a space where your team is empowered to do what they're good at and, when necessary, you step in to remove impediments from their path so that they can get on with delivering value. In some respects, your concern will need to widen so that you can support the requirements of Ops, Infrastructure, Networking, Information Security etc. For instance, there could typically be a much stronger dependence on external providers in this space.

On the other hand though, much of what would normally have been perceived as impediments (i.e. dependence on Ops), will now fall within the domain of your autonomous team and will be dealt with as a matter of course.

But creating an environment where people are empowered to produce excellent work is not just about removing impediments. It's much more about making safety a priority. Your team needs the ability to find defects early, to call them out, to recover quickly when something goes wrong and, most importantly, to feel safe enough (given these) to experiment with changes.

Your next challenge is to really understand Lean Systems Thinking and Lean Principles... deeply. Understand why limiting WIP is important and why we want to tend towards a Single Work Item and Continuous Flow.

“

Going forward, you will need to work at guiding the team into the habits of experimentation and rapid learning so that you might end up in a hypothesis-driven environment

Guide the team towards identifying, gathering, interpreting and acting on useful process metrics. This will be key in helping them become self-aware and begin their journey of continuous improvement.

Finally, start to shift your thinking away from the concept of 'potentially shippable' as a measure of 'done'. You're moving towards shipping features to users much more continuously. The columns on your Agile boards will need to expand to include deployment to production and feedback received from real-world usage.

## **PRODUCT OWNER**

As with the Ops team, you stand to experience a very profound shift. As the first primary beneficiary of DevOps, you will not only get to enjoy the prize of drastically reduced feedback loops, but also be instrumentally responsible for creating and improving them.

Presently, you probably look to the Review session to verify that something has been built that will satisfy what you think the customer wants. Going forward, you will need to work at guiding the team into the habits of experimentation and rapid learning so that you might end up in a hypothesis-driven environment. Then, you will look to a deployment and the subsequent feedback to verify that something was delivered that does satisfy the customer and you'll be able to use empirical data to inform further changes or features.

To get there, the team will need a considerable amount of time to invest in themselves, their processes and their technologies. Your challenge will be to support them so that they're able to do that.

Having said that, experimentation in this context does not necessarily need to be of a purely technical nature (requiring a deploy to provide feedback, that is), but could even be as simple as a survey to establish preferences about a feature. Focus on learning Design Thinking techniques and finding ways to grow a culture of experimentation and learning within the team and you should be well rewarded when their delivery capabilities kick into the next gear a little way down the line.

# TRADITIONAL ROLES

## **BUSINESS ANALYST**

While Business Analysis undoubtedly remains a vital input, it rarely acts directly to put value in the hands of a customer. With the surrounding pressures driving towards such outcomes (i.e. faster time to market), common understanding and upfront agreement needs to be reached with regards to this function, in operational terms.

You will need to decide whether to keep Business Analysis as a separate and distinct upstream contributor to the delivery team. It is not uncommon to provide this analysis service to produce the work for delivery teams. However, keeping such a shared upstream resource pool is an anti-pattern that should be avoided if possible as the

resulting point of hand-off inevitably is also a point of push-back and contention – and thus a potential bottle neck.

The alternative (and recommended) approach is to decentralise and democratise the Business Analysis function, with various people and roles contributing to the analysis of work item requirements. In such situations, the person in a ‘traditional’ Business Analysis role is typically moved into one of Product Owner or Service Owner or similar.

The mid-way position of including a dedicated Business Analysis function as part of the autonomous delivery team is, of course, an entirely valid option, given that it was reached within the framework of a subjective decision reached by the team.



“

Going forward, you will need to work at guiding the team into the habits of experimentation and rapid learning so that you might end up in a hypothesis-driven environment



## PROJECT MANAGER

In a stereotypical world, one might see a typical waterfall-to-scrum transformation go something like this... Project teams become Product teams, Business Analysts are dedicated to specific teams as Product Owners and the Project Manager becomes Scrum Master.

In fact, in situations where there is a strong shift from project to product and stable teams form around business capabilities – negating the need for much (if any) resource coordination – it is not uncommon to see the project management role disappear completely.

