LEARNING MADE EASY

DataOps Special Edition

DataOps

for dummies

A Wiley Brand

Understand DataOps and #TrueDataOps

Balance governance with agility in data

Get started with DataOps

Brought to you by

Dataops

Justin Mullen
Guy Adams

About DataOps

Dataops.live started with a clear vision: to build data products and environments the same way the world builds software products. The company was born out at the meeting of two teams, one with a decade of enterprise data warehouse and data engineering projects and the other with a decade of software DevOps and CI/CD.

DataOps.live co-founders Justin Mullen and Guy Adams and their team developed the platform to address common data problems that afflicted most clients: slow development times, backlogged teams, and poor quality and version control. They then collaborated with other DataOps pioneers, including Kent Grazianio, Wayne Eckerson, Mike Ferguson in developing the #TrueDataOps philosophy, the seven-pillar model, and subsequently to write this book.

To support #TrueDataOps features, the DataOps.live team built its platform around Snowflake — the only data platform with key features such as Zero Copy Clone. As a result, DataOps.live established a strong partnership with Snowflake and became a certified product on the Snowflake Partner Connect marketplace, where customers can instantly provision trials.

DataOps

DataOps Special Edition

by Justin Mullen and Guy Adams

for dummies®
A Wiley Brand

DataOps For Dummies®, DataOps Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-119-73678-3 (pbk); ISBN 978-1-119-73683-7 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

We're proud of this book and of the people who worked on it. Some of the people who helped bring this book to market include the following:

Contributing Writer: Ted Coombs

Development Editor: Martin V. Minner

Project Manager: Martin V. Minner

Senior Managing Editor: Rev Mengle

Acquisitions Editor: Ashley Coffey

Business Development Representative: Matt Cox

Production Editor: Tamilmani Varadharaj

DataOps Contributors Team: Kent Graziano, Wayne Eckerson, and Mike Ferguson

Table of Contents

INTRODUCTION 1
About This Book 1
Icons Used in This Book 1
Beyond the Book 2

CHAPTER 1: **Recognizing the Challenges and Opportunities** 3
Recognizing the Data Product 4
Overcoming the Lack of Agility 5
Improving Business Users' Trust in Data Products 6
Proving You Are Doing It Right 7
Taking Down the New Privacy Silos 8

CHAPTER 2: **Understanding DataOps** 9
Defining DataOps 9
Introducing #TrueDataOps 11
The Seven Pillars of #TrueDataOps 12
Understanding the Source of Truth 15
Combining Governance and Agility 18

CHAPTER 3: **Benefitting from DataOps** 19
Shortening Time to Value 19
Creating Trusted Data Products 20
Making Business Users Happy 20
Reducing Deployment Failures 21
Improving Collaborative Pipeline Development 21
Scaling Development Teams and Boosting Efficiency 22
Going beyond Hope, Heroism, and Caution 23
Building Efficiency through Reusable Atomic Components 24
Simplifying the Orchestration of Massive Data Products 25
Enjoying Reduced Cost 25

CHAPTER 4: **Learning the Seven Pillars of #TrueDataOps** 27
Imagining a New Spirit of ELT 27
Working with Continuous Integration and Continuous Delivery 29

Writing Maintainable, Reusable, Testable, and Understandable Code 30
Environment Management 32
Conquering Governance with Change Control 34
Scaling with Automated Testing 34
Implementing Monitoring and Alerting as a Partner to Automated Testing 35
Collaboration and Self-Service Analytics (Data Catalogs) 35

CHAPTER 5: **Putting Together DataOps Resources** 37
Core Building Blocks of DataOps 37
Automated Regression Testing in a Test-Driven Development 41
Managing Continuous Pipeline Development and Version Control Using Git 42

CHAPTER 6: **Getting Started with DataOps** 47
Embracing the #TrueDataOps Philosophy 47
Selecting the Right Business Case 48
Engaging with the Right Stakeholders 48
Building a DataOps Team with the Right Help 49
Choosing the Right Technology Platform 51
Get on With It 53

CHAPTER 7: **Ten Reasons to Move to DataOps** 55
Using What You Already Have 55
Enabling Agility and Governance 56
Responding More Quickly and Shortening Time to Value 56
Demonstrating Assurance 57
Improving Security, Governance, and Change Control 58
Lowering Project Costs 58
Taking Advantage of Modular Components and Code 59
Increasing the Capacity of Your Team 59
Collaboration Across an Organization 60
Lowering Barriers to Success 60

Introduction

DataOps is a philosophy that has grown out of the successful DevOps culture but adds some effective twists to help you work efficiently with large amounts of data and achieve the best business insights. The truest form of DataOps is the #TrueDataOps philosophy. It is embodied in seven core principles that improve quality and reduce your company's time to see value from your data products. #TrueDataOps also explains the importance of treating your data as a *product* rather than a *project.*

In DataOps, Agile development, borrowed from the core of DevOps, is balanced with governance and data assurance through continuous development, experimentation, and automation.

Most importantly, DataOps describes a novel way of development teams working together collaboratively around data to achieve rapid results and improve customer satisfaction. It achieves significant increases in both agility and governance without compromise.

About This Book

This book is intended for everyone looking to adopt the DataOps philosophy to improve the governance and agility of their data products. The principles in this book should create a shared understanding of the goals and methods of DataOps and #TrueDataOps and create a starting point for collaboration.

This book will help you learn the basics of DataOps and #TrueDataOps, a practical philosophy that, when followed, will guide your business toward rapid success, greater data assurance, proper governance, reduced costs, and better stakeholder satisfaction.

Icons Used in This Book

As with most For Dummies books, you will find icons in the margins that help you spot important information.

REMEMBER

The Remember icon identifies things you might want to file away for later use and reminds you of important details.

TIP

The Tip icon points out helpful information.

Beyond the Book

The world of DataOps is continuously being refined and evolving. To keep up to date with the latest information or to increase your knowledge of topics covered in this book, consult these sources:

- `https://www.truedataops.org/`
- `https://www.dataops.live/`
- `https://www.dataopsmanifesto.org/`
- `https://www.youtube.com/channel/UCd3pZhhXBrBUkvUrqXbDhkw/videos` (the Dataops.live YouTube channel)
- `https://en.wikipedia.org/wiki/DataOps`

IN THIS CHAPTER

- » **Recognizing the data product**
- » **Replacing the waterfall model with Agile**
- » **Building assurance and user trust**
- » **Creating an audit trail**
- » **Overcoming privacy silos**

Chapter 1 Recognizing the Challenges and Opportunities

It is no secret that data and the IT requirements to manage it are increasing exponentially. As the technology world increases in complexity, the need to tame its development with a measured balance of governance and agility grows with it. The constraint of this balance — that if you want more agility you must give up some governance, or vice versa (as shown in Figure 1-1) — has long been accepted.

One of the key promises of DataOps is that it provides big increases in both agility *and* governance, neither having to suffer as a result of the other.

An early challenge in IT came with the burgeoning growth of program code that led to the principles that govern software development today: DevOps, a set of guiding principles that allow for agility while maintaining governance over the development and deployment of code.

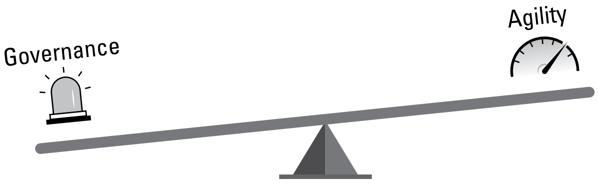


FIGURE 1-1: The perceived balance of agility and governance.

DevOps has led to principles and tools that provide the ability to maintain configuration and code in repositories, check-in/check-out functionality, and most importantly, the ability to roll back code in the event of a failure. Encapsulated, reusable code, and automated testing are further features of DevOps that revolutionized code creation. These are just a few of the reasons DevOps has become the standard for all software development (from Amazon and eBay to any startup you can name) and is likely to retain that role far into the future.

These principles, tools, and features are now being applied to the next logical level, the *data* upon which that code most often relies or creates. The challenges of working with data within today's growing data complexities and massive volumes, as well as the opportunities they present, have brought about the next obvious step, DataOps. That is not to say that DataOps is nothing more than DevOps and Agile for data. Differences clearly exist between DevOps and DataOps but the overarching principles are the same. Chapter 2 explores DataOps and the principles of #Truedataops in detail.

Recognizing the Data Product

Within enterprises, it has been typical to talk about *data projects,* works with a defined scope, start, and end time. This approach does not reflect the reality of data in modern organizations. A typical software product like an eCommerce portal has an overall vision, perhaps a pilot, but then moves into an ongoing steady state of continuous requirements collection, backlog grooming, prioritization, and continuous delivery to the product's customers. For data to achieve the same agility and governance, it must follow the same process. Organizations must consider their portfolio of data products (rather than an assorted mix of discrete data projects).

REMEMBER

Some of the critical differences between a *data project* and a *data product* include:

- **Duration:** A *project* has a defined start and end. A *product* is ongoing.
- **Management:** The project team looks after a *project* for the duration of the project. A product team looks after a *product* throughout its life.
- **Release:** A *project* has a small number of releases at its end. A *product* has regular releases throughout its lifetime.
- **Scope:** A *project's* scope is fixed. A *product* includes an evolving, prioritized backlog.
- **Agility:** A *project* has low agility with high barriers to change. Once the project scope has been defined and signed off, minimal scope for change exists. A *product* has high agility and a vast capacity for change. A workload backlog is continuously reprioritized.
- **Testing:** A *project* has testing built into the project plan; no testing takes place outside of project timelines. A *product* has automated regression testing built into the release process.

REMEMBER

Thinking about your data as a product will put you in the right way of thinking to be successful for the future.

This new product-based way of thinking certainly presents some challenges. The following sections covers how some of these challenges are met by applying the principles of DataOps.

Overcoming the Lack of Agility

Agility in software development is an approach that emphasizes the delivery of developed code, which is developed, tested, and ideally deployed in increments, as opposed to the older waterfall method where code was released upon project completion. Code created using the waterfall method was typically released into a beta version where people would start initial testing to find errors. The Agile code development methodology, primarily created by early web developers, revolutionized software development. Agile development embodies principles such as:

- Incremental code delivery
- Automated testing
- Team collaboration
- Continual deployment
- Active stakeholder involvement
- Continued learning

Developing data pipeline code and logic for data analytics involves the same challenges and can likewise benefit from principles like incremental releases and automated testing.

Data analytics must meet an exponentially growing demand to provide data-derived insights with increased speed and efficiency, and data engineering drives this. This demand can no longer be met using traditional development methodologies. An opportunity exists to vastly improve the speed and reliability of data analytics. The Agile method of development has proven to be a superior process that yields faster and more dependable code and pipeline releases, driving increased speed and reliability of data analytics.

REMEMBER

What businesses are demanding from data at an increasing rate is *insight*. Delivering that insight at an increased speed, level of efficiency, and reliability is a critical goal.

Business users' requirements and expectations are evolving quickly, and data engineering must meet those ever-increasing expectations. Delivery times in weeks or months are things of the past as stakeholders have become accustomed to same-day delivery by software development teams.

Improving Business Users' Trust in Data Products

Businesses rely on insights from data, though a lack of trust can be created when business users find errors in their data that render those insights meaningless. The trouble with mistakes is that they reduce trust in the data and the data team, increase stress levels, and increase the risk of embarrassment when business users notice the errors. And mistakes reduce the time the business invests in innovative development.

Data is being updated continuously, but your business users don't want to be the test team for each data product — they need to trust the data you provide, the instant it is available.

