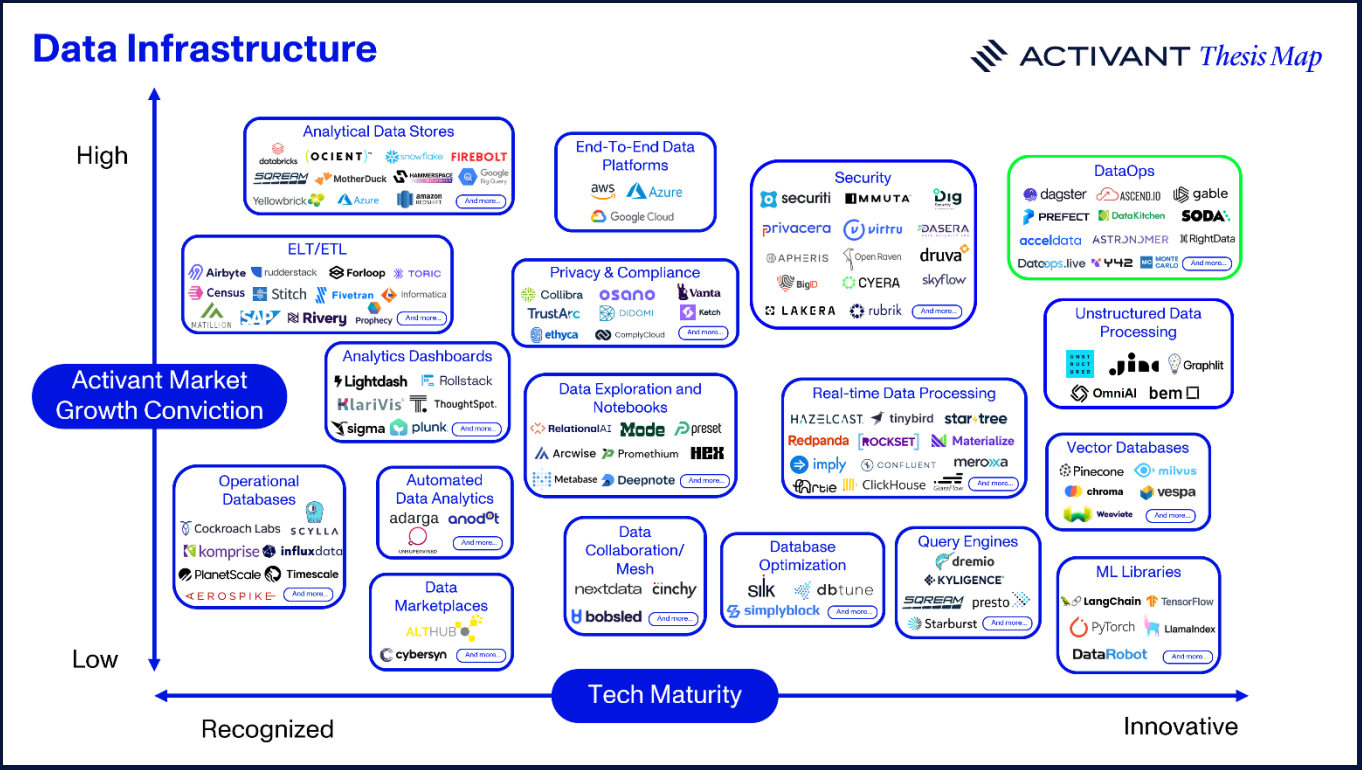
ACTIVANT RESEARCH

The DataOps Playbook

Lessons from the DevOps success story



Nina Matthews, Jonathan Vickery

Q4 2024

DevOps revolutionized software development 20 years ago and has grown into a robust ~$20 billion market.[1] Yet the data industry still lags, mired in issues DevOps solved long ago. As we discussed in our previous DataOps article, teams and data remain siloed as organizations rely on an average of 89 different software applications.[2] For the largest enterprises, this number climbs to 383, further increasing fragmentation.[3] Bureaucratic hurdles delay data projects by up to 18 months[4] and nearly half of data workflows remain manual.[5] The impact is severe: 87% of data analytics projects never reach production,44% can't track how their data flows through their organizations, and 96% of analytics leaders report stifled innovation due to data management challenges.[6,7,8] The good news? DataOps can learn from DevOps' success and build upon its proven playbook to solve all these issues for the data industry.

