LIMEPOINT

Ebook

Best Practice to Conquer Configuration Drift

CONTENTS

1.

WHO THIS GUIDE IS FOR

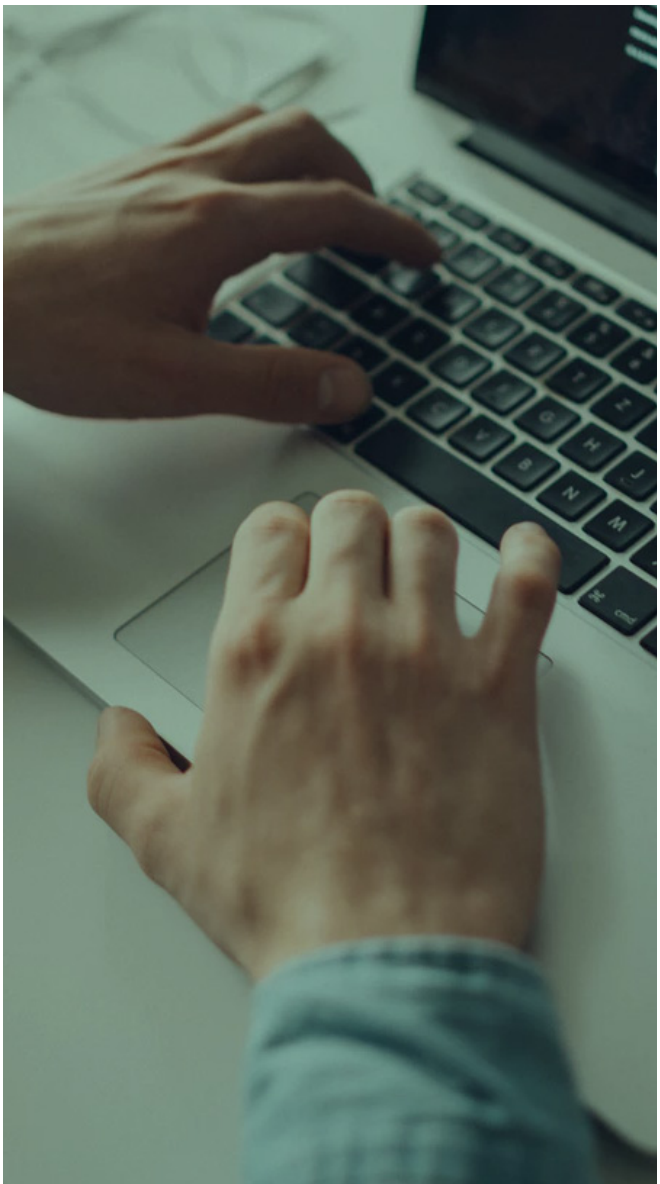
This guide is for ***IT specialists and operations managers*** of enterprises whose businesses run on Oracle technologies.

This guide is for you if you are a:

- Project or Program Director or Manager
- Architect or Subject Matter Expert
- CTO, CIO or IT Manager
- Environment or Operations Manager
- Oracle Applicatios Team Leader

According to Oracle, 'Configuration Drift is one of the main causes of instability and downtime of applications' [1].

This guide explains the causes of Configuration Drift, explores the various solutions and recommends best practice to detect, fix or prevent it.



2.

WHAT TYOU'LL GAIN FROM THIS GUIDE

This fully-referenced ***Best Practice Guide focuses on the impact and causes of Configuration Drift***, and explores the most advanced ways to overcome it.

In particular, you'll gain insights into:

- The true costs of Configuration Drift
- What industry experts recommend
- What automation can and can't do about Configuration Drift
- The vital role of advanced diagnostic tools
- how to achieve DevOps through Continuous Operation as well as Continuous Delivery of Oracle technologies

3.
CONFIGURATION DRIFT

3.1 What it is

Configuration Drift results from gradual, unplanned changes between environments in the Software Development Life Cycle, from Design and Build through Test and Release and finally to Production.

Although each environment is different, the configurations between them must be consistent, if the software is to perform as expected. Often even the tiniest discrepancy can make a major impact and it can be tough to find, let alone fix.

If these differences between environments persist, the result is Configuration Drift. If unchecked, Configuration Drift worsens with time. If not discovered, it's impossible to know where, when and how it may cause instability or a major failure – until it happens.

3.2 What it does

You might know the impact of Configuration Drift first hand: how it feels when a new software release fails on deployment.

When this happens, it won't be easy to find or fix the bug in Production. Even if you manage to pull it off, the fix won't be applied to preceding environments like Pre-Production, so your change will be overwritten with Pre-Production values next time you deploy to Production. Hence the problem will persist.

Configuration Drift is not a minor or theoretical problem. It's also not just an irritant in the gears of operation, or another glitch IT has to fix or another pain in the lives of SysAdmins.

Configuration Drift can wreak havoc in enterprises: it can cause customer-facing systems to crash, disaster recovery systems to fail and cause painful delays in critical projects or crippling cost blow-outs. Configuration Drift can bring businesses to a halt, frustrate new initiatives and damage reputations built with care over years.

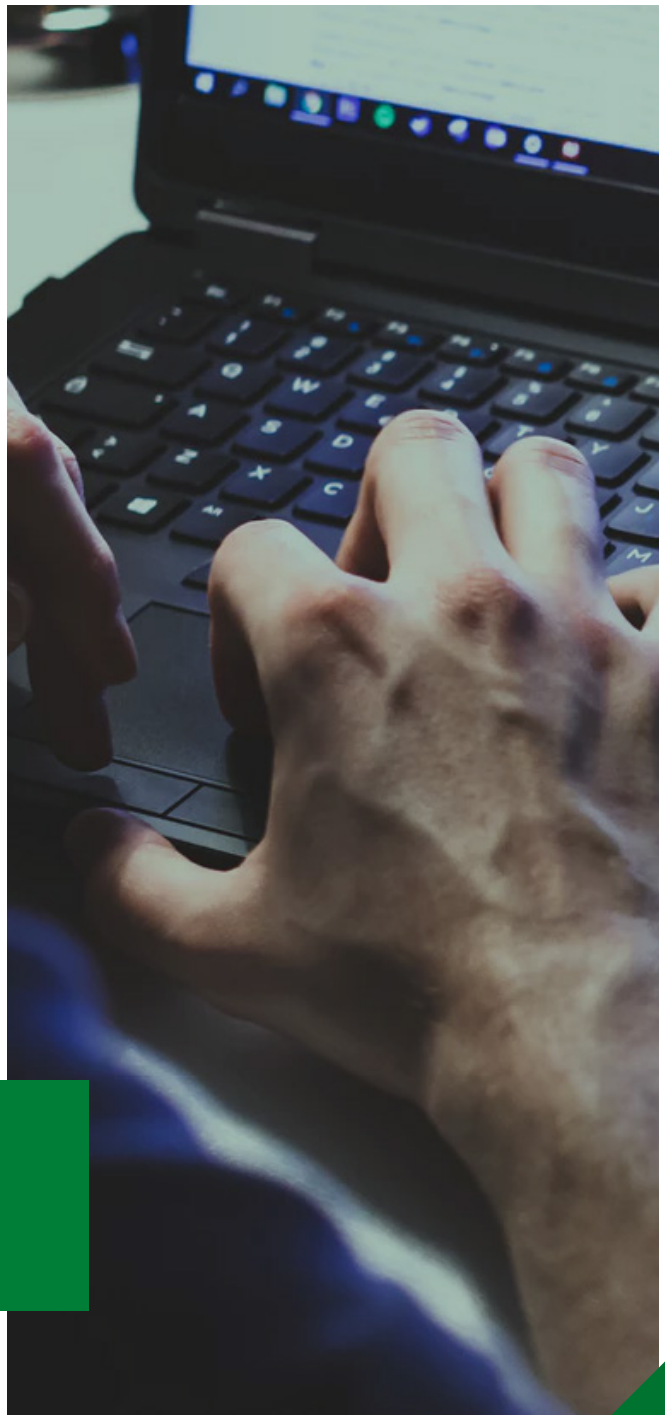
Yet, for most modern enterprises, Configuration Drift is a reality.

> *"If not discovered, it's impossible to know where, when and how Configuration Drift may cause instability or a major failure – until it happens.'*

3.3 How it happens

Configuration Drift occurs over time and is largely accidental. It happens through routine actions taken without consideration for the impact, such as when:

- Well-meaning team members apply a **minor fix, update a new software version or install a conflicting package** or service
- Developers **build environments manually,** which leads to discrepancies due to different processes being followed by individuals
- **A simple typo** in configuration in one environment can cause a discrepancy with the next, even when the same people are doing the task
- In the heat of a production incident, **an urgent fix is applied manually to the production system,** but is never retrofitted to Preproduction, Test or Development



3.4 What it costs

The causes of Configuration Drift may be benign - but its impact can be anything but.

Take a real example: a major financial institution involved in a large Oracle project. ***A single breakage caused by Configuration Drift took a full day to track down and fix, while 100 testers had to wait to gain access to the system.***

It doesn't take many like this before you're facing a 7-figure budget blow-out, according to IDC's 2014 Survey of Fortune 1000 Companies[2]:

- Average total cost of unplanned application downtime per year: $1.25 billion to $2.5 billion
- Average cost of an infrastructure failure: $100,000 per hour
- Average cost of a critical application failure per hour: $500,000 to $1 million
- Average number of deployments per month is expected to double in two years

The last point is critical, as a doubling of deployments will increase the frequency of Configuration Drift, and limit effectiveness of agile methodologies and your IT team's ability to deliver business initiatives.

Reducing the incidence and cost of failures and downtime is clearly critical, yet to do so and ***to make DevOps a reality, IDC says you'll need better tools***[2].

> *"Average total cost of unplanned application downtime per year: $1.25 billion to $2.5 billion'.*
>
> Stephen Elliot, IDC

3.5 What the experts say

Diagnosing and fixing the problems caused by Configuration Drift is time-consuming and expensive - if you try to do it downstream and manually. It could take weeks or months and, meanwhile, other teams are sitting around with the meter running.

As Lori MacVittie put is: '***the most difficult tasks in the network are not provisioning or configuration, but troubleshooting*** ... if you just deployed ten or fifteen different configuration changes across the core and per-app service infrastructure and suddenly something is "wrong", it's going to take some time to figure out exactly which configuration change caused the problem in the first place [3]'.

The smartest place to avoid these problems is upstream in Design and Build, by making your environments consistent using automation, and implementing practices that discourage unauthorised or un-notified environment changes.

It's like Mike Fal in DevOps and the DBA says: 'A key mantra about automation is that ***you do not automate to speed up your work, but instead automate to make your work repeatable and consistent.*** The benefit, of course, is that you remove work from your plate and allow it to be accomplished in a faster time frame 4.'

> *If...suddenly something is "wrong", it's going to take some time to figure out exactly which configuration change caused the problem in the first place [3].'*
>
> Lori MacVittie, DevOps.com

3.6 Can you just live with it?

Will the CEO accept the business impact of one or more systems failures caused by Configuration Drift? Probably not and, it's not just the impact of operational failure either: it's hobbling your ability to respond quickly, to gain market advantage through superior initiatives and to ensure BAU (Business As Usual).

You can't afford the lengthy delays due to software and infrastructure configurations drifting out of alignment, let alone the downtime caused by failures of back-up, recovery or failover systems. It's ommon knowledge that most unplanned drift discoveries are only made during analysis of such incidents.

Wouldn't it easier if you could just prevent Configuration Drift?

4.

WILL AUTOMATION **FIX CONFIGURATION DRIFT?**

4.1 What you can automate

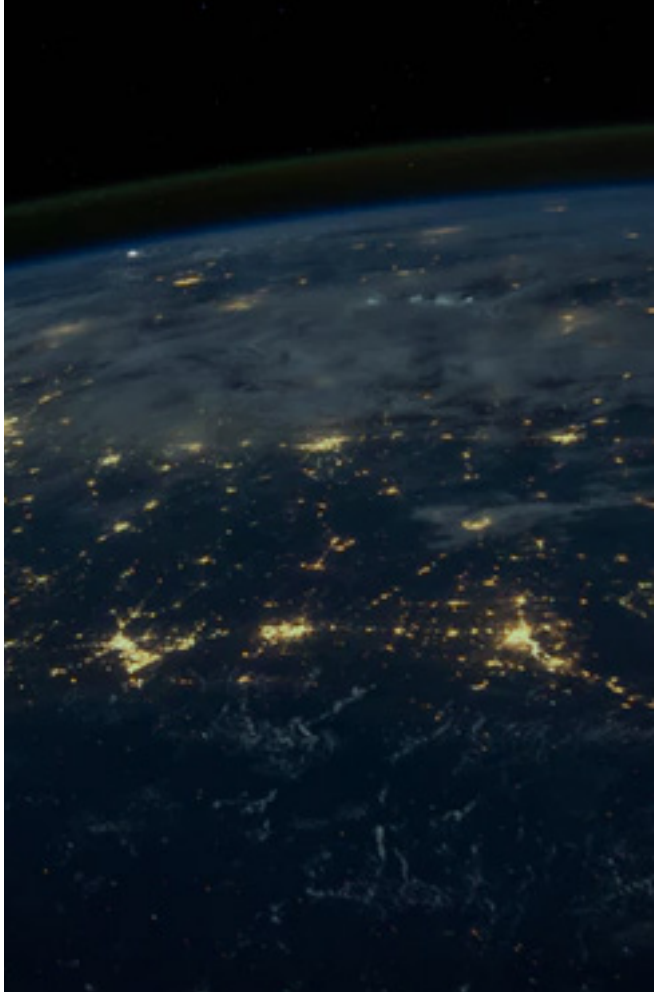
The catchcry of the DevOps movement is 'automate everything'. That's because a main cause of instability is manual processes that introduce human error.

'Here is the most ideal outcome of automating everything,' says Subbu Allamaraju at Making Cloud. 'Every node in the system is automated, has the right configuration, gets updated in synchrony with others, and everything is as it is supposed to be, right from day one. That's a naïve view 5.'

Of course it is. ***When you're managing hundreds of different nodes in various locations, there's every chance that you'll run into Configuration Drift***, or it will run into you: suddenly your disaster recovery process breaks, or an HA (High Availability) system stops working and customers are sending incendiary emails.

The goal of automation is that 'software is deployed in the same way that a car factory builds an automobile,' as Mike Fal puts it in Devops and the DBA: ***'Components should be standardized, builds made consistent, and tasks automated. The result is speed, yes - but speed that is the result of control and standardization*** [4].'

That's not an easy goal to achieve, because you can't just automate everything. Take a legacy system that was developed by programmers who left the enterprise years before, and is running on old hardware that's no longer supported by the manufacturer. You can't just port the application to the cloud and automate it into submission.



> *"Here is the most ideal outcome of automating everything: every node in the system is automated, has the right configuration, gets updated in synchrony with others, and everything is as it supposed to be, right from day one. That's a native view.'* [5]
>
> Subbu Allamaraju, **Making Cloud**

4.2 What you can't automate

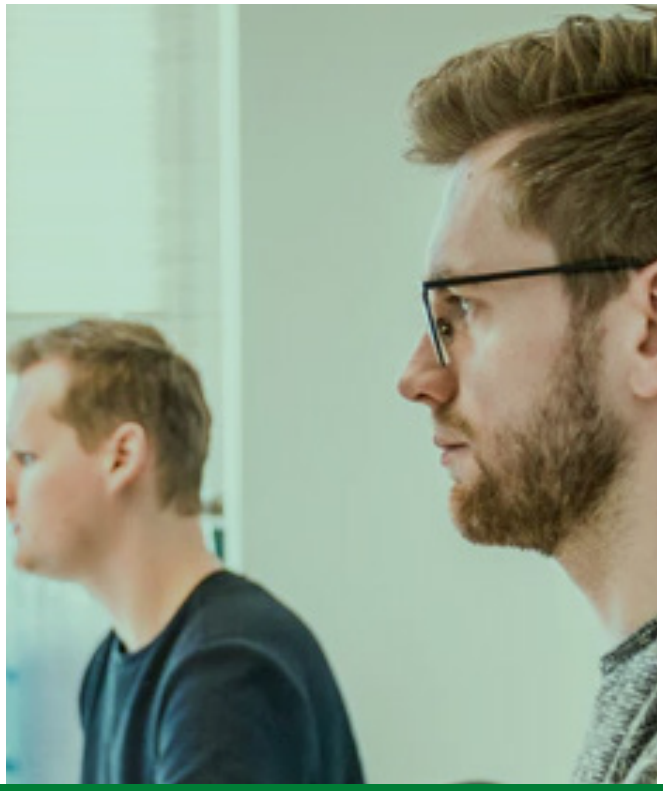
'To make a configuration error is human – to do it across a thousand servers is DevOps,'[6] John Balena at DevOps.com believes.

Of course it's an exaggeration but it makes the point. Think of the classic French film Mon Oncle, where M. Hulot fiddles with the settings in a plastics factory. Suddenly thousands of plastic pipes shaped like sausages explode out of the machine.

You can't automate what you don't understand, so you need an SME (Subject Matter Expert) to look at the legacy system and figure out what it's doing and how it works. That will create a gap in your production line, which is an ideal entry point for Configuration Drift. Automation in DevOps is an iterative process as code is promoted through the various stages (and back again if it fails), so there will be gaps.

There are gaps in communication as well, most of all in IT teams using the Waterfall Method of software development. ***Rigid demarcation lines create silos that made it difficult to manage Configuration Drift between developers, testers and production teams.***

'Ideally you want to develop with production in mind,' [6] says John Balena in DevOps Best Practices, 'but that gets harder and harder when both sides are changing and different teams are responsible for different environments and live in their "own worlds." (e.g. QA teams, release teams and production operations).'[6]



> *"Ideally you want to develop with production in mind, but that gets harder and harder when both sides are changing and different teams are esponsible for different environments and live in their own worlds.' [6]*
>
> John Balena, DevOps Best Practices

> *"If you try to replicate the automation solutions used by Facebook and Netflix, you'll quickly find yourself in a world of hurt. The investment you have made in DevOps and automation will blow up in your face - not because you automated, but because you attempted to automate too much [7].'*
>
> David Linthicum, Techbeacon

4.3 Is DevOps Automation the answer?

DevOps helps to break down those siloes and encourages close communication and collaboration between teams. The continuous integration pipeline has built-in checks for errors and tests for functionality. Once verified, changes can be deployed to a target set of instances. ***That's a huge advance in software development but it doesn't fix Configuration Drift, nor can you automate Configuration Drift out of existence.*** Automation is not a panacea; it is a great advance but needs well-managed processes and good drift detection tools to ensure that you get the best results.

'DevOps tool providers may sell automatio products that promise to make you the next Facebook or Netflix,'[7] says David Linthicum at Techbeacon.

These two companies are the examples often quoted by vendors extolling the virtues of DevOps Cloud. ***They usually forget to mention that Facebook and Netflix had the luxury of building new systems from scratch,*** as well as choosing the most advanced software platforms.

These start-ups had no existing or legacy systems, no mix of vendors to integrate with new systems and no existing customers. That's why Linthicum adds this caution: '***If you try to replicate their automation solutions, you'll quickly find yourself in a world of hurt.*** The investment you have made in DevOps and automation will blow up in your face -not because you automated, but because you attempted to automate too much.' [7]

Clearly automation of environment builds will make a major impact on Configuration Drift, if you can achieve the automobile production line that Mike Fal outlined earlier. However, as Lori McVittie made clear, ***it's the troubleshooting the environments once built that's the tough bit so, while automation is vital, it's only one side of the coin.***

5.
HOW TO FIX CONFIGURATION DRIFT

5.1 Monitor manually

Manual monitoring is an option often discussed, but it's not a practical one: you'd have to scan your servers as frequently as once a day, assuming your developers are using the Continuous Delivery model. Yet 'scanning' in this context is a cursory glance for obvious red flags that popped up since the last scan.

To do it thoroughly, you'll need to compare files in great detail, becaus even a small difference can make big impact but are very hard to find. Unless you have an interface that gives you good visibility into your systems, and the ability to compare the files without the need to drill down to minute detail, the people you've assigned to this task will soon start looking for new jobs. The bigger your network is, the more impossible monitoring becomes.

It pays to remember what Oracle said at the start: 'Configuration drift is one of the main causes of instability and downtime of applications.'[1] ***If undetected and uncorrected, Configuration Drift will, at some or another, cause system failure*** which will have major impact on your business, because manual monitoring is just not feasible.

5.2 Build or borrow tools

' ... build tools to regularly audit for drift,' advises Subbu Allamaraju at Making Cloud. 'Automation is expected to reduce drift, but like most things, automation too may be work in progress and shall have bugs. ***Use audits to discover the state of drift. Awareness is a prerequisite for mitigation***.'[5]

Allamaraju suggests ***extending the 'measure everything' maxim to include Configuration Drift so you'll always know what nodes or systems are in drift.*** The next step is to 'Wire up these metrics to your alerting systems so that the team gets alerted when drift is discovered.'

> *"Automation is expected to reduce drift, but ...automation too may be work in progress... Use audits to discover the state of drift. Awareness is a prerequisite for mitigation.[5] '*
>
> Subbu Allamarajut, Making Cloud

Do you have enough available staff to create all the drift definitions you need, and the templates and snapshot reports? You will be rare if you do. Of course, there is the option to use some of the community-based plugins and scripts, but this has drawbacks too:

- You may have to modify or extend the plugins and scripts to suit your environment
- You'll need to keep your collection of plugins and scripts updated
- Community-based plugins and scripts tend to offer limited support for the Oracle stack

5.3. Use Configuration Management

'What Configuration Management(CM) brings to the table is a means to obtain this knowledge,'[8] says Reid Vandewiele on the Puppet blog. '***Unlike custom scripts, CM should be running continuously ... and assert and report on the state for every system under management.'***

Vandewiele lists the ***benefits of CM for operations:***

- The confidence to know that changes are made to a well - understood field of systems
- The ability to audit a configuration state without having to create a custom scripts
- The kind of visibility into current configuration states that removes surprises, reduces unexpected downtime and speeds up the audit process

'A corollary of visibility in the context of CM is exposure and remediation of configuration drift,' he adds and says that CM offers the option of forcibly eliminating it, or alerting sysadmins to differences in configurations and allow them to decide on the best response. 'In either mode of operation,''he says, 'good CM gives you a means to anchor configuration drift, prevent ad-hoc configuration changes and keep your environment, as a whole, aligned with a single source of truth.' [8]

All three of these options may work, yet ***they assume you have a surfeit of IT resources***, and can allocate quite a few to fixing Configuration Drift. Most enterprises don't have this luxury.

6.

PREVENTION: **THE SMART WAY TO CONQUER** CONFIGURATION DRIFT

6.1 Use the smartest tool

Puppet, Chef and Ansible are the products used by many organisations for Configuration Management, and they go some way in making it easier to manage Configuration Drift. The only problem is, they're generic.

If your enterprise is based on Oracle technologies, you'll need specific products designed for Oracle - yet flexible enough to accommodate any legacy or third party technologies you may have in-house.

To date there has only been one real-time diagnostic and remediation toolset designed for Oracle environments that can do this. It's called DriftGuard and it's part of the LimePoint Automation Suite for DevOps.

6.2 Detect defects early

DriftGuard is built on the logic that you want to find defects early - before new code is released to your Production teams - not afterwards. Not only will this save you a chunk of post-event diagnostic time, it will eliminate the failure and impact on your business in the first place.

DriftGuard enables your team to:

- Compare one environment with another or with itself, to detect any inconsistencies
- Compare your roadmap with what has been built, and rectify any differences
- Gain a real time, in-depth view of all your environments (Oracle and non-Oracle as well as those built manually)
- Ensure that configuration changes have been done properly, by comparing before and after.
- Identify issues in configuration or systems audits, in days not months
- Detect unauthorised changes in any environments in a systematic and efficient way

> *"To date, there has only been one real-time diagnostic and remediation toolset designed for Oracle environments that can do this. It's called DriftGuard and it's part of the LimePoint's Automation Suite for DevOps.'*

6.3 Find & fix in real-time

If you haven't used DriftGuard to find defects before release, the next best thing is to find and fix them in real time Production without impacting business operation. DriftGuard lets you do this too.

Configuration management and drift detection form a vital 'early warning system' to alert to future failures and process inefficiencies that might otherwise remain hidden. ***DriftGuard also reveals who has gone outside of automated build practice, enabling redress and retraining to be targeted at specific staff members.***

In addition, DriftGuard lets you:

- Use scripted collections to find configuration drift in non-data & non-file-based databases that are otherwise opaquet
- Determine the root cause of defects in real time so you can rectify them quickly
- React faster by receiving alerts to configuration drift, automatically by email
- Save time by downloading all metadata for a Dimension into a single Zip file

6.4 Audit & secure

DriftGuard stores all configuration data for audit and quality assurance purposes, making it easy to ensure that your processes and people comply - with enterprise standards and those of industry or government bodies.

DriftGuard provides a clear audit trail of changes made when, where and by whom. In addition, it:

- Detects unauthorised changes in any environment, in a systematic and efficient way
- Raises alerts when it detects unintended changes to configurations
- Raises alerts when it detects changes that appear to be malicious in intent
- Alerts on the detection of changes to sensitive files you've selected for specific monitoring

The notification process ensures that key staff are advised of potential problems, so they can take action before serious damage is done. ***DriftGuard also has intelligent rules built in,*** to eliminate false positives that can arise during monitoring for malicious interference.

> *"Usually there are hundreds of defects in going live with Oracle deployments, including serious ones. With LimePoint, there was a single defect which had no customer impact. We found it using DriftGuard, fixed it and migrated to live production with no outage whatsoever.'* [9]
>
> Phil Horton, **Program Director, MaxGroup**

6.5 Harvest existing Configurations

This feature of DriftGuard makes it easy to ***harvest configurations from an existing environment,*** so you can duplicate it. DriftGuard imports configurations into MintPress, the other product in the LimePoint Suite, which produces an identical build instantly. ***This closes the loop on the DevOps principles that ccomponents should be standardized***, builds made consistent, and tasks automated – that software is deployed in the same way that a car factory builds an automobile, as Mike Fal puts it in DevOps and the DBA. [4]

DriftGuard's harvest function:

- Helps organisations migrate configurations from on-premise to cloud deployments
- Saves time and effort compared to building environments from scratch
- Allows for manual configuration changes made outside MintPress to be harvested and applied to other environments to ensure consistency

DriftGuard also captures undocumented configurations in the environment, which ensures that all configurations are known and documented.

6.6 Use with non-Oracle Systems too

While DriftGuard was designed specifically for Oracle environments, many enterprises have a combination of Oracle, non-Oracle systems and legacy systems. DriftGuard's data collector lets you import data files from other systems, and set its rules engine up to include these systems in the drift detection process. ***That makes it possible to monitor mixed environments for drift detection, and to apply DriftGuard's alerting and diagnostic capabilities.***

DriftGuard provides a number of standard reports, and also includes an extensible framework for creating your own reports for configurations of Oracle and non-Oracle technologies.

> *"... software should be deployed in the same way that a car factory builds an automobile. Components should be standardized, builds made consistent, and tasks automated. The result is speed, yes - but speed that is the result of control and standardization.'* [4]
>
> Mike Fal, **Simple Talk**

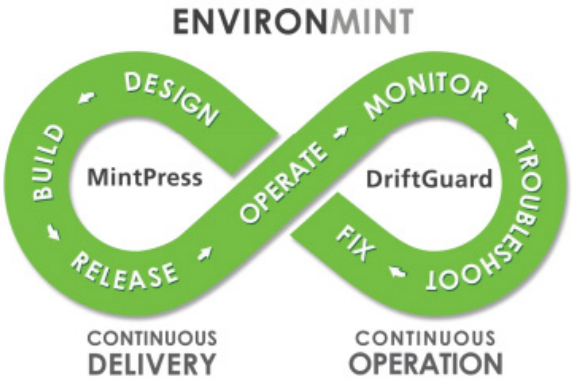
7.
DRIFTGUARD: **DEVOPS REALISED**

6.1 Use the smartest tool

DevOps has been a goal for many enterprises for some years, yet realising it has been theoretical or patchy for most.

Achieving the Continuous Delivery part of DevOps is now possible using smart automation platforms, such as MintPress, the other product in the LimePoint's Automation Suite for DevOps. Yet, as we've seen, while automation is vital for building consistent environments, it's no guarantee o preserving them or avoiding Configuration Drift.

With the development of DriftGuard, the other vital part of DevOps, Continuous Operation, is now possible. DriftGuard provides the critical diagnostic capability to find and fix or prevent defects in environments, in real time, before or after software release.



Together, MintPress and DriftGuard make it easy and costeffective for enterprises to build high quality, consistent environments for Oracle quickly in-house, and to find and fix or prevent Configuration Drift in Oracle or non-Oracle environments, in real time.

8.
ABOUT LIMEPOINT:

LimePoint is a highly experienced Oracle specialist based in Australia.

Our people have ***over 20 years' experience with large scale, multi-million dollar Oracle deployments*** in Finance, Government, Education, Retail and Utilities. We've helped them streamline their deployments, achieve DevOps and improve business agility and value.

What also sets LimePoint apart is our ***Automation Suite for DevOps for Oracle.***

Developed in-house, our ***Automation Suite*** comprises two products: ***MintPress*** which designs, builds and deploys high quality Oracle environments in hours not months, and ***DriftGuard*** which detects and fixes or prevents Configuration Drift in Oracle and other environments, in real time. Available together or separately, ***MintPress*** and ***DriftGuard*** make the impossible a reality: high quality environments that are fast to build and easy to manage and troubleshoot, in-house.

LimePoint has rare ***Oracle expertise across the whole Oracle stack and the full environment life cycle***. We work from project planning to financial establishment and stakeholder management, through to implementation and measurement, providing solutions that are fit for purpose and support innovation, business value and competitive advantage for your clients.

Whether you choose our advanced ***Automation Suite, our deep Oracle expertise or both,*** LimePoint is the considered choice for complex or diverse Oracle environments.

9.
ABOUT **THE AUTHOR**

GORAN STANKOVSKI

Goran is our Founder and CEO, and the innovator behind our technology.

He's a recognised thought leader in Oracle architecture and had an international career with Oracle before forming LimePoint. Goran is the mentor for the LimePoint technical team and the brains behind MintPress, our advanced tool for Continuous Delivery and automation of Oracle deployments. Goran has presented at Oracle Open World several times and is the author of many articles about Oracle architecture, automation and performance.

10.
REFERENCES

1. ***'Answers to Your Common Oracle Database Lifecycle Management Questions'***
Scott McNeil. Oracle Enterprise Management & Oracle Management Cloud Blog. 12 November, 2012

2. ***'DevOps and the Cost of Downtime: Fortune 1000 Best Practice Metrics Quantified'***
Stephen Elliot. IDC Insight October, 2014

3. ***'How Deploy Frequency Impacts Infrastructure Stability'***
Lori McVittie. DevOps.com. 26 February, 2015

4. ***'DevOps and the DBA'***
Mike Fal. Simple Talk. 08 May, 2015

5. ***'On Influencing'***
Subbu Allamaraju. Making Cloud. Making Cloud. 07 August, 2016

6. ***'DevOps Best Practices: Break Down Silos, Avoid Drift and Optimise for Flow'.***
John J Balena. DevOps.com. 16 May, 2014

7. ***'DevOps automation best practices: How much is too much?'***
David Linthicum. TechBeacon. 04 August, 2015

8. ***'How To Sell Configuration Management to Your Boss'***
Reid Vandewiele . Puppet Blog. 24 February 2014'

9. ***'MaxGroup Case Study LimePoint 2016'.***
MaxGroup is an Australian corporation who asked for names to be changed for privacy reasons

CONTACT US

www.limepoint.com

info@limepoint.com

LIMEPOINT