TIP

Your data product will always be tested. The only question is whether you'll test it before release, or whether you'll use your business users as your testers. The second approach is becoming much less acceptable.

Trust in data also extends into governance oversight. This challenge can only be overcome with the assurance brought about by proper governance and validation of both the data *and* the code used to analyze it. The challenge is that governance historically increases overhead, reduces speed, and decreases your data team's agility to respond rapidly to business demand. Governance affects agility and responsiveness.

Proving You Are Doing It Right

A data engineer's job is complicated and has primarily been a manual one to date, involving data integration development, data pipeline assembly, data testing, managing environments and production deployments, and creating documentation. Keeping up with stakeholder demands and same-day expectations becomes impossible because of these slow, manual, error-prone processes.

Beyond merely keeping up with demand, you have the increased burden of proving that you're "doing it right." To accomplish this, you need change control and an audit trail that provides accountability, assurance, and dependability.

In addition, a key requirement for assurance and dependability is testing. Testing at every stage before the code is moved into production provides positive assurance.

TIP

With the increased burden of complexity that user expectations, governance, and testing bring with them, the only way forward is to do more with less. One way to accomplish that is to free data teams from tasks that have significant value but are also time-consuming, repetitive, and automatable, like testing.

As with many IT challenges, the answer is automation — of *everything.* For example, testing is a laborious and mundane task that should be automated. Automated regression testing and automated monitoring are the answer to catch and eliminate any future problems before the user sees them. These systems check the quality of your production data, analytics processes, and new code releases.

You can also automate things like traceability and issue tracking with continuous code releases. Automating these day-to-day tasks of testing and monitoring increases the efficiency and agility of the entire data analytics process.

TIP

In addition to freeing your team from mind-numbing tasks, automation has a direct impact on the Total Cost of Ownership (TCO), which includes the ongoing cost of maintaining your code.

Taking Down the New Privacy Silos

Data that ends up in a data platform often has certain information security restrictions, making it either difficult or impossible for business users in the same organization to integrate data between data sets. Business units protecting access to their data will often create new privacy silos. Overcoming these privacy silos requires the use of techniques like anonymization and masking. This is a way of efficiently allowing business units to sanitize and anonymize their data to enable others to access it.

IN THIS CHAPTER

» **Defining DataOps**

» **Exploring #TrueDataOps and the Seven Pillars**

» **Building from DevOps**

» **Understanding the Source of Truth**

» **Balancing agility and governance**

Chapter 2 Understanding DataOps

Following a path of continual improvement, balancing agility versus governance, DevOps-based concepts are being applied to data engineering and analytics. This approach incorporates the concepts of Lean and Agile to the way data is stored, transformed, and viewed.

Defining DataOps

REMEMBER

DataOps means "Building, testing, and deploying data platforms, the exact same way we do software platforms."

DataOps, short for data operations, was born from applying Agile, Lean, and DevOps philosophies and principles to the creation of data products. DataOps (like DevOps) is a philosophy, not a technology (though as with DevOps, key technologies underpin it). DataOps is a way of approaching and delivering enterprise data operations incorporating both agility and governance. DevOps, and by inference, DataOps, relies on continual and Agile development that incorporates creativity and balances governance with agility, resulting in enhanced collaboration between developers and stakeholders.

Continuous, incremental development, testing, and code delivery lead to an accelerated lifecycle that increases user satisfaction while reducing overall development costs. In contrast to the plan-first, develop-later methodology (the waterfall method), changes to applications and data platforms are made during development, based on the continuous and collaborative feedback of the stakeholders, as well as automated testing and monitoring, as shown in Figure 2-1.

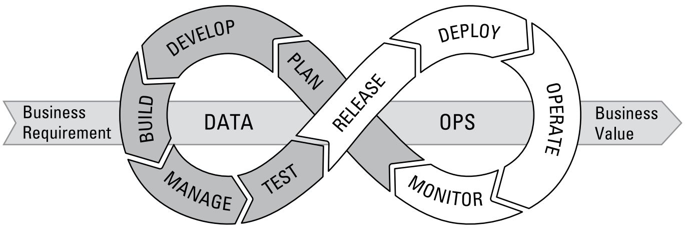


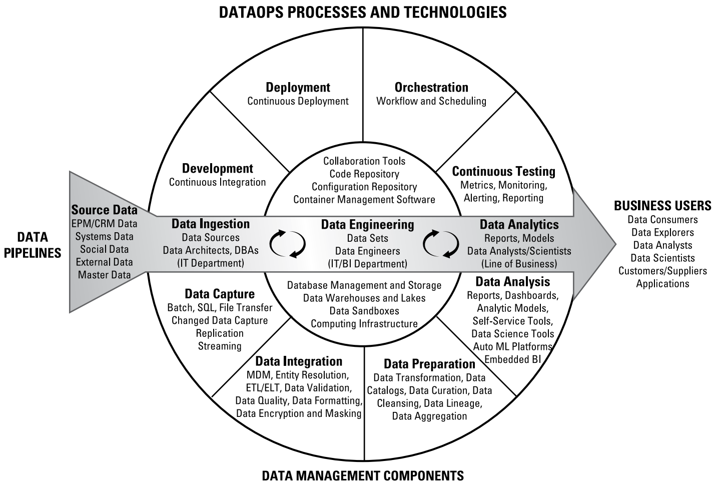
FIGURE 2-1: The cycle of continuous development is at the heart of DataOps.

A common misconception is that DataOps is DevOps applied to data engineering, data analytics, and data science. The name *DataOps* communicates that the DevOps philosophy applied to data analytics and the creation of data products can produce an exponential improvement in quality, governance and change control, and cycle time. However, it is so much more.

REMEMBER

The ability to balance agility and governance is integral to the DataOps philosophy and method. The ability to respond promptly to user demands while remaining true to the principles of governance like change control, accountability, data quality, and data security is key to any DataOps project. Thus, governed agility; Agile governance.

Figure 2-2 reflects the needs of data pipelines and where DataOps fits in these pipelines as well as in a typical organization.



Courtesy Eckerson Group (www.eckerson.com)

FIGURE 2-2: Where DataOps fits in most data pipelines.

Introducing #TrueDataOps

The DataOps Manifesto is a valuable resource to define what we are aiming to achieve with DataOps and what it should deliver. That's the "What," but it's missing the "How." #TrueDataOps is a philosophy that defines core principles or pillars of how data is managed and delivered with both agility and governance, forming the "How" of DataOps.

TIP

Learn more about the DataOps Manifesto at `https://www.dataopsmanifesto.org`. For more information about the #True DataOps philosophy, visit `https://www.truedataops.org`.

Using the truest principles of DevOps, Agile, Lean, test-driven development, and Total Quality Management, #TrueDataOps applies these principles to the distinctive disciplines of data, data platform management, and data analytics. Additionally, regular, daily, and hourly collaboration among members of a data team becomes essential for evolving stakeholder requirements and the development of diverse and self-organizing teams. When teams collaborate in this way, the need for individual or single-discipline heroism is reduced and scalability is increased. Collaboration is essential for continual learning, improvement, and growth.

Because data analytics involves using a variety of tools that either directly or ultimately create code, organizations need end-to-end orchestration of each component in the process leading to the end product: data insights. An important element in this environment is a versioning system, similar to that used in DevOps, to manage access to and governance of the data, the components and code used to build and populate the data products. Additionally, creating temporary (feature branch) versions of the code and data repositories in disposable environments eliminates development on stale or non-representative data and allows stakeholders to review work as it is being undertaken.

#TrueDataOps focuses on value-led development of pipelines — the goal of any pipeline must be to deliver value to the end-users. Examples include reducing fraud, improving customer experience, increasing uptake, identifying opportunities, and so on.

TIP

#TrueDataOps focuses on simplicity and quality in the development of data pipelines. These pipelines must be developed with the inclusion of automated testing and monitoring to ensure continued data quality and consistency. To create simplified and dependable data pipelines, reusing code reduces complexity, adds to dependability, and improves cycle times (see "Code design and maintainability" in the next section).

The Seven Pillars of #TrueDataOps

These seven pillars define the key requirements for a successful DataOps implementation (see Chapter 4 for more details):

- **The spirit of extract, load, transform (ELT)**

  Extract, load, transform (ELT), not traditional extract, transform, load (ETL), provides for the loading of data into the data platform in its raw format, reducing load times, maximizing the ability to derive new value from the data in the future, and pushing its transformation down the data pipeline. However, the spirit of ELT is more fundamental — it's about doing everything possible to the data to ensure that value and potential future value, in whatever form, is not discarded.

- **Continuous integration/continuous delivery (CI/CD)**

  One of the hallmarks of DevOps is the continuous integration (CI) and continuous delivery (CD) cycle from a code repository. A DataOps repository allows configuration and code to be stored, tested, promoted, released, and rolled back and each time there is a change to deploy this change to a target environment.

- **Code design and maintainability**

  Code maintainability is critical to the DevOps and DataOps lifecycles. DataOps has adopted the use of atomic code or small, reusable code and components, with a specific function, from DevOps. The #TrueDataOps philosophy prioritizes small, reusable code units to increase efficiency.

- **Environment management**

  One of the most critical and important features of DataOps and #TrueDataOps (and one of the biggest differences between DevOps and DataOps) is environment management. The ability to create and efficiently maintain both long-lived and short-lived environments from long-lived and short-lived branches underpins much of the DataOps requirements.

- **Governance and change control**

  #TrueDataOps must have strong governance and privacy models included in the core design, not added later. With the ever-increasing criticality of data for business decision-making, ensuring that this data is well-governed is essential. Governance and change control in DataOps environments are similar to DevOps environments and include who made every change, when, why, who reviewed the changes, and the results of automated testing.

- **Automated testing and monitoring**

  Agile development is about continuously releasing code incrementally. To keep up with the rapid pace of releases and the increasing size of code bases and data sets, test coverage using fully automated testing is key. Testing the production quality of the data and code modules for errors at each release is also critical to know if your changes worked and that they did not negatively affect the changes that occurred before the one you just made.

- **Collaboration and self-service**

  Collaboration and self-service are the results and benefits of the successful #TrueDataOps process. Collaboration means many people from different departments or disciplines can work together effectively. Self-service allows many people, traditionally excluded from data engineering processes, to contribute, increasing the bandwidth for change and increasing engagement.

- **Building from DevOps**

  Software development has evolved over time and matured with concepts borrowed from object-oriented programming to create modular programs with reusable code. The DevOps philosophy arose from the need to streamline the software development and operations lifecycles, delivering custom software quickly and efficiently.

DevOps challenged conventional IT thinking with its innovative approach. DataOps is doing the same today. Both have foundations in an Agile and continuous delivery methodology, though they remain different because the end products are dissimilar. Programmers in a DevOps environment focus on an evolving code base and developing functionality. Data, on the other hand, is dynamic. Data engineers are focused not only on application functionality but also on business goals and metrics. These must be continuously validated so that stakeholders trust the data and the insights provided by data analysis.

Fundamentally, the key distinction of the #TrueDataOps philosophies compared to others is that #TrueDataOps starts with DevOps, keeping everything positive that has been refined and battle-hardened for many years by thousands of organizations. Key differences exist between DevOps and DataOps, but rather than throwing away the positive elements of DevOps, we work out these differences and address them specifically.

REMEMBER

DataOps must retain everything that's good in DevOps, while adding changes and enhancements to make it work well for data.