Although the technology landscape has advanced significantly since DevOps first transformed software development and data infrastructure presents its own unique complexities, there is much that DataOps can leverage to foster collaboration, create feedback loops, do away with complexity, and embrace innovation.

DevOps: The Success Story

In the early days of software development, developers (Devs) and IT operations (Ops) teams were two ships passing in the night. Developers would craft code, test it in their own environment, and then "toss it over the wall" to IT operations for deployment. This process involved no collaboration and no feedback loop. It was just a handoff, following the sequential legacy waterfall approach to development. A substantial amount of time and energy was wasted on manual code testing and integration, as well as on infrastructure provisioning and maintenance. This disjointed process led to frequent errors, frustrating delays, and a fair amount of finger-pointing.

As software became an ever-greater piece of modern enterprises, this model was no longer fit for purpose. Teams needed to change how they worked to meet demand. With DevOps, a transformative approach emerged: it broke down silos and brought teams together. As a result, teams now work in lockstep, with processes constantly becoming faster, smoother, and more reliable. Tasks that used to take days or weeks are now automated, freeing individuals up to focus on innovation rather than manual grunt work. What once seemed impossible is now standard practice.

But this transformation did not happen overnight. It was an evolution fueled by the adoption of Agile development practices and a cultural shift toward shared responsibility. As a result, the DevOps movement has grown into an undeniable success. By 2019, DevOps had been embraced by over 77% of organizations.[9] Large enterprises are realizing significant annual savings of $37 million to $231 million by reducing costs associated with system downtime, error resolution, and slow recovery times after failures, all made possible through DevOps practices.[10]

The Principles of Success

DevOps succeeded in addressing the longstanding challenges in software development and operations by adopting four core principles: 1) foster collaboration; 2) implement rapid feedback loops through automation; 3) abstract away complexity; and 4) embrace a culture of innovation. Together, these principles, which are explored in the sections below, form the DevOps playbook.

1. Foster Collaboration

Agile development methodology emerged to promote responsiveness and feedback throughout development. The goal was simple but powerful: break down slow, cumbersome projects into smaller sprints. Shorter sprints enabled faster feedback loops, in turn breaking down silos and laying the foundation for DevOps by promoting continuous improvement across teams. This need for responsiveness led to more frequent and constructive interactions and created a collaborative environment where everyone could align on shared goals and follow a clear, well-defined path to achieve them.

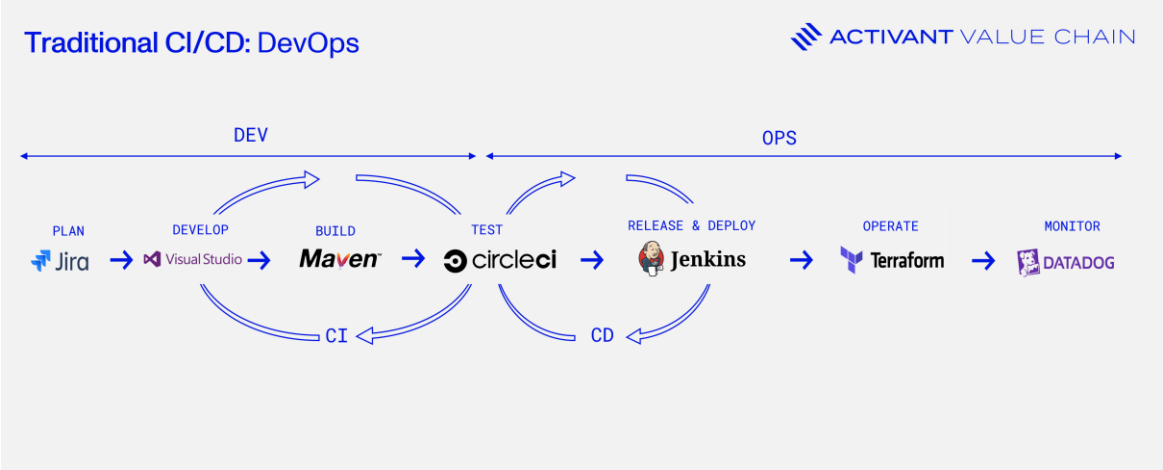
2. Implement Rapid Feedback Loops Through Automation

Rapid feedback loops were made possible by implementing Continuous Integration and Continuous Deployment (CI/CD) pipelines. Continuous Integration (CI) allows Devs to frequently and automatically add code to a shared, version-controlled repository like GitHub. Every change is tracked and reversible, enabling teams to add new features without the constant fear of breaking existing functionality. Incremental improvements are subject to automated testing through tools like CircleCI to catch issues before changes are merged, preventing errors from becoming roadblocks.

Continuous Deployment (CD) takes this a step further by automating the release process, pushing tested code directly to production with minimal human intervention and accelerating the development cycle. For the average IT team, deployment frequency can rise from 12 times per year to roughly 730.[11] Elite DevOps teams, like those at Netflix, have mastered automation to the point where they can deploy to production thousands of times a day.[12]

Imagine a team building a web application with each developer working on different features simultaneously, one on the homepage and another on the search bar. Without CI, they'd merge their work at the end, risking conflicts and tricky errors. But with CI, changes are integrated and tested frequently, catching issues early on. Add CD to the mix, and the new features automatically go live post-testing.

DevOps teams now follow the seven steps of the CI/CD pipeline, as illustrated below, to create rapid feedback loops between development, testing, and deployment.



1. **Plan**: Teams define project goals, requirements, and roadmaps using tools like Jira to align all stakeholders from the start.
2. **Develop**: Code is written and modified in integrated development environments (IDEs) like Visual Studio. Code changes are then managed using version control systems like GitHub and Bitbucket, which allow further team collaboration.
3. **Build**: Code is compiled and packaged to prepare it for testing. Automation tools like Maven and Gradle make this possible by turning code into deployable artifacts.
4. **Test**: Automated tests, which include unit, integration, and functional tests, are run before deployment through platforms like CircleCI or JUnit to validate the software and any changes made.
5. **Release & deploy**: Tools like Jenkins and Argo CD are used to automate the delivery pipeline by releasing and deploying tested code to production with minimal manual intervention, reducing the potential for errors.
6. **Operate**: Post-deployment, runtime environments need to be managed for scalability. Tools like Kubernetes are used for container orchestration, and Terraform to automate the provisioning of the underlying infrastructure.
7. **Monitor**: Alerts and dashboards are set up with tools such as Datadog or New Relic to track performance, identify issues, and gather insights for future improvements, continuously keeping tabs on the system's health.

When implemented effectively, CI/CD delivers impactful outcomes. Google's research shows that the average large enterprise can save up to $36.5 million annually by reducing unnecessary rework, while mid-sized enterprises can save around $8.6 million.[13] Proactive monitoring alone can drive additional savings, with the average enterprise benefiting from $9.6 million to $13.7 million annually by avoiding downtimes.[14] The financial impact is clear: these rapid feedback loops created by CI/CD pipelines not only streamline processes, they also deliver measurable value that justifies the investment.

3. Abstract Away the Complexity

Within the “operate” step of CI/CD, DevOps teams faced another challenge: managing the complex infrastructure needed to support their applications. Infrastructure as Code (IaