SAS® GLOBAL FORUM 2020 #SASGF

Paper 4649-2020

Exploring DataOps in the Brave New World of Agile and Cloud Delivery

Shane Gibson, PitchBlack

ABSTRACT

DataOps is the new black, providing a combination of DevOps and agile practices to automate the creation and management of data, analytics, and visualization on cloud platforms. DataOps is revolutionizing the way we deploy and manage data platforms, and SAS® Viya® is embracing DataOps within it's core. In this session, I outline some of the key components required to adopt the new DataOps way of working and provide examples of the benefits each component will deliver. If you want to understand what DataOps is and how you should leverage it to build and manage better data platforms and processes, then this is the session for you.

INTRODUCTION

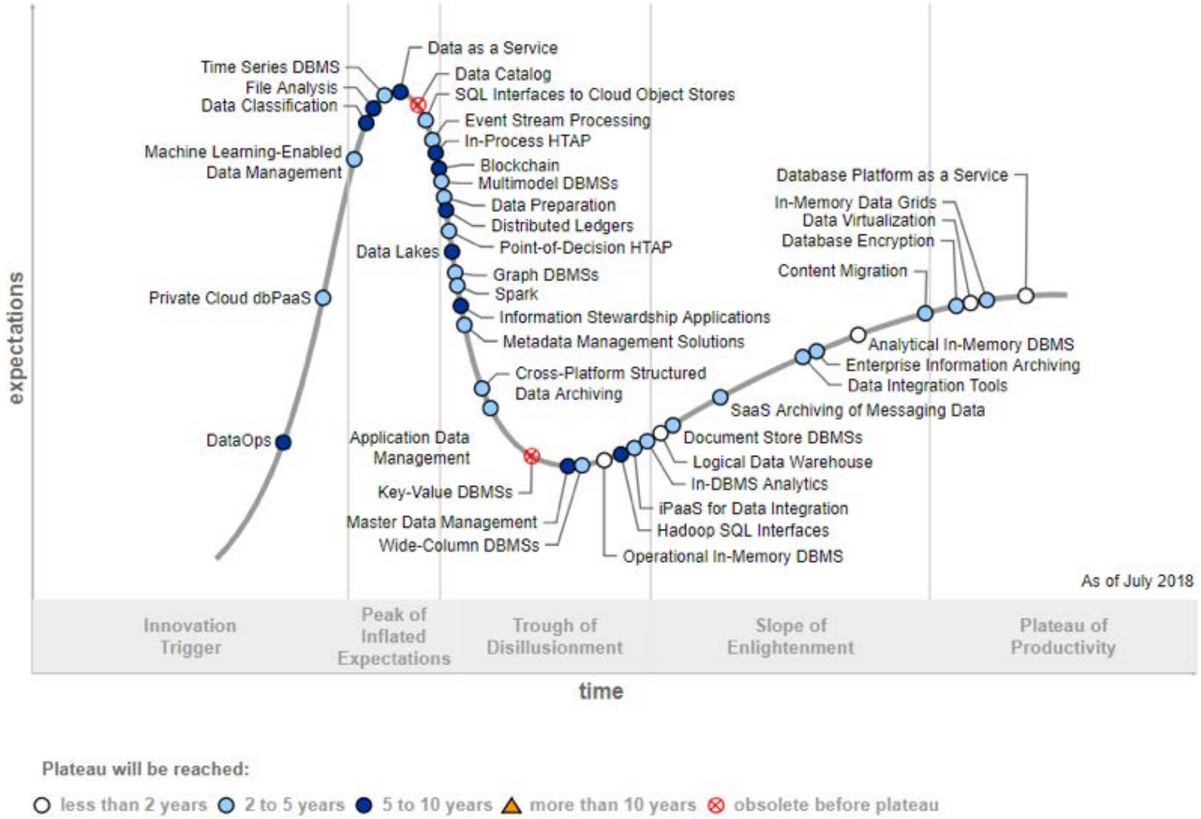
DataOps is the new black, providing a combination of DevOps and agile practices to automate the creation and management of data, analytics, and visualization on cloud platforms. DataOps is revolutionizing the way we deploy and manage data platforms.

WHERE DID DATAOPS COME FROM?

It is widely recognised that the term DataOps was coined by Lenny Liebmann in a blog post titled "3 reasons why DataOps is essential for big data success" posted on the IBM Big Data & Analytics Hub on 19th June 2014. An interesting fact is that a google search lists a blog posted on Lenny's personal website titled "DATAOPS: WHY BIG DATA INFRASTRUCTURE MATTERS" on 30th May 2014 as well as a number of other DataOps related content that predates Lenny's blog.

Andy Palmer is widely credited as popularising the term in his blog post "From DevOps to DataOps, By Andy Palmer" on the Tamr website on the 7th May 2015.

Like all things in the technology domain the use of the term was accelerated when an analyst firm, in this case Gartner, added DataOps to their hype cycle for data management in September 2018.



[1]

Gartner stated in the hype cycle article:

> *"DataOps is a new practice without any standards or frameworks," said Nick Heudecker, research vice president at Gartner. "Currently, a growing number of technology providers have started using the term when talking about their offerings and we are also seeing data and analytics teams asking about the concept. The hype is present and DataOps will quickly move up on the Hype Cycle."*

WHAT IS DATA OPS?

It is common for new terms that have started their journey through the hype cycle to be hijacked by industry analysts and vendors alike, resulting in a myriad of similar but disjointed definitions. We saw this happen recently with the term "Big Data," and are starting to see this behaviour happen with DataOps and AI.

So let's have a look at some of the definitions that have been created to date by individuals, industry analysts or vendors:

> *"DataOps, the set of best practices that improve coordination between data science and operations." –– Lenny Liebmann*[2]

[1] https://www.gartner.com/en/newsroom/press-releases/2018-09-11-gartner-hype-cycle-for-data-management-positions-three-technologies-in-the-innovation-trigger-phase-in-2018

[2] https://www.ibmbigdatahub.com/blog/3-reasons-why-dataops-essential-big-data-success.

> *"DataOps is a data management method that emphasizes communication, collaboration, integration, automation and measurement of cooperation between data engineers, data scientists and other data professionals." –– Andy Palmer*[3]

> *"DataOps is a collaborative data management practice focused on improving the communication, integration and automation of data flows between data managers and consumers across an organization." –– Gartner*[4]

> *"DataOps is an integrated approach for delivering data analytic solutions that uses automation, testing, orchestration, collaborative development, containerization, and continuous monitoring to continuously accelerate output and improve quality. The purpose of DataOps is to accelerate the creation of data and analytics pipelines, automate data workflows, and deliver high-quality data analytic solutions that meet business needs as fast as possible." –– Eckerson Group*[5]

> *"DataOps is a new way of managing data that promotes communication between, and integration of, formerly siloed data, teams, and systems. It takes advantage of process change, organizational realignment, and technology to facilitate relationships between everyone who handles data: developers, data engineers, data scientists, analysts, and business users. DataOps closely connects the people who collect and prepare the data, those who analyze the data, and those who put the findings from those analyses to good business use" ––Joydeep Sen Sarma; Ashish Thusoo*[6]

> *"DataOps: an automated, process-oriented methodology, used by analytic and data teams, to improve the quality and reduce the cycle time of data analytics." –– John Schmidt*[7]

No doubt additional variations of these definitions will arise in the future as DataOps traverses the hype cycle.

There does seem to be two themes used to describe what DataOps is, one focusing on the outcome of successfully using DataOps and one focussed on what is needed to be in place or leveraged to execute using DataOps.

In this paper I would like to outline a number of DataOps patterns I have seen my start-up company, AgileData.io and/or data analytics teams I coach, leverage to accelerate their DataOps behaviour. With this lens in mind I will suggest we adopt the following definition:

> *"DataOps combines agile development, DevOps and statistical process controls and applies them to data analytics." –– DataKitchen*[8]

Automate Everything

One of the core principles when adopting a DataOps way of working is to automate anything and everything that can be automated. The goal is that once the team has spent their limited time writing code, moving and changing data, developing analytical models, gone through the pain of ensuring it is all tested and documented and finally pushed it to production, then the things they have crafted should look after themselves from then on.

One of the DataOps mantra's we typically inherit from DevOps is "you build, you ship it, you run it". We want to automate everything as much as possible so the team doesn't

[3] https://www.tamr.com/blog/from-devops-to-dataops-by-andy-palmer/

[4] https://www.gartner.com/en/newsroom/press-releases/2018-09-11-gartner-hype-cycle-for-data-management-positions-three-technologies-in-the-innovation-trigger-phase-in-2018

[5] https://www.eckerson.com/register?content=dataops-industrializing-data-and-analytics

[6] https://learning.oreilly.com/library/view/creating-a-data-driven/9781492049227/

[7] https://www.amazon.com/DataOps-Authoritative-John-Schmidt-ebook/dp/B07X1SRSXH

[8] https://datakitchen.io/dataops-cookbook-second-edition.html

encounter the dreaded “BAU bleed”, where they spend the majority of their time fixing things in production rather than crafting new data assets.

Infrastructure as code

When thinking of automation most people think of infrastructure as code, another pattern we adopt from the DevOps stable.

Infrastructure as code is where we write code that provisions our infrastructure, rather than using a GUI or any other manual process. This pattern ensures the provisioning process is 100% repeatable, our infrastructure is standardised across environments and in theory, if required, we could tear it all down and build it all up again from scratch.

Technologies such as cloud providers, containers, git repositories and CI/CD pipelines have made adopting this pattern easier.

One thing to watch out for is the infrastructure you choose (and often the software) must be able to be provisioned via code, something a few vendors still struggle with.

Documentation as code

We can extend the automation pattern to areas outside of managing infrastructure, for example, creating and maintaining data pipeline and analytical model deployments.

One area that has had a number of surprising benefits as a result of automation is the area creation and publishing of documentation.

It is common for people to use Word Docs, Wiki’s or Atlassian Confluence as the place to create and maintain documentation. Documentation is manually typed into these tools, manually updated and source control is reliant on the versioning capability embedded within the publishing platform (if it even exists). When applying a DataOps lens to these processes, they can be categorised as anti-patterns.

There are a number of open source tools that enable us to treat documentation as code. We can create documentation in a markup language (for example, reStructuredText) in the same tool we are developing our code (for example, Visual Studio), we can commit it to our standard source control (for example, GitHub) and we can use Infrastructure as Code to generate a static html version of the website (for example, using Sphinx).

Within teams that I have seen, who have adopted the ability to be able to write documentation as code following their standard development process, I have observed documentation being written and updated with greater frequency and quality. There is something about creating and publishing documentation in small chunks that removes our aversion to doing it.

This documentation pattern also allows us the flexibility to “write once, publish many”. Often a team will be happy to have a static html documentation website as it meets their own needs, but the organisation has mandated tools such as Atlassian Confluence as the enterprise wide standard. I have worked with teams who have deployed their documentation to both a static site, via Sphinx, and published a portion of the same content automatically to Confluence via the Confluence API’s

Test Everything

Within the data and analytics domain we have a dirty little secret, developers will often push data to production without testing it is fit for purpose.

Good developers will create and run unit tests to ensure their code is valid, and good teams will create mechanisms to reconcile that the data acquired from the systems of record match the data that has been loaded into the data platform.

However, teams frequently fail to create tests to ensure the data transformations they have applied actually meet the business intent.

In the traditional project way of working the development team would often hand off to a testing team who would run through a raft of testing scripts they had created to ensure the data was fit for purpose. But this was a one off task bound within the project construct. If data mutated after the code had been deployed to production, it was up to a user to identify this had happened and raise a “bug”. And often the unachievable timeframes the project would be assigned would result in the coverage of this one off testing being constrained.

In the DataOps way of working, we need automated tests that constantly checks for code or data mutation after the code is running in production. The challenge is that there is a dearth of proven tools and techniques for this in the data and analytics domain.

One pattern we can adopt to test data comes from the Behavioural Driven Development (BDD) camp. Using languages such as Gherkin[9] and frameworks such as Behave[10] and Cucumber [11] we can craft and automatically execute tests which validate data transformations that are behaving the way we expect them to and, therefore, the data is more likely to be fit for purpose.

A typical example of a Gherkin script is:-

```
Scenario: Eric wants to withdraw money from his bank account at an ATM
    Given Eric has a valid Credit or Debit card
    And his account balance is $100
    When he inserts his card
    And withdraws $45
    Then the ATM should return $45
    And his account balance is $55
```

As you can see this is focussed on testing a feature within an application. But it is a simple exercise to use the language to define a business rule to describe the transformation of data:-

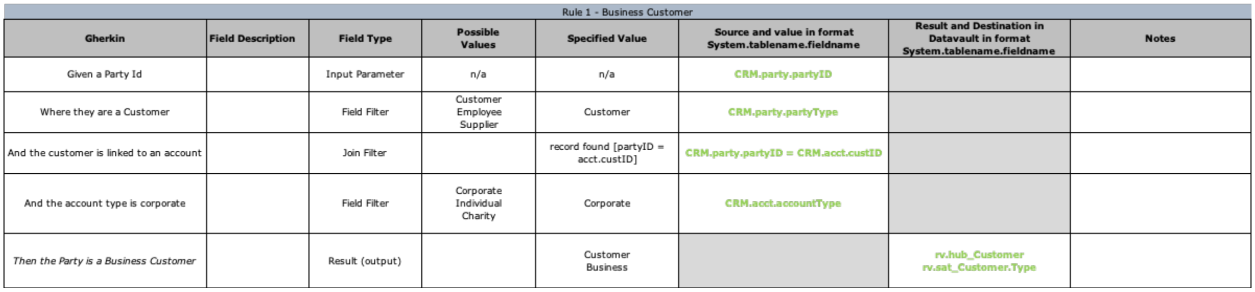
```
Scenario: Flag a party as a Business Customer
    Given a PartyID
    And the PartyID is a Customer
    And the Customer is linked to an Account
    And the Account Type is Corporate
    Then the Party is a Business Customer
```

We can then codify the rule into a technical mapping, which will help inform the transformation we need to create within a data pipeline:-

[9] https://en.wikipedia.org/wiki/Cucumber_(software)#Gherkin_language

[10] https://behave.readthedocs.io/en/latest/

[11] https://cucumber.io/



Rule 1 - Business Customer

Gherkin	Field Description	Field Type	Possible Values	Specified Value	Source and value in format System.tablename.fieldname	Result and Destination in Datavault in format System.tablename.fieldname	Notes
Given a Party Id		Input Parameter	n/a	n/a	CRM.party.partyID		
Where they are a Customer		Field Filter	Customer Employee Supplier	Customer	CRM.party.partyType		
And the customer is linked to an account		Join Filter		record found [partyID = acct.custID]	CRM.party.partyID = CRM.acct.custID		
And the account type is corporate		Field Filter	Corporate Individual Charity	Corporate	CRM.acct.accountType		
Then the Party is a Business Customer		Result (output)		Customer Business		rv.hub_Customer rv.sat_Customer.Type	

And finally we can craft unique Gherkin rules to test for any data mutation on a repeatable basis:-

```
Scenario: Check Account Type mutation
    Given a Party Account Type
    Where the Party Account Type is unique
    Then the count is equal to 3
```

This test could be deployed via a framework such as Behave and triggered every time a data pipeline is executed to check if a new Party Account Type has been created in the System of Record, and if any new values arrive we can issue a notification. The person receiving the notification can then validate if this new Account Type will require the transformation rule to be changed, or if it is simply a case of changing the Gherkin test rule assertion to equal 4.