Understanding the Source of Truth

The most important area in which DevOps and DataOps are the same is in the concept of the *Source of Truth.*

One of the biggest challenges organizations have with data products is the creation and maintenance of different environments. Organizations planning legacy data projects often schedule "three weeks to set up a QA environment," "two weeks to copy changes from dev to QA," and so on. The challenge is that each environment is its own self-contained Source of Truth.

Databases were never designed for this and practically all manual or DIY-scripted solutions for maintaining dev/test/prod environments are problematic and slow at best. The challenge is compounded because they become the Source of Truth for the "configuration and code" logic that manipulates the data *and* the data itself, as shown in Figure 2-3.

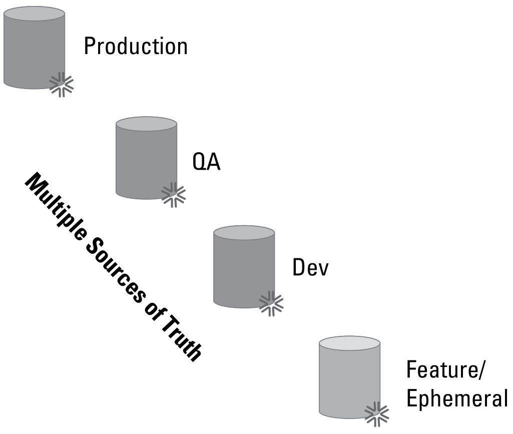


FIGURE 2-3: Source of Truth: managing multiple environments.

The software world (especially the web development world) solved this problem many years ago. They moved their Source of Truth out of the execution engines (web servers, for example) into a central Source of Truth. In virtually all cases this is now a Git repository, as shown in Figure 2-4.

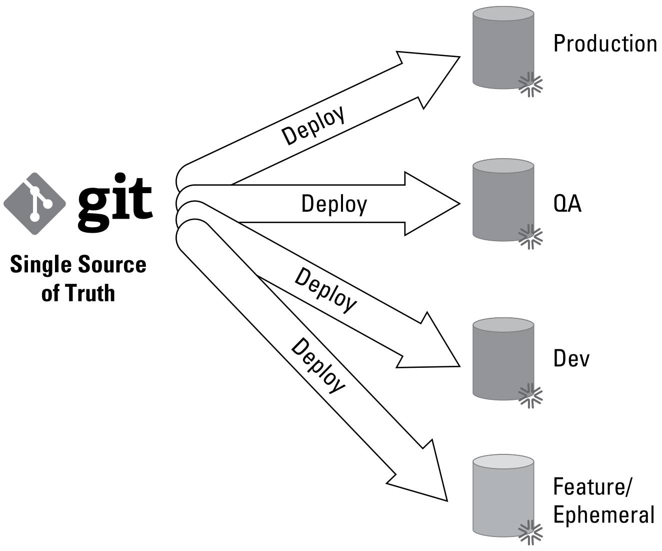


FIGURE 2-4: Source of Truth: software development.

The goal then can be stated as moving the Source of Truth for this data configuration and code out of the target systems and into a central repository, as shown in Figure 2-5.

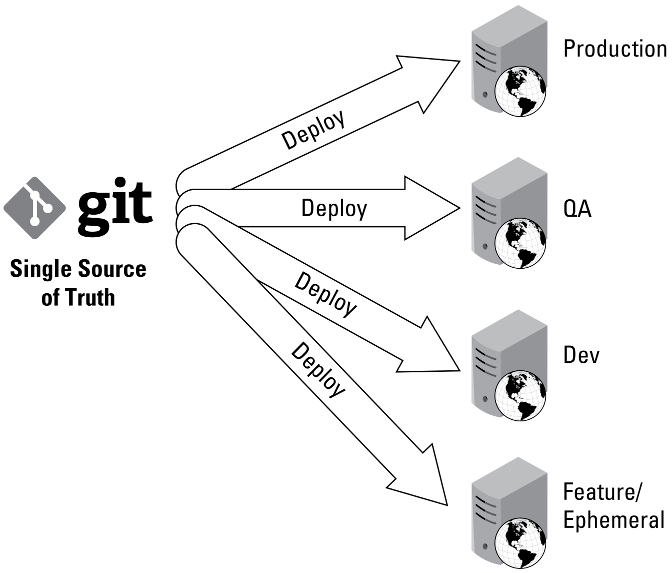


FIGURE 2-5: Source of Truth: data development targeting modern cloud data platforms like Snowflake.

When this single Source of Truth for the data product's configuration and code is combined with the actual source data (the other Source of Truth), you have all the elements required for DataOps to be successful, as shown in Figure 2-6.

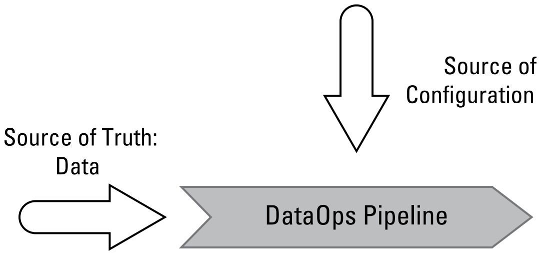


FIGURE 2-6: Source of Truth for data, configuration, and code powering a DataOps pipeline.

REMEMBER

Different versions of the Source of Truth exist for code and configuration logic, but these are manifested as *different branches inside the Git repository.*

The overall problem remains the same: "How do I move all or some of the changes from one version of the Source of Truth to another version?" However, because the code and configuration logic have been moved inside the Git repository, the problem has been transitioned from a system totally unsuited for the purpose (like a database) to a system built and used by millions of developers for exactly this purpose.

One definition of DataOps (or at least a DataOps pipeline) might be "everything needed to take code and configuration from different branches in the Source-of-Truth Git repository and use this to make the target environment correct, valid, and up to date."

What "correct, valid, and up to date" means, *and* deciding how to achieve that, is the job of the pipeline.

REMEMBER

DataOps projects the Source of Truth onto the target environment.

REMEMBER

Finally, don't forget that your DataOps repository only reflects the Source of Truth for your configuration and code. The Source of Truth for your data is still the source data from your operational or transactional systems.

Combining Governance and Agility

Traditionally, agility and governance for data products presented a classic "pick one or the other" quandary. Achieving either alone is straightforward, but at the direct expense of the other:

- Being very agile by deploying changes straight to production with zero change control, review, or testing, is simple, but governance is almost nonexistent.
- Achieving great governance with highly formal processes and only updating production twice a year is similarly straightforward, but it kills agility.

REMEMBER

One of the realities of a DataOps environment is that agility is balanced with governance, so you achieve both. Agile brings a less cumbersome, or less preplanned, approach to development and maximizes speed of delivery for stakeholders. Governance is embedded in the pipelines and supplies the rules, processes, and procedures that maintain the security, quality, and deliverability of data. Not only are agility and governance decoupled, but the core DataOps philosophy adds considerable value to both at the same time.

IN THIS CHAPTER

» **Increasing efficiency with trusted data assets**

» **Working smarter, not harder, with code reuse**

» **Reducing errors**

» **Building collaborative teams**

» **Saving money**

Chapter 3 Benefitting from DataOps

You'll begin experiencing the benefits of DataOps immediately. Happier customers, happier data teams, reduced costs, better business insights, and reduced time-to-value are just a few. However, the truest value of DataOps is not what is achieved in a few days or weeks, but the long-term value as the data product grows bigger and more complex over time. The return on investment (ROI) period for a DataOps implementation is short, and the value increases exponentially over time.

Shortening Time to Value

The pace of software development has drastically increased. The same is now expected of the data team. Business users ask for answers tomorrow instead of next month or in six months. Customers now think of these short timelines as reasonable expectations.

Continuous development in an Agile development cycle means that you begin developing immediately once a goal is set out and agreed on by project stakeholders. Refinement comes later.

Develop some code, create a few tests to catch any obvious problems, and give it a try. Involve team members and end-users in the results. The team provides feedback, and a new cycle of refinement can then begin. Most of all, your end-users will see results even if they aren't perfect the first time. This method gives the end-users confidence and a stake in the outcome.

Creating Trusted Data Products

The world of DevOps teaches that reusing tested and trusted code, automating testing, and providing good peer review and governance not only save a considerable amount of development time and cost, but deliver much higher quality, trusted products. The same is now true for data products. A trusted data product is a reusable business asset used in multiple places and multiple times. This method saves time because you no longer need to create new data from scratch every time you use it. For example, you can have trusted data sets of customer or production data.

REMEMBER

Trusted data products include preconstructed data sets or views that can span multiple data sources and may involve data transformation to anonymize or obfuscate sensitive data.

Making Business Users Happy

Business users are happier in a DataOps world for several reasons. First, they are not excluded from the process. Their participation and collaboration, as shown in Figure 3-1, ensure that they have a say in the end result of a data product.

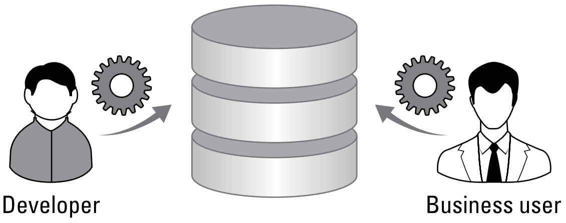


FIGURE 3-1: The developer and business user collaborating around the same database.

A second reason is the abbreviated time it takes for users to begin seeing results. Because the process is iterative, the outcomes will be apparent almost immediately. This process allows them to participate in the evaluation and improvement of the product, resulting in a data product that best meets their business needs.

Reducing Deployment Failures

Automated testing and peer review reduce the likelihood of deployment failures. One of the great things about testing data over testing source code is that errors in data processing usually fail in a spectacular way. If you expect a query to return a million lines and it returns ten, you know something is wrong. In many ways, writing good tests for data is easier that writing good tests for software — as long as you have the right testing framework.

The iterative nature of Agile development lets you "fail fast." Get in, write what you think will work, test it, find the failure, and fix it. Then move on to the next iteration, test it again, and fix it again if necessary. This "try it, fix it, try it again" method of developing gives you faster end results.

REMEMBER

Before pushing a change to production, you should always have at least two pairs of eyes on the change. This practice helps to ensure that changes are solid. Do not rely on a single person who may be having a bad day.

Improving Collaborative Pipeline Development

Collaboration is one of the hallmarks of a successful DataOps philosophy. Data engineers, data scientists, and data analysts collaborate with stakeholders from many different disciplines. These stakeholders share their knowledge and provide feedback along the way. Working together results in the most effective and successful data projects.

DataOps makes use of modern collaborative tools to make communicating at this level more efficient.

Scaling Development Teams and Boosting Efficiency

In order to scale up development teams and boost efficiency, it is necessary to set up a complete development sandbox fully ready to develop in minutes and destroy when done in seconds. With frequent stakeholder feedback, deployment teams will spend very little time building a product that the end-user doesn't need. Efficiency is improved by writing tests once, to benefit hundreds of times, always being confident that not only is your feature working, but that it hasn't broken any other part of the system.

TIP

Borrowing from the Lean model of software development, using Git repositories allows work to go on in multiple branches simultaneously without affecting the production code. Remember, data analytics is in essence just code written in languages like SQL and other lower-level languages such as Python, R, Java, and others.

A key benefit of good DataOps development is *sustainable development.* As with any other new digital product, making, testing, and deploying changes should be quick — mainly because the number of other things you have to avoid breaking is small. However, with legacy development, as the system grows larger, the time taken for any given development takes longer because everything takes longer to set up and test, and the likelihood of breaking something increases (the bottom curve in Figure 3-2). This can lead to huge costs to the business in the future and is ultimately unsustainable.

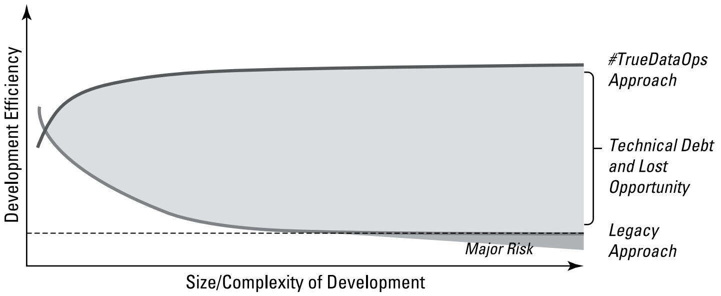


FIGURE 3-2: The challenge of sustainable development.

With DataOps development (and precisely the right sort of automated testing), development efficiency does not drop over time; it flattens off (the top curve in Figure 3-2). A given piece of work is as quick to develop, test, and deploy when the product is five years old as it is when it's only six months old. This approach is totally sustainable.

The cost of the legacy approach is a large accumulation of technical debt and lost opportunity because extra time spent doing some things slows progress on other business priorities. Eventually, the business will have enough and will limit this loss in development efficiency by capping the amount of manual testing time allowed (the dashed line in Figure 3-2). However, the effect is to introduce significant risk — the system is becoming more complex, and with a fixed manual test resource, less of it will be tested over time.

REMEMBER

The total cost of a data product must be measured over years. Many organizations don't focus on the full lifetime cost and development efficiency over time and end up with an unviable result. DataOps prevents this.

Going beyond Hope, Heroism, and Caution

Attempting to be a hero doesn't work. This behavior is untenable. Rather, work together with a multi-disciplinary team to achieve the project's goals together. Don't spend endless hours working on your own through your stakeholder requirements to meet rapidly shrinking deadlines. This behavior is unscalable and limiting. Being overworked leads people to release untested code and "just hope it works." Unfortunately, that is typically when end-users find your errors.

You can go overboard in trying to reduce errors and heroism by adding caution to the mix. You plan each phase, carefully testing each outcome. Your product becomes over-engineered and no longer fits the Agile development philosophy where you produce a deliverable as quickly as possible and refine it in the next cycle. Stakeholders rapidly become disillusioned when delivery timeframes don't meet their expectations.

TIP

Don’t be a hero. Heroism is unsustainable. Instead, share the burden and the glory with your team.

Building Efficiency through Reusable Atomic Components

“Build once and reuse often” is a paradigm long understood in software development. Creating small pieces of code that perform one single function and can be developed, tested, and reused time and again shortens future development times and leads to consistency and dependability. Paying someone to develop code is expensive, so maximizing the use of existing code makes great sense.

TIP

If you can grab a block of code that has already been battle tested in production, you save time coding and testing it yourself. You achieve the greater efficiency, simplicity, and extensibility inherent in the Agile development process.

The Agile process

Like DataOps, the Agile development process has a manifesto that embodies a set of principles, the priority of which is “to satisfy the customer through early and continuous delivery of valuable software.” The Agile process also incorporates collaboration in building products. Success is measured in how well a product works. In DataOps, success is measured in how well a data product produces reliable insights from data. Agility also comes from simplicity, or “the art of maximizing the amount of work not done.” That is, don’t do anything you don’t have to do.

Refactoring

Refactoring in code development is a way of refining code without changing its function. Refactoring can bring about increases in speed and reductions in complexity. The goal is to make your code more maintainable and extensible.

TIP

Don’t get too involved in refining code early in your development cycle. Get deliverables into the hands of stakeholders and then refine.

TIP

You can use a data modeling technique, like Data Vault 2.0 or something else, that lends itself to iterative development and minimizes the need to refactor.

Simplifying the Orchestration of Massive Data Products

DataOps simplifies the movement of data and code logic between various tools along a data pipeline. Data sets can be massive and span multiple data sources. The movements of data through the pipeline are not necessarily serial. Multiple processes may happen in parallel and then repeat for further testing and refinement.

Good DataOps also understands the need within data for incremental data capture, incremental updates, and incremental data transformations. Not every part of a data product can or should be rebuilt each time a pipeline runs.

Automating the orchestration of the data through the pipeline improves collaboration and simplicity. DataOps platforms act like "control rooms" to visualize and manage the movement through different tools. Essentially data must be acquired or extracted, loaded, transformed, and analyzed as described in the section on ELT in Chapter 2.

Enjoying Reduced Cost

The cherry on the DataOps cake is that once your business users are happier, greater business insights are being achieved from your data, and both your code and data are better maintained and better governed. It all works like a well-oiled machine, saving your business a great deal of money.

Software products within a single orchestrated framework work together cohesively. This may even enable a reduction in the number of tools you need to maintain or to achieve the same results. And it lowers learning curves and leads to dependable, repeatable results in a far shorter time frame.

TIP

One of the great results of DataOps is that it leads to greater self-service data. This allows stakeholders and data analysts to construct their own data analytics pipelines (without giving up review and governance); thus, reducing the number of data staff necessary to maintain and support growing and increasingly complex data needs.

IN THIS CHAPTER

» Storing raw data, then transforming it

» Developing smart and Agile

» Maintaining governance

» Building with collaboration that leads to self-service

Chapter 4 Learning the Seven Pillars of #TrueDataOps

The #TrueDataOps philosophy consists of the seven important pillars introduced in Chapter 2. This chapter goes further in helping you understand these concepts, critical to the success of DataOps.

Imagining a New Spirit of ELT

When you're working with data, your first step is to extract it from somewhere. The data might originate as customer data, sales data, website queries, Internet of Things (IoT) device output, or an infinite number of other sources. What happens to the data next has changed from historical data processing methods. Rather than transforming the data and altering it to meet specific needs, as in the extract, transfer, load (ETL) model, storing the data in its raw form makes much more sense. Storage is cheap. And having raw data that can be transformed time and again and can be used for future requirements without having to redo ingestion is priceless.

Loading the raw data has the advantage of getting to value quicker because no transformation takes place, and no engineering must take place to transform it. The data is merely stored. The process is fast and straightforward. Once data has been altered, it's been degraded in some form. By starting with raw data, you can build an audit trail and lineage of what has happened to the data over time, creating better governance and building a more trusted data set.

REMEMBER

Simplicity is one of the key goals of #TrueDataOps.

In some cases, you can't avoid regulations that require that some data be removed, encrypted, or anonymized for privacy. In these cases, a small "t" is inserted into the ETL acronym (EtLT), signifying minimal, but required, change.

Moving beyond ELT but staying true to the broad idea of storing raw data that can be accessed later, DataOps proposes that no data that may be useful later should ever be removed. This is the "new spirit of ELT." The "lift-and-shift" pipeline model meets the age-old challenge of data lost because of changes. For example, in human resources data, you might store an employee's choice of insurance plan. When that choice changes, the data about the initial plan may be lost. What do you do? Make sure the data is only overwritten after a secondary table records the change (slowly changing dimension), ensuring that all previous versions of the data remain accessible. Although not part of ELT, this approach ensures that data that may be valuable in the future is not lost. Thus, this method is consistent with the "new spirit of ELT," the first of the seven pillars of #TrueDataOps, as shown in Figure 4-1.

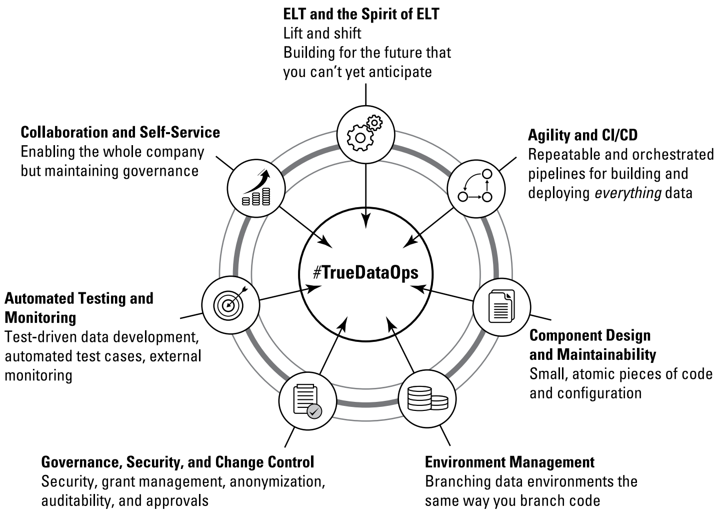

ELT and the Spirit of ELT
Lift and shift
Building for the future that you can't yet anticipate
Agility and CI/CD
Repeatable and orchestrated pipelines for building and deploying *everything* data
Component Design and Maintainability
Small, atomic pieces of code and configuration
Environment Management
Branching data environments the same way you branch code
Governance, Security, and Change Control
Security, grant management, anonymization, auditability, and approvals
Automated Testing and Monitoring
Test-driven data development, automated test cases, external monitoring
Collaboration and Self-Service
Enabling the whole company but maintaining governance
#TrueDataOps

FIGURE 4-1: The seven pillars of #TrueDataOps.

REMEMBER

When implementing ELT, keep the complexity of configuration and implementation extremely low. Do this by simply extracting and loading data in raw format with zero or near zero changes. Only then should you start to transform and change the data.

Working with Continuous Integration and Continuous Delivery

In an Agile DevOps development process, code development is followed by continuous integration (CI), which includes the build and test steps. This is followed by a continuous delivery (CD) step and then finally a run step. In DataOps, the CI stage primarily incorporates orchestration of the data testing followed by continuous delivery (CD) with orchestration and monitoring steps. Integrations should occur in such a fashion that no window exists between commit and build, at least in development branches. Errors are then caught and immediately handled.

TIP

To achieve continuous integration, you need to implement a source code revision control system that allows branching. Stored within the source code control system are each of the components necessary to build the data product.

#TrueDataOps incorporates every aspect of life cycling the data product: model generation, orchestration, deployment, diagnostics, governance, and metrics. This mirrors many of the original concepts from DevOps. However, some concepts in DataOps don't have parallels from the DevOps world. A key one is lifecycling database objects. These create a unique challenge because they are often *stateful* (they cannot be overwritten or recreated each time). The traditional approach to this issue has been an imperative one (essentially creating a list of specific actions to be taken in order — see `www.dataops.live/blog`), but a declarative approach is far more flexible and more powerful. Modern data platforms such as Snowflake, with advanced features like time travel and zero copy clone, prevent the effective use of imperative approaches.

When starting a new feature, you begin by creating a branch in your code repository. In the DataOps world, creating a new branch of your "code" must include creating a branch of the data (branching the data platform). This complete environment then becomes your sandbox where you begin developing a prototype. The iteration of continual improvements, with collaboration and feedback, occurs until you are ready to merge your code into a production branch.

Lean software development is a process derived from car manufacturing. At its simplest, Lean development means that you build a minimum viable product or develop just enough to see if the product has value before continuing. Also, create just enough to see how things should progress. For example, when developing a new presentation model or definition, it's far better to build something quickly and solicit feedback from stakeholders. Based on this feedback, you decide the next steps to take in development or even decide not to create the product. This is a far better approach than building something and getting feedback after the fact.

Lean development can save a great deal of time in wasted development. This method can lead more directly to a finished product that meets everyone's needs and best complies with security and governance. However, Lean development, with all its strengths, doesn't work without collaboration. Small and rapid development cycles, collaborating, and testing to catch potential errors in an Agile manner lead to the best results.

Sometimes a stated goal of DataOps is to streamline the data lifecycle management or even to make it invisible. In this lifecycle, you collect the data, store it somewhere secure, make use of the data, and get rid of it when you no longer need it. This includes moving data between hot and cold storage based on usage.

Some DataOps tools allow you to streamline data lifecycle management.

Writing Maintainable, Reusable, Testable, and Understandable Code

Building cars is an excellent example of an industry where complexity is a challenge for assembly, testing, maintainability, and repair. The solution was modularization, a type of plug-and-play. When car manufacturers began using replaceable components, cars became simpler to design, build, and fix. In addition, an aftermarket for the components developed.

Reusable code modules offer the same attributes as modularized vehicle manufacture.

The software world has refined and perfected this concept. In a typical software product, the number of lines of code custom written for that application versus the total number of lines included (by the time every third-party module, library, and so on has been counted) is usually relatively small. Advanced concepts are achieved quickly and easily because developers are building on well-written modular code. They also then develop their own modules to be reused within their organizations.

Another key advantage of this modularization is testing and understandability. Because a good module is well defined and has a single, clear scope, the module can have its own tests and is easy to understand.

Writing reusable modules for DataOps has the same advantages in terms of reduced cost and complexity. A mature DataOps organization has its own repository of standardized components that are reused across its different data products.

There are also best practice elements to writing good code. It's always been true that writing well-documented code that conforms to a style and format is easier to understand and change later. Creating a style, even in how SQL is written, allows simpler debugging or changing over time. Many engineers have strong opinions over style, and these can become a blocker for good progress.

REMEMBER

A reasonable, agreed-upon style well followed by everyone is far more effective that different people following their own "better" styles. Standardization is more important than perfection.

Modularized code enables the reuse of trusted components used and tested in multiple products. Using trusted parts already creates greater stability in your projects. This practice doesn't remove the need for testing, but it reduces the number of errors you will have to fix later. Reusable models can also serve as a base that more complex modules are built upon. Inheritance allows you to take the core, trusted parts of a component and insert them into new and possibly more complex components. This practice saves a great deal of time, developer resources, and money.

The DataOps philosophy stands with modularized, or atomic, code components. Because the idea is to simplify, reduce errors, and engender trust, as well as to save time and money, adopting this form for your development makes sense.

For larger organizations, it's sensible to create a central repository of these reusable components for many different business units, departments, or data products to use. This method reduces the work for individual projects (because they build on work already done) and increases standardization (because everyone uses the same logic).

REMEMBER

As with motor vehicles, an aftermarket for reusable components exists. This supports the development of a self-service environment.

Environment Management

Version control systems in software development have revolutionized the way programs are developed. Code branches are created, allowing developers to work on different features within a codebase while other branches are in testing, and still, the master branch is in production.

Multiple versions of environments exist whether you are discussing DevOps or DataOps. Some of these are long-lived environments, such as production, quality assurance, and development (prod, QA, dev). One goal of DataOps is to keep these three long-lived environments in sync, a difficult task because they tend to diverge quickly once created. The task is complicated further in the data world because of the *data* (test data, QA data, and production data). For more details, see "Understanding the Source of Truth" in Chapter 2.

Dynamic environments can also exist for specific purposes. Feature branches allow developers to create environments that enable them to do their own development and integration testing. In data engineering, creating feature branches has been historically tricky because of the size and complexity of most data sets.

DataOps manages these challenges by building, changing, and destroying these feature branch environments automatically (see Figure 4-2).

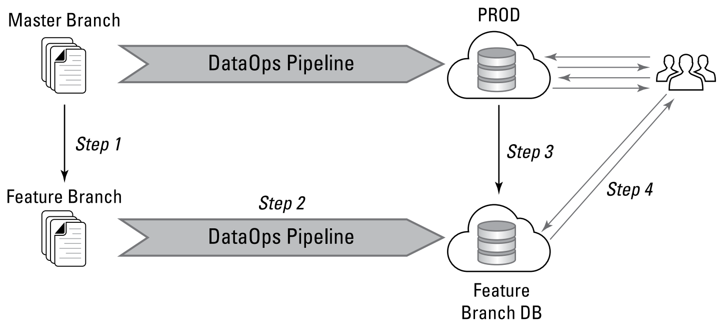


FIGURE 4-2: The challenge of sustainable development.

The process is as follows:

1. The DataOps user creates a standard Git branch of the configuration and code from a long-lived branch like `master` or dev .
2. The DataOps user runs a standard pipeline.

   REMEMBER

   Pipelines are inherently environment-aware. This isn't a special pipeline — it's the same pipeline definition that is running in `master`, but it knows it's running in a different context and changes its behavior accordingly.
3. Rather than creating a new database, the pipeline uses the Zero Copy Clone feature. This allows you to create copies of massive, multiple-terabyte datasets rapidly and on demand.
4. Because the end product is "just another database," stake-holders can interact with it just as they would with the production database. This supports many of the benefits discussed in Chapter 3, especially regarding collaboration and self-service analytics.

REMEMBER

Agile isn't just about developers being able to work faster or more independently, it's also about developers iterating with stake-holders more frequently and easily. The ability for a stakeholder to see, in a fully working database, the effects of changes made by a developer just a few minutes ago is revolutionary.

Conquering Governance with Change Control

DataOps requires that strong governance be part of the initial plan rather than something dropped on a project later. This is *governance by design* and *privacy by design*. When changes are made to a data product, every change is automatically tested for either code or data errors.

Additionally, the DataOps philosophy requires that at least two people review changes before they are pulled from a repository or merged into the codebase. Repositories should contain every aspect of your data product. They should monitor when anything is accessed or changed, who changed it, why it was changed, and who approved it, thus creating an audit trail that will remain with the data forever.

REMEMBER

In DataOps, having a single Source of Truth is an absolute requirement and is the root of much of the governance capabilities. As in DevOps, your principal Source of Truth is the code and component repository.

Scaling with Automated Testing

Automated testing is one of the features brought from DevOps, where automated testing and deployment increased the ability to scale. Data requirements are already pushing data teams to their limits. The ability to test code and data is one of the factors limiting growth. To overcome the problem of data products being released while merely "hoping" they will work, automated testing can be put in place to catch most of the errors. Rapid testing and deployment enable large, data-heavy companies to code data pipelines and deploy updates at an incredible rate. It also avoids having to use your business users as your testers, which erodes confidence in the data.

In the real world, data drives critical business decisions and those decisions must be informed by data that the business can trust. Automated testing of data leads to greater assurance and quality in the data.

REMEMBER

Automated testing isn't just about governance. Trusted data is critical to great business decisions.

Implementing Monitoring and Alerting as a Partner to Automated Testing

Testing is great at catching errors prior to releasing a data product to production. However, automated testing is only as good as the tests you have created and can never cover all eventualities. Automated testing reduces the number of issues that get into production dramatically but can never eliminate them entirely.

TIP

Once a data pipeline is in production, monitoring and alerting are put in place to catch errors that may have been overlooked as well as new errors caused by changes. You don't want the end-user to become your monitoring and alerting system!

When the automated monitoring system detects errors, the data analytics team is automatically alerted. Because of the Agile nature of development, change is expected, and corrections are simply added to the next release. Error corrections are thus included more quickly. In the past, developers fixed code and waited for a major release, in which many errors were fixed at the same time. With Agile development, errors caught by monitoring and alerting can be handled immediately.

Collaboration and Self-Service Analytics (Data Catalogs)

DataOps has a core principle that collaboration is the key to success. One factor that has always limited data teams is the ability for large team sizes to effectively collaborate. Collaboration among members of a team, and across disciplines, leads to creative solutions and better results. It is paramount to include end-users and stakeholders in your discussions because they have the best idea of what they need from a data product and will be the ones to determine if your data product is a success. Including end-users also minimizes the amount of time spent heading in the wrong direction. When end-users and stakeholders feel they've been heard, they remain happier along the path to the end result.

Self-service data analytics, underpinned by rich, accurate, and up-to-date data catalogs are the utopia of data platforms today. To enable self-service data analytics, business users, data scientists, and data analysts must be able to access an up-to-date data catalog, easily find the data they want, get access to or request access to that data, and upon receiving that data, be able to make use of it within the privileges granted. #TrueDataOps is the orchestrator that enables this utopia.

REMEMBER

The development of the #TrueDataOps philosophy and the seven pillars of #TrueDataOps are revolutionary. Like all new paradigms, #TrueDataOps requires company-wide participation in its adoption. Collaboration brings the company together in the creation of its data products. And collaborative practices lead to better understanding and greater trust in one of the company's most important assets: its data.

IN THIS CHAPTER

- **Executing pipelines**
- **Testing code**
- **Managing code branches**
- **Auto-documenting code**
- **Creating self-service using data catalogs**

Chapter 5 Putting Together DataOps Resources

Working with DataOps, you often hear the word *orchestration.* Orchestration is the essence of DataOps. Like a symphony, data is orchestrated from its inception through to its storage and delivery. Many tools are used at different stages of the orchestration, like the instruments in an orchestra, to bring about results.

One of the keys to using different tools within a DataOps orchestration is that these tools must have a way of intercommunicating about the data using metadata. It is therefore critical that you use metadata-driven development tools.

Core Building Blocks of DataOps

The practical application of the DataOps philosophy consists of implementing several core building blocks. Without the use of all of the components listed in this section, the adoption and implementation of the DataOps philosophy cannot and will not be successful. Therefore, this section looks at these components.

Pipelines

DataOps fosters building value-led data pipelines. These pipelines are value-led because their adoption leads to a reduction in fraud, improvements in customer experience, increased uptake, and the identification of opportunities.

Two types of pipelines exist:

- **Development and deployment pipelines:** In DataOps, these pipelines use continuous integration and continuous delivery (CI/CD) to build, test, and publish a data platform's containers, application programming interfaces (APIs), device drivers, and libraries. They contain all the capabilities of a typical CI/CD pipeline. DataOps just adds functionality to the CI/CD pipeline.
- **Data pipelines:** In data pipelines, data moves through from ingestion, transformation, testing processing, and numerous other steps. The pipeline is an orchestration component — the pipeline is orchestrating the jobs that move data, but it isn't moving the data itself. A company likely has several data pipelines. These can run anywhere between every 5–10 minutes to hourly, daily, or even weekly, depending on the type and depth of testing that must take place. For instance, very large data sets may involve a great deal of testing that you wouldn't want to do hourly or even daily.

Pipelines control the logical flow of jobs: which ones need to happen in which order, which pass data to each other, which much success for others to be attempted, and so on.

A good pipeline should have a number of characteristics:

- **Repeatable:** A pipeline should be able to be run many times.
- **Idempotent:** Repeated runs for a pipeline with no changes to input states should not change the target system.
- **Branch- and environment-aware:** There should not be different pipeline definitions for different branches. A single pipeline definition should be inherently branch-aware.
- **Selectable:** It should be possible to select which pipeline (chosen from multiple pipelines, perhaps based on frequency) should run in any circumstance.

A pipeline is usually presented visually as a set of jobs including their sequencing and dependencies, as shown in Figure 5-1.

FIGURE 5-1: Pipeline visualization.

Jobs

Jobs are the elements in a pipeline created to orchestrate the data pipeline's workflow. Jobs consist broadly of:

- Which technology they apply to (Snowflake, Talend, Stitch, Python, HTTP API, and so on).
- Configurations needed to drive that step.
- Details of which location this step should be executed in. For example, it may need to be executed securely inside a data center or private cloud if it needs access to specific data or systems.
- Criteria for this job to succeed.
- A definition of how this job should change its behavior when running in different environments or branches. For example, a job that ingests large amounts of source data in production may not do the same in development.

A job is usually visualized as a node within a pipeline, as shown in Figure 5-2.

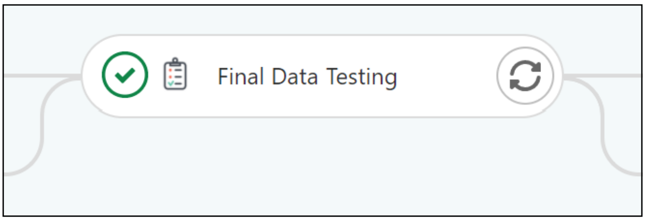


FIGURE 5-2: Job visualization.

Version control

In the DevOps paradigm, *version control* (or source control) is the practice of tracking and maintaining changes to software source code. It helps solve the challenges of multiple software developers working on the same software development project by tracking individual changes, keeping a history of these changes, and preventing concurrent work from conflicting. In the rare event of a failure, it also allows changes to be rolled back and the target systems to be reverted to their previous state.

The DataOps philosophy adopts these principles and adapts them for data, keeping all the inherent governance and change control implicit, but adding key capabilities that make it work with large-scale data environments.

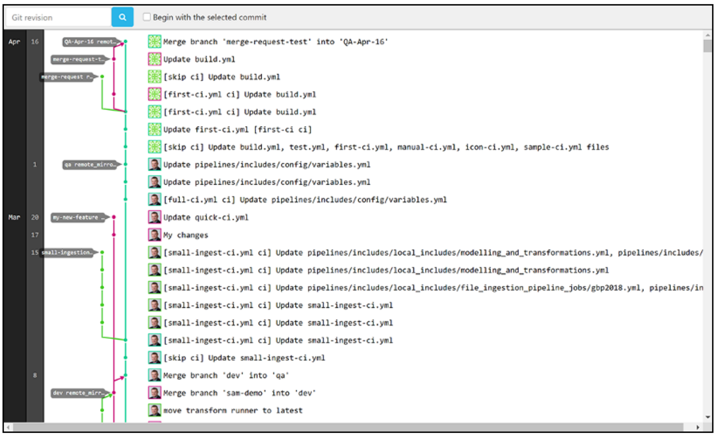
TIP

Git is the version control system most commonly used by DevOps and DataOps implementations. It is recommended by major role players in this space.

Branch

It is impossible for multiple members of a data team to work simultaneously on the same files without negatively affecting each other's changes. Modern source control tools like Git have the functionality that makes it easy to create individual feature branches of the code repository. One of the key requirements of DataOps is the ability to tie together branches and environments. For example, the master branch is the one that drives the production environment, the QA branch is the one that drives the QA environment, and so on.

Managing branches becomes important in large projects. Having a way to see where a particular branch originated (where it was "cut from") and where and when it was merged back is very important. Figure 5-3 shows an example.

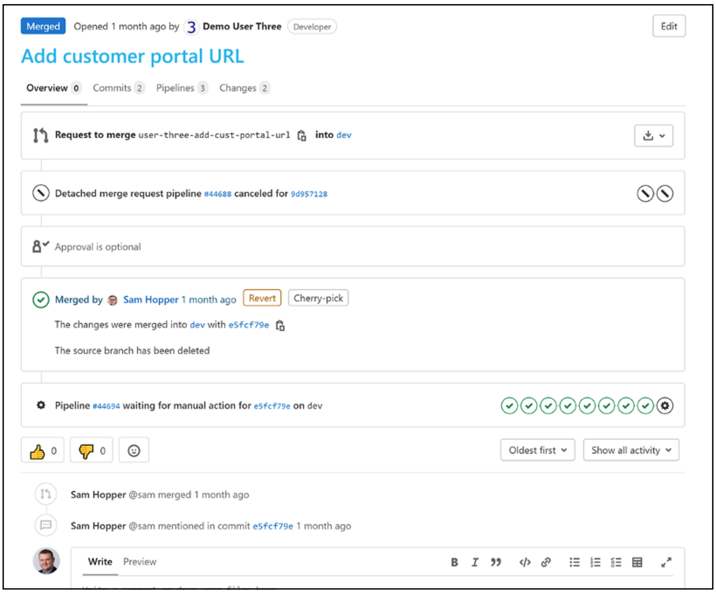


```
Merge branch 'merge-request-test' into 'QA-Apr-16'
Update build.yml
[skip ci] Update build.yml
[first-ci.yml ci] Update build.yml
[first-ci.yml ci] Update build.yml
Update first-ci.yml [first-ci ci]
[skip ci] Update build.yml, test.yml, first-ci.yml, manual-ci.yml, icon-ci.yml, sample-ci.yml files
Update pipelines/includes/config/variables.yml
Update pipelines/includes/config/variables.yml
[full-ci.yml ci] Update pipelines/includes/config/variables.yml
Update quick-ci.yml
My changes
[small-ingest-ci.yml ci] Update pipelines/includes/local_includes/modelling_and_transformations.yml, pipelines/includes/
[small-ingest-ci.yml ci] Update pipelines/includes/local_includes/modelling_and_transformations.yml
[small-ingest-ci.yml ci] Update pipelines/includes/local_includes/file_ingestion_pipeline_jobs/gbp2018.yml, pipelines/in
[small-ingest-ci.yml ci] Update small-ingest-ci.yml
[small-ingest-ci.yml ci] Update small-ingest-ci.yml
[small-ingest-ci.yml ci] Update small-ingest-ci.yml
[skip ci] Update small-ingest-ci.yml
Merge branch 'dev' into 'qa'
Merge branch 'sam-demo' into 'dev'
move transform runner to latest
```

FIGURE 5-3: Branching graphs.

Merge request

Once a feature branch has been checked, it can be changed, tested, and then sent back to a higher-level branch, such as the master branch, using a merge request. The merge request also serves as a code review tool. This is where the "other pair of eyes" to review your changes comes into place. Once your colleague has approved your changes, the code branch is merged back to the master branch. A DataOps tool should support a robust process for merge requests including approvals, comment, and discussions. You should be able to see the results of pipelines for that merge request and all the changes it contains. Figure 5-4 shows an example.


Merged Opened 1 month ago by 3 Demo User Three Developer

Add customer portal URL

Request to merge user-three-add-cust-portal-url into dev

Detached merge request pipeline #44688 canceled for 9d957128

Approval is optional

Merged by Sam Hopper 1 month ago

The changes were merged into dev with e5fcf79e

The source branch has been deleted

Pipeline #44694 waiting for manual action for e5fcf79e on dev

Sam Hopper @sam merged 1 month ago

Sam Hopper @sam mentioned in commit e5fcf79e 1 month ago

FIGURE 5-4: Initiating a merge request.

Automated Regression Testing in a Test-Driven Development

Testing is critical in any product. In the software world, the concept of releasing a new version of a product without a lot of testing (in reality, virtually all automated testing) is unthinkable. And yet, this is what happens with data in most companies all over the world. Even in a simple case where just a few hours' worth of data has been loaded, you still have a new version of the data product, and you shouldn't think about releasing this unless you've performed a good set of tests against it. Users typically prefer data that is slightly older than data that is up to the minute but known or suspected to be bad.

In test-driven development (TDD), you write testing procedures before you write a single line of code. Amazingly, knowing the criteria you'll be testing for allows you to plan for failure conditions. Thus, you can reduce failure and the number of iterations you may have to go through in development.

TIP

Implementing test-driven development and planning tests ahead of time allows more collaboration. When considering constraints, you can talk with others about what those constraints should be. Although iterative development may catch these refinements later, planning for them beforehand saves time and results in a better end product. TDD is also a rigorous way of defining the expected behavior of the system. Overall, it results in a better-defined requirement with many more of the edge cases thought through.

Managing Continuous Pipeline Development and Version Control Using Git

Many version control systems (VCS) have existed over the years, though Git recently has become overwhelmingly dominant. A good Git system is the key to massively concurrent development and collaboration (the largest Git repositories have thousands of commits per week by hundreds of different people). A large ecosystem of tools and development environments is also available to support Git.

Intelligently managing branches with Git

A linear version control system is simple to imagine because each version follows its predecessor. That's also the problem with this system — each version must be based on the previous version.

Linear versioning was common in the old waterfall development method where versions were identified by a cryptic string of numbers, such as Version 1.6.7.2.

The process is challenging if several colleagues are working on different versions of the code and want to merge their changes. In the Git feature branch method, developers can create individual branches that are safe and isolated, only merging back to the master branch when the changes have been approved.

REMEMBER

Using a Git repository over a linear repository gives you far more flexibility, particularly when your development team might be distributed or when the developers may not always have access to the most recent codebase.

Requiring push-button pipeline execution

One of the essential features any DataOps platform must provide is push-button data pipeline execution. With a simple click of a button, the DataOps platform must be capable of executing several pipelines against the data simultaneously. Push-button means that it's the job of the pipeline to decide what work needs to be done based on the branch it's running in and the state of the target system. Pipeline executions should be repeatable and idempotent (meaning they can be run repeatedly producing the same result).

Managing the environment with branching

The benefits of implementing the DataOps philosophy is that it allows for diverse data teams to work simultaneously on a data product or data pipeline implementation.

This is achievable by using a version control system that allows full branching. Working within the Git repository, all users can collaborate on different branches in much the same way that software developers work. Two main types of branches exist:

- **Long-lived branches:** These branches generally always exist and are connected to data platform environments that are similarly long-lived.

» **Feature branches:** These should be short-lived branches, living just long enough to develop and test a feature or enhancement before being merged into a long-lived or integration branch. The data platform environments they dynamically create should be similarly short-lived and deleted as soon as the development has finished. Creating a feature branch is simple using DataOps for Snowflake. Zero Copy Clone is a feature of Snowflake that allows a branch of the production environment to be instantly created in the feature branch, where analytics can be coded and tested.

A third type of branch is helpful for advanced use cases. If a large feature is being worked on — call it "Feature X" — it is common to create an integration branch, such as `feature_x`, which is created from dev. Many feature branches are then cut from there, developed in, and merged back (through merge requests) into `feature_x`. This acts as the integration areas for all the subfeatures of Feature X. When all the features of Feature X are completed and tested together in `feature_x`, a merge request is then created to merge this whole feature atomically into dev.

Automate your testing — monitoring for what falls through the cracks

In a traditional software development model, the more money invested in the data platform, the higher the cost of future work, including maintenance and new developments, resulting in sprawling code and increasing complexity. These factors add up to the fact that testing will be more resource-intensive and convoluted.

Automated regression testing solves these challenges, reducing cost and the burden on the data team. The equation is as follows:

In order to deploy new updates confidently and quickly, you need good test coverage all of the time. As the data product increases in size, the amount of testing required increases exponentially.

The only way to solve this equation without automation is to apply an exponential amount of resources over time — clearly unacceptable from a cost perspective. The alternative is to add automated testing where the test sets are ever-increasing as the data product increases. This solution creates an additional compute workload over time where the cost to test the data product over time increases, but only at a tiny fraction of the cost of having people do it. The elastic compute of data platforms like Snowflake means that this can still be performed quickly.

Automated testing is about preventing errors and mistakes from creeping into production. However, it cannot and will not ever be perfect. Therefore, adding automated monitoring into the mix is advantageous. The principle of automated monitoring means accepting that, although automated testing is integral, it can never be perfect, and eventually an error will slip through. This is where monitoring plays its part. It can pick up the small mistakes that might slip through the automated regression testing process. The DataOps philosophy does not see this as a negative but as a controllable reality.

Auto-documenting for governance and configuration control

No one likes to create documentation, but it's necessary. In the software world, code documentation and architecture diagrams used to be created before development started. This was important for legacy waterfall development because the time and cost to fix a mistake or envisioned scenario were high. A long and invariably losing battle then ensued to try to keep the documentation up to date with the real code as it changed over time.

Once Agile development became commonplace, this approach was abandoned in favor of documentation and diagramming that were produced automatically from the code itself as part of the build process. The need for large upfront designs was replaced with a development process that was fast to adapt to changes and therefore did not penalize upfront mistakes or omissions.

The same holds true for DataOps. Traditional data architectures and documentation are far less important, especially after the first version of a data product, replaced instead with rapid iterations and automatically produced documentation and diagrams. Using automation to do the mundane work for you only makes sense.

Documentation is usually formatted using the Markdown language (amusingly enough, a *markup* language) to format plain text documentation stored in the repository with the code.

TIP

In a CI/CD environment, you don't want to get bogged down in, or skip, the documentation to meet your Agile commitments. Documentation is an integral part of governance. Therefore, automating it is a core requirement. Finding tools that will work within your orchestration is key.

Building data catalogs from facts, not harvesting and inference

One of the goals of an exemplary DataOps implementation is that it generates user-facing data catalogs. When you go to a restaurant, it's nice when the menu describes the offerings in some detail so you can make a good choice. The same is true of data catalogs. Describing what the data contains allows users to select data in a self-service fashion. Gartner says that data catalogs "have graduated from an up-and-coming technology to a must-have."

The data cataloging ecosystem is rapidly developing powerful new concepts like machine learning (ML)-augmented catalogs that can use ML to discover your data, curate it, update a profile, perform tagging, and create semantic relationships between distributed and siloed data to allow simple keyword searches through the metadata. ML can even recommend transformations and auto-provision data and data pipeline specifications.

However, even with ML augmentation, any data cataloging is only as good as the data (and metadata) it has access to. A DataOps pipeline has access to a huge array of metadata, including where data came from, how it has been transformed and processed, how it has been tested, and those test results, to name just a few. This data is valuable for the enrichment of a data catalog to augment what it has learned from harvesting specific systems.

IN THIS CHAPTER

» **Getting started on the right path with DataOps**

» **Selecting the right business case**

» **Building the DataOps team with stakeholders and data experts**

» **Choosing the right DataOps platform**

Chapter 6 Getting Started with DataOps

You can develop data applications that deliver value faster, with greater customer satisfaction, at a fraction of the cost of traditional data analytics. At enterprise scale, embracing DataOps is the way your business will remain competitive. Smaller companies that use data to gain insight can benefit by following many of the principles of DataOps, even if they aren't managing terabytes of data on a data platform. All data-centric businesses can benefit from the principles of DataOps, and in particular, #TrueDataOps.

Embracing the #TrueDataOps Philosophy

#TrueDataOps is a set of principles translated into technical practices, cultural norms, and an architecture that enables rapid experimentation and improved insights for customers. Of principal importance is the collaboration among data engineers, data scientists, analysts, stakeholders, and end-users. Collaborative practices are established across complex sets of people, technologies, and environments. This communication begins any #TrueDataOps adoption because buy-in to the philosophy is more important than purchasing tools. Without a shared understanding of how the application of these principles should work, you won't achieve consistent results. Collaboration, although made easier by DataOps tools, can't be automated. Everyone must participate.

REMEMBER

Although DataOps is not exactly DevOps, the #TrueDataOps philosophy is born out of the DevOps principles; therefore, in #TrueDataOps, you'll experience many of the life-changing strategies taught by DevOps that have the potential to supercharge your business. DataOps provides rapid time-to-value through Agile development using an orchestrated data pipeline, a source code control system that supports Agile development and automates testing, documentation, and audit trails for better governance.

Selecting the Right Business Case

When your data platform looks more like a data swamp and gaining insight from it seems like wizardry over a cauldron, it's time to begin unlocking its value one small project at a time. Pick a use case that will return value quickly and easily, then rinse and repeat. Starting the engine of continual development is all it takes. If you begin with a small project that leads to quick success, you'll motivate all the people involved to achieve more success with even more data. Before long, the platform will begin looking manageable and everyone will have bought into a new, and much more modern, way of working. Soon you'll have a well-organized data product including a catalog with detailed metadata that provides true business value.

Engaging with the Right Stakeholders

Discussions of data analytics or data platform development often begin with the technical team and the end-users. Instead, consider starting your DataOps journey by enlisting executive stakeholders who can best align your DataOps efforts with company goals. Once you have executive enthusiasm for your project, that support will permeate every department.

REMEMBER

Often these executive stakeholders already understand the benefits of DevOps and Agile development and are seeing the benefit of it elsewhere in the organization. The goal is not to explain something new, but to take something trusted and familiar and apply it in the context of data.

One of the challenges in some companies is that departments have created data silos. Thus, getting stakeholders from all departments to buy into the DataOps philosophy is crucial. Once these department heads understand how their data will be protected, they may be more willing to discuss how it can be transformed and incorporated into new data products that will benefit their departments as well as others in the company.

REMEMBER

When the goals have been set and departments are on board, collaboration among end-users, data scientists, data analysts, and data engineers will lead to data products that provide insight, are reusable and trusted, and are accessible from a catalog.

Building a DataOps Team with the Right Help

The goal of any DataOps team is to deliver high-quality data analytics rapidly and securely. The DataOps team should include the skills found in the following roles:

Data engineers perform crucial initial steps such as migrating data from business applications. These can include, for example, customer relationship management (CRM) or human resources systems. Data is moved to the data platform, and the data engineer develops the code to transform raw data and populate schemas within the data platform. Data engineers also understand test-driven development principles and help design and implement tests to assure data quality.

Data analysts and business analysts provide insights to clients from the data stored in the platform, know the data catalog inside and out, and know the tools that generate meaningful views of data. The insights usually provided by the data analyst generally answer questions about how things have performed in the past and how they are performing now. Increasingly, this role is filled by the business departments themselves and in DataOps platforms that support self-service (such as DataOps.live). In many cases, analysts are not simply consumers of the data but are making and submitting new changes to the data product, especially around how data is transformed and tested, to support their own consumption requirements.

Data scientists create algorithms that solve problems by creating unique combinations of data and working with data to provide forward-looking insights. This multi-disciplinary field is most associated with data mining of Big Data and the use of artificial intelligence (AI) and machine learning (ML) to glean new insights from distributed and massive data sets, often to produce predictions about the future.

TIP

AI and ML are here to stay, and businesses that use them to gain insights from their data are succeeding over those that don't. AI and ML require exceptional access to trusted data. Overcoming siloes, whether physical or virtual, based on security, is the way forward that allows AI access to heterogeneous data. This is true not only when using AI but for all data analytics.

DataOps engineers understand and apply the principles of DataOps to a data product. A team's DataOps engineer has a broad understanding of DevOps principles like Agile development, data orchestration in a DataOps environment, and the tools used to automate processes in the data pipeline. It may be possible to combine the skills of the DataOps engineer with other skills on the team, eliminating the need for a specific role as a DataOps engineer.

TIP

In DataOps, as in software development, not everyone on your team needs to know everything. Typically, a person with DataOps engineering skills focuses on building data and orchestration pipelines, with less concern about the details of individual steps, while a data engineer or analyst may rely on pipelines that have been built for them and focus on the internals of each job. This closely mirrors the relationship between DevOps and software engineers.

One other team or function — data governance — may also cross the disciplines. Data governance ensures that data is secure and meets all government and privacy requirements. Because of the high business cost of failure in this area, this function is crucial.

TIP

Because all the team members must know and adopt the overall principles of DataOps, collaboration should include a discussion of how to implement these principles and not return to older and less efficient methods.

Choosing the Right Technology Platform

Snowflake's data platform is built on a SQL engine designed to work in the cloud and separates compute from storage. Because it employs the software-as-a-service (SaaS) model, Snowflake has no equipment or maintenance overhead. It embodies the architectures of both shared-disk and shared-nothing architectures where data repositories are stored. Queries are performed using massively parallel processing (MPP) compute clusters, with each node in the cluster containing a portion of the data locally.

The goal in selecting the right DataOps technology platform is to find the platform that works best with the tools you already use. It should create an end-to-end orchestration of the tools you use and allow for continuous integration/continuous delivery (CI/CD) development. It should incorporate a repository that enables flexible development with branching, enforces security, creates an audit trail, and makes use of automated documentation and automated testing for governance.

Until recently, many organizations that wanted to follow a DataOps methodology had to build everything themselves. This typically involved selecting a range of technologies and undertaking significant work to assemble them into a viable DataOps platform. Figure 6-1 shows an example of one such assembly.

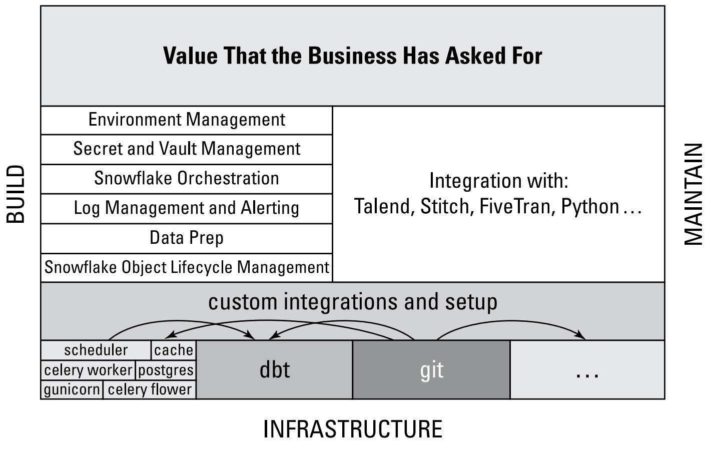

Value That the Business Has Asked For
Environment Management
Secret and Vault Management
Snowflake Orchestration
Log Management and Alerting
Data Prep
Snowflake Object Lifecycle Management
Integration with:
Talend, Stitch, FiveTran, Python …
BUILD
MAINTAIN
custom integrations and setup
scheduler
cache
celery worker
postgres
gunicorn
celery flower
dbt
git
…
INFRASTRUCTURE

FIGURE 6-1: Trying to build your own DataOps platform.

Initially, the work required to select and deploy these technologies is relatively small, though the overall work required for such an endeavor is far larger than most organizations think. It includes:

- Select technology components. Install and configure them correctly.
- Build comprehensive integrations between components.
- Build a large number of data- and enterprise-specific features on top of these components because none of them support these features.
- Create integrations with all the current and future data technologies you will use.
- Design and build a highly available and scalable architecture where you can run all of this.

The biggest pain point, however, is not the initial build. Rather, it's the ongoing maintenance and work required to keep the platform healthy and up to date with tens of different technologies and integrations, many of which are incompatible with different versions of each other.

All this work is necessary to build and maintain a steady state, but that's only the beginning. DataOps is progressing rapidly, and cloud data platforms like Snowflake are releasing advanced new features all the time. Your organization can't take advantage of these innovations if your DataOps platform doesn't support them.

TIP

Practically, the minimum team size required for an organization to assemble and build its own DataOps platform long term is five full-time staff members, which is rarely cost-effective.

REMEMBER

DataOps.live for Snowflake manages your DataOps pipelines and embodies all seven pillars of #TrueDataOps, discussed in Chapter 4. Your data, stored in Snowflake, moves through a pipeline of orchestrated tools in an extract, load, transfer (ELT) fashion with the tools you already use.

CI/CD with automated testing in a Git repository enables rapid, trusted, and parallel development using branches. Snowflake creates branched data sets using zero-copy cloning. New branches of the data product can be quickly created so that developers can work with a current copy of the production data and then destroy the copy when development and testing are complete.

TIP

Some DataOps platforms provide excellent DevOps capabilities (they are really DevOps platforms) with little understanding of data. Others have a good understanding of data and the data ecosystem but have poor DevOps capabilities. Only `DataOps.live` has both.

REMEMBER

The DataOps.live platform allows agility through simplified orchestration and management, parallel collaboration leading to faster development times, and automated testing.

Get on With It

DataOps is all about achieving faster time-to-value. The principles of Lean and Agile development promote diving in and getting started. Creating the team, getting them talking about requirements, and perhaps creating tests of how the results should appear, is enough to begin creating your data product.

Here are some pointers to keep in mind:

- Start with a clear understanding of how critical storing data in raw format through ELT can be.
- Create atomic and reusable code to begin creating value.
- Automate what you can at each step.
- Fail fast: Fix code often enough so errors don't appear in your data for users to find.
- Overcome siloed data wherever you find it; address physical silos and logical siloes or those created by security barriers.
- Involve the end-users and collaborate daily or even hourly.
- Avoid use cases where a large number of different data sets are needed to deliver even basic value.
- Don't forget the value of automated testing. Nothing shows more value than showing that the data the business has been using for years has errors in it!

TIP

Many of the most successful DataOps projects began as simple pilots. After a few weeks, these pilots showed such good results that the business adopted them as the new way of working. A few such projects have had terabytes of data in production and delivered business value in even less time — days rather than weeks!

IN THIS CHAPTER

» Learning why a move to DataOps makes sense

» Discovering why DataOps leads to lower costs and greater customer satisfaction

» Understanding how to justify DataOps to the rest of your business

Chapter 7 Ten Reasons to Move to DataOps

Today's world is data driven. Being able to gain insights from this data to make better business decisions is key to success. Many reasons exist to change your data project management to the DataOps method, but here are the ten most powerful.

Using What You Already Have

Many companies have a large investment in the tools they use to manage data, develop data products for analysis, and provide governance. If these tools can work in an orchestrated environment, you don't need to discard them, along with the training required to become proficient with them. Tying your existing tools together in an automated and orchestrated environment makes them more powerful. For example, not running data processing until the required sets of data ingestion have finished and passed automated testing ensures the processes don't overlap but run as quickly as possible. It also ensures that ingested data isn't processed if it's bad, or the ingestion process fails. The tools used may be unchanged but orchestrating them together provides considerable additional value.

In many cases organizations have legacy tools that they plan to move away from, but it's far easier to do this in a planned and controlled way if the new and old systems can be orchestrated as part of the same pipeline during a transition period. This also allows automated testing to be applied to confirm the functionality of the new system matches the old one.

Enabling Agility and Governance

Agility, the rapid cycle of development and integration, must be balanced with good governance, or projects will become ungoverned, unmanageable, and a nightmare at the end when you try to discern documentation or apply security after the fact.

Working within a repository such as Git, you can branch your code development so that development is done in parallel and merged later. Using a repository creates an audit trail of who made changes, when they were made, and why. It also layers in security by limiting access.

Typically, a good governance process imposes a minimum amount of time for a change to be fully pushed through the correct process — this can often be measured in days or even weeks. When a requirement is urgent or critical, good governance is often bypassed. By contrast, a good DataOps process should allow a complete change lifecycle in about an hour. Even urgent requirements can be done the right way, and no time or effort is wasted for routine requirements.

TIP

Automating your testing is an integral, and often overlooked, part of achieving agility and complying with good governance. Good governance should require you to ask, "How do I know that what I have changed doesn't break anything?" Automated testing ensures this.

Responding More Quickly and Shortening Time to Value

When you receive a data analysis or new feature request, the requirement is typically immediate. Lean principles promote the idea of providing a solution as quickly as possible with the smallest amount of work expended. You can begin immediately and then continue improving incrementally, in collaboration with the entire DataOps team, including the stakeholder. Thus, the client begins seeing results immediately. Even if the results must be refined, end-users have the satisfaction of immediacy, participation, and ownership of the end result. Ultimately a fast response to the business comes as a combination of:

- Getting started quickly and minimizing wasted manual effort setting up environments
- Efficient development
- Rapid testing and validation of technical functionality
- Rapid validation of functionality with the stakeholder
- Rapid ability to follow a good governance process
- Rapid deployment of changes into production once approved

TIP

All these factors combined can reduce the time to deliver a new requirement into a product from months to days, or even hours.

Demonstrating Assurance

Demonstrating assurance means not just knowing that you've done the right thing, or even that you have your own test results so you can demonstrate to the business that you're doing the right thing when asked. Demonstrating assurance means being able to proactively show the business that you know you're doing the right thing, 24/7, even when no one is asking. This capability gives the business much higher confidence in the data. Several reasons account for this, but the most fundamental is automated testing that provides results you can share with the business (for example, through a data catalog).

TIP

Try implementing a test-driven development methodology by testing your data as you develop against a test plan that you create before you even begin coding. You'll trap errors before the end-users catch them, further increasing stakeholders' confidence.

An audit trail of all changes, showing when they were made and why, allows you to document your development path and show test results. Finally, adhere to the #TrueDataOps principle that before you merge code, another person reviews it to make sure it's correct. This practice gives the business confidence in being protected even when someone is having a bad day.

TIP

Once your data is in production, #TrueDataOps suggests that you use automated monitoring. This approach makes sure your testing doesn't miss anything and that you catch errors in the data when they occur. It is most relevant for providing visibility and showing business users good results without them asking. Don't wait for users to ask how well you are doing and how good your data is — show them proactively. Greater data assurance leads to greater customer satisfaction and a lower workload on data teams.

Improving Security, Governance, and Change Control

Security concerns are what often drive stakeholders to create siloed data. Creating parameterized and anonymized views of the data allows it to be used and reused in data projects without compromising security.

TIP

Automated regression testing through the cycle, along with automated generation of documentation, accelerates you along your way without sacrificing critical governance requirements.

In addition, the security elements of data access (the permissions and grants on the data itself) are some of the most critical requirements to automate with DataOps. For example, if a mistake is made manually, adding or removing a column is undesirable. But if a mistake is made while manually applying a grant to data, the results might be catastrophic — potentially exposing personally identifiable information (PII) or commercially sensitive data to the wrong people. Automating the application of all security policies allows them to be tested safely in non-production environments, giving assurances that they will perform as required when deployed to production.

Lowering Project Costs

The Agile development method, along with development efficient tooling, gets you to results sooner. These improvements are possible, in part, because testing and documentation have been automated and implemented as an integral part of development rather than being tacked on at the end.

REMEMBER

Lowering the number of tasks done manually frees your data teams from mundane tasks to tackle projects more efficiently. Thus, you can effectively scale your teams instead of simply throwing more people at the growing need for business intelligence.

Because DataOps can automatically provision additional resources when necessary, it assures that your environment is always ready, no matter the load placed on it, but with no need to over-provision.

Taking Advantage of Modular Components and Code

Data products built from reusable, atomic code modules enable fast development with the least risk. Using pretested and proven code saves valuable human development time. Reusing data modules increases data assurance because the modules have been pretested and used in a production environment. They also lead to the development of a self-service environment where a greater understanding of what is already available avoids redundancy and engenders creativity.

Data products increasingly employ a large degree of potential repetition, applying the same logic again and again to different parts of the system. Repetition is the enemy of productivity and maintainability. A good DataOps approach allows procedural generation of code — for example, applying a simple snippet of code that can loop through a list and apply each element to a template. In many cases, this practice can reduce the volume of code by as much as 95 percent, meaning not just less work to implement but a dramatic reduction in duplication.

Increasing the Capacity of Your Team

Turning your data team into a DataOps team increases its capacity by incorporating collaboration, code reuse, continuous integration/continuous delivery (CI/CD) development, product orchestration, and automation. When all the mundane mechanics are either removed or fully automated, the team is free to devote most of its time (previously spent on these tasks) to providing business value.

The team's effective capacity can also be increased by allowing a greater degree of self-service. For example, a business analyst might write a requirement as a set of bullet points to be translated into code (with the risk and overhead of misunderstanding, incomplete requirements, and so on). Alternatively, the analyst can create a feature branch, define the logic as a simple SQL statement, and submit a merge request. At best, the core development team can simply merge the request; at worst, this method is a more rigorous way for analysts to define their needs. In most cases, the request needs only a bit of tweaking to prepare it to merge.

Collaboration Across an Organization

At the highest level, collaboration among stakeholders, data engineers, scientists, analysts, and end-users builds a shared vision. More tactically, a #TrueDataOps approach allows teams from an individual up to hundreds, or even thousands of contributors, to collaborate within a single project. However, true agility isn't only about collaboration within the development team; collaboration with stakeholders is crucial as well. A #TrueDataOps approach allows the team to develop a new feature or enhancement and share the results with stakeholders virtually, in real-time, to get their feedback.

REMEMBER

Agile practitioners don't just acknowledge that it's hard for stakeholders to communicate perfect requirements the first time; we actively seek stakeholder review of our work while in progress and solicit feedback and improvements to make the final results better. Involving end-users and department managers in the process creates a sense of ownership and can encourage a future interest in self-service.

Lowering Barriers to Success

For most businesses, a requirement for a 6- to 12-month data project before any value is seen is unacceptable. #TrueDataOps has shown that iteration and adaptation beat upfront planning almost every time, and therefore the value of a full, upfront detail plan is low anyway. Having an idea of what you want and how you want it to work is all you need to begin developing. By doing so, you can quickly show value to the business. You also create a much lower barrier for sign-off by the business.

DataOps.live

DataOps for Snowflake is a one-stop platform for 100% of your DataOps lifecycle to enable agility and responsiveness, with no compromise on data security and governance. It provides end-to-end orchestration, environment management, CICD, automated testing, and ELT wrapped in an elegant UI.

Find out more at www.dataops.live

Or join our community at www.truedataops.org

Handle data the same way you handle software

DataOps is a philosophy that has grown out of the successful DevOps culture and philosophy but adds its own elements for managing the things that are unique about data. This book helps you learn the basics of DataOps and the seven pillars of #TrueDataOps, a practical philosophy that, when followed, will guide your business toward rapid success, greater data assurance, proper governance, reduced costs, and better stakeholder satisfaction. It also gives practical advice on how to start your DataOps journey.

Inside…

- Balance Agile development with data governance and assurance
- Treat data as a product, not a project
- Increase collaborative development
- Build efficiency through trusted assets
- Work smarter with code reuse
- Automate data regression testing
- Enable self-service and data catalogs

DataOps

Justin Mullen is CEO and co-founder at DataOps.live. He has been building data systems for the last 30 years and was the first Snowflake partner in EMEA in 2017. **Guy Adams** is CTO and co-founder at DataOps.live. He is a passionate advocate of #TrueDataOps and the #1 Snowflake Data Superhero globally.

ISBN: 978-1-119-73678-3
Not For Resale

9 781119 736783