There are of course no rules that say that either of these is how it should be done. In truth, such changes would be largely informed by individuals, their skill sets and organisational

goals and requirements – and the skills that a good project manager possesses remain invaluable when applied with relevance.

Being relevant simply means that the project manager (or product manager if that's the case) will know how to contribute in a way that complements and assists the delivery team. Be proactive in identifying and removing impediments from the team's path. Understand how metrics can be used to keep them aware of their performance in terms that are relevant to them and help them improve how they work.

In a subtle sense, your biggest challenge is to become included in the team.

## CHANGE MANAGER

Change Management is a big hurdle for many teams. The role changes from minimising change to optimising change. It has been said analogously that this is a move from having locked gates to having open gates with security cameras.

With the rigour that is built-in when DevOps is applied well, most changes can become standard changes, fully automated and audited by the DevOps processes. This includes multiple levels of change record, including request approval, code approval, automated test and deployment. The role moves towards one of understanding and mitigating risk. This also 'shifts left' like the testing role, with many risks averted at the planning stage. Risk itself can be reduced by moving from monolithic to micro service architectures (although there are inherent risks in the increase in the number of deployable components which require strong DevOps practices to mitigate).

# LEADERSHIP PERSPECTIVES

## TEAM MANAGER

Many who speak about DevOps refer to it as ‘a culture’. As the person who cares for the people that make up the delivery team, it will be up to you to help the team unlock this sense of culture in a way that will benefit both them and the company. You will therefore be challenged quite early on to have a strong idea of where individuals might need to move to, and/or upskill for, so that the team may come together as a whole.

Furthermore, while you are guiding the growth of individual team members, your primary focus should undergo a shift towards creating an environment of psychological safety. People need both the ability (tools and processes) to fail safely and the belief that it is okay to do so (learning culture), before they can create effective habits of experimentation and improvement.

Have an awareness of the challenges that various team members will face based on their roles as described in the preceding sections and use that awareness to guide people appropriately towards upskilling in areas of trunk-based development, continuous integration, coding skills, automation skills, continuous delivery, micro changes etc.

## EXECUTIVE COMMITTEE

The single most crucial realisation that you need to attain is that DevOps improves Technical Agility, with the eventual goal of creating Business Agility, but the latter is only realised when everyone understands how the former is accomplished. This means that you, your leadership team and your management teams, all need to have a strong grasp on the basic underlying principles of Lean, Agile and Continuous Delivery. To reiterate, make sure that everyone – not only the technical functions – are equally committed to the change.

Once the business benefits and the means for realising them are well understood, focus on creating the environment that is required for people to change and empower them to change structures and processes where needed.

# CONCLUSION

---

Bringing an organisation steeped in traditional management techniques into a space where people are trusted, empowered and freed to do what they're good at, is no small feat. The scale and impact can be quite daunting.

Still, recent figures show that most large corporate organisations are actively engaged in, or planning towards, such transformational initiatives. That so many organisations would take on such difficult initiatives is a testament to the tremendous benefits attainable through such initiatives.

I trust that this eGuide will in some measure assist with your initial concerns about the feasibility of taking on something that has such a wide footprint – and show that there are indeed some pragmatic approaches to igniting a wave of change throughout an organisation.

When you're ready to begin turning the wheels of change, bring someone on board with expertise and experience in change engagements. They would help with planning, coordination, guidance, coaching and generally caring for the change process.

It often makes the world of difference simply having someone available who isn't somehow vested in maintaining the status quo.

# assurity<sup>+</sup>



@AssurityConsulting



@AssurityNZ



Assurity Consulting Ltd

**assurity.nz**

## AUCKLAND

Level 6  
22 Fanshawe Street  
PO Box 106 949  
Auckland 1143

**t. (64) 9 354 4901**

## WELLINGTON

Level 7 Todd Building  
95 Customhouse Quay  
PO Box 25 440  
Wellington 6146

**t. (64) 4 473 0901**

## CHRISTCHURCH

Level 2  
53 Victoria Street  
PO Box 25 443  
Christchurch 8144

**t. (64) 3 379 9146**