Monitor Everything

Formula 1 motor racing provides the epitome of monitoring everything. Every aspect of a formula 1 car is monitored by hundreds of sensors. These sensors measure lap times, tyre and brake temperatures, air flow, engine performance and driver biometrics.

In an article by Intel they state:

> *"Before data analytics permeated the sport, success or failure in a race was solely down to the driver and the split-second decisions he made on the track."*[12]

This reflects the state of data and analytics when we don't adopt a DataOps way of working. We are reliant on the skills, experience and fortitude of the data engineer or the data scientist for the success or failure of our data and analytics.

As formula 1 drivers are now supported by a team of multi-disciplined experts, DataOps helps us form multi-disciplined teams that can monitor the entire data supply chain.

And like the innovation in technology that can be leveraged to monitor formula 1 cars as they hurtle around a race track at over 200mph, the data and analytics space has had a number of innovations that can also be leveraged to monitor everything.

Leverage the "Lake"

One of those is the advent of the "data lake". Data lakes have surfaced as a pattern out of the "big data" hype that has occurred over the last couple of years. So while not living up to the industry analysts and vendors overhyped promises of "the data warehouse is dead",

[12] https://www.intel.co.uk/content/www/uk/en/it-management/cloud-analytic-hub/big-data-powers-f1.html

they have introduced a number of patterns that are valuable when monitoring using a DataOps way of working.

Data lakes can be used to ingest and store data in its native format, following a schema on read pattern. This means that logs from all components of your data and analytics platform can be loaded into the logging lake with minimal effort. This provides a rich data source for monitoring improvements in the way you work, or identifying what is the next priority to work on.

For example, you can query this data to show an increase in the number of report queries being executed by users to indicate what you delivered added value to the users. You can identify long running user queries and then work to optimise them to provide a better level of service for these users.

Another useful pattern from the big data hype is real time streaming. Using streaming technologies such as Kafka and a Lambda architecture you can ensure you are alerted to failures that you care about in real time, rather than waiting until you identify them manually, or the users complain about the unavailability of their data or reports.

The DataOps way of working for storing and using your data and analytics log data is comparable to the DataOps way of working for testing, it should not be an afterthought. You should develop proven patterns that allow new technology components to sync their logs to your logging lake from the outset, and all engineers should bake this in as a standard task when engineering a new capability.

OKR’s

When looking to define metrics that can indicate if your DataOps journey is going well or not, you may have a tendency to boil the ocean and define a mass of metrics. The issue with this approach is you will probably get stuck in the detail, arguing which metric is the best one, or worse yet, spend all your time trying to figure out how to implement systems that actually measure these things (“what isn’t measured isn’t managed” right?), rather than focusing on the DataOps practices that would improve the results of these metrics.

To mitigate this risk we can leverage the OKR pattern from the agile world.

The original concept for Objectives and Key Results (OKR) came from Intel and has been leveraged by other Silicon Valley companies, for example, Google, Spotify, Twitter, LinkedIn, and Airbnb.

The structure of an OKR is:

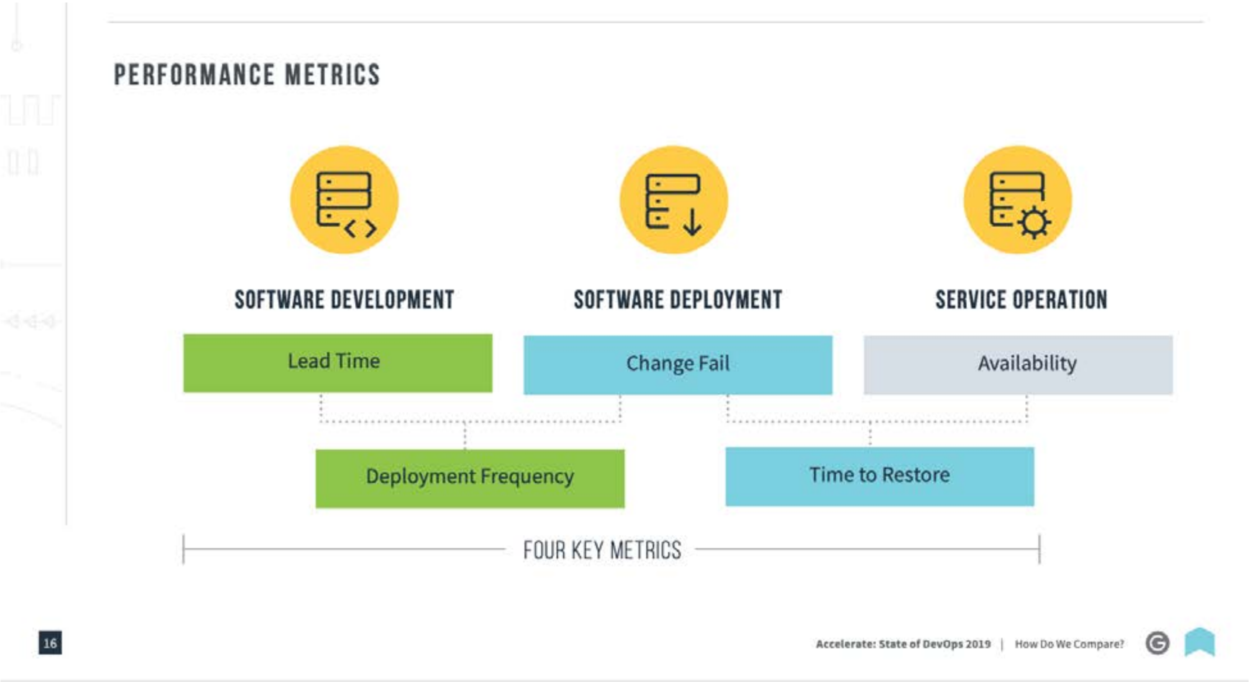
> *I will (Objective) as measured by (this set of Key Results).*

So an example might be:

> We will *improve the quality of the data our users access* as measured by *implementing behaviour driven development in 1 of our development squads* and *increasing our test coverage so 90% of all code has repeatable tests defined.*

For each objective, you should have no more than two to five key results. More than that returns us to the problems of having hundreds of metrics to manage. OKR’s should also be living things, as we achieve the key results for the objective we should add new key results to aim for. By doing this we are leveraging the agile practice of continuously inspecting, adapting and improving.

We can leverage the great work that is happening in the DevOps world when it comes to defining our initial objective and key results.   In the 2019 DevOps state of the nation report they have developed and validated five metrics that provide a high-level systems view of software delivery and performance and predict an organisation’s ability to achieve it’s goals.



PERFORMANCE METRICS

SOFTWARE DEVELOPMENT

SOFTWARE DEPLOYMENT

SERVICE OPERATION

Lead Time

Change Fail

Availability

Deployment Frequency

Time to Restore

FOUR KEY METRICS

16

Accelerate: State of DevOps 2019 | How Do We Compare?

They state in the report:

"The first four metrics that capture the effectiveness of the development and delivery process can be summarized in terms of throughput and stability. We measure the throughput of the software delivery process using lead time of code changes from check-in to release along with deployment frequency. Stability is measured using time to restore—the time it takes from detecting a user- impacting incident to having it remediated— and change failure rate, a measure of the quality of the release process."[13]

If you are struggling to get started defining your OKR's I suggest you use these metrics as examples of what you could focus on first. For example, you could focus on getting the deployment frequency of your new data pipelines down to weekly instead of monthly.

CONCLUSION

Moving to a DataOps way of working is a journey, and it is important to try not to boil the ocean at the beginning of this journey. So rather than attempting to implement all the new mindsets, techniques and patterns into a new DataOps way of working in one go, you should adopt an agile approach. Implement a small number of patterns, inspect the outcomes of using those patterns and then adapt your way of working to refine those patterns or experiment by adopting new patterns.

Every team, organisation and platform has unique nuances that means a cookie cutter approach to which pattern to apply first is flawed.

[13] https://cloud.google.com/devops/state-of-devops/

When looking at the patterns you could implement, I suggest you ask yourself these questions first and then determine if you need to implement new patterns to provide a better answer.

Who deploys your infrastructure?

- Is it your team or another team you are dependent on?
- Do they deploy via automated code or are there multiple manual steps involved?
- How long does it take to provision a new end-to-end platform or environment?
- Could you safely burn down your production environment and rebuild it from scratch?

How quickly can you roll back?

- If you deploy a change to your infrastructure, how quickly can you roll that change back?
- If you deploy a piece of code, how quickly could you roll that code back and remove any impact from its deployment?
- If you find a data issue due to “incorrect code” do you do a data fix, or do you roll back your code and redeploy?

Where are your bottle necks?

- Can your team explain each of the steps they typically take to deliver a working piece of code, data or analytical model?
- Do you measure the time taken for each of these steps to see if they are getting better or worse?
- Do you measure the rate of failure for each of these steps?
- If you had $100k to spend on improving one of these steps, how would you identify which step was the most valuable to invest in and if that investment was successful?

Who tests your “stuff”?

- Are tests created and executed by your team or is there a separate team that tests your stuff after the fact?
- When you provision your infrastructure, what tests do you run to confirm its operational, who runs them and are they automated?
- When you create a new piece of code or introduce a new piece of data, what tests do you have in place to ensure previously deployed code or data is not impacted?
- How do you know which tests have run successfully and which tests have not?
- Do you write your tests before you write your code?

REFERENCES

Liebmann, Lenny. “DATAOPS: WHY BIG DATA INFRASTRUCTURE MATTERS”. 30th May 2014. Available at http://lennyliebmann.com/?p=90

Liebmann, Lenny. “3 reasons why DataOps is essential for big data success”. 19th June 2014. Available at https://www.ibmbigdatahub.com/blog/3-reasons-why-dataops-essential-big-data-success.

Palmer, Andy. “From DevOps to DataOps, By Andy Palmer”. 7th May 2015. Available at https://www.tamr.com/blog/from-devops-to-dataops-by-andy-palmer/

Sen Sarma, Joydeep; Thusoo , Ashish. “Creating a Data-Driven Enterprise with DataOps”. 2017. Available at https://learning.oreilly.com/library/view/creating-a-data-driven/9781492049227/

Gartner. “Gartner Hype Cycle for Data Management Positions Three Technologies in the Innovation Trigger Phase in 2018” 11th September 2018. Available at https://www.gartner.com/en/newsroom/press-releases/2018-09-11-gartner-hype-cycle-for-data-management-positions-three-technologies-in-the-innovation-trigger-phase-in-2018

Eckerson Group. “DataOps: Industrializing Data and Analytics” June 2018. Available at https://www.eckerson.com/register?content=dataops-industrializing-data-and-analytics

DataKitchen. “The DataOps Cookbook” 2019. Available at https://datakitchen.io/dataops-cookbook-second-edition.html

Schmidt, John. “DataOps: The Authoritative Edition” 14th September 2019. Available at https://www.amazon.com/DataOps-Authoritative-John-Schmidt-ebook/dp/B07X1SRSXH

Intel. “The Secret to Formula 1’s Success? Big Talent Meets Big Data” 2019. Available at https://www.intel.co.uk/content/www/uk/en/it-management/cloud-analytic-hub/big-data-powers-f1.html

DORA. “2019 Accelerate State of DevOps Report” 2019. Available at https://cloud.google.com/devops/state-of-devops/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Shane Gibson
PitchBlack
dataops@pitchblack.nz
https://twitter.com/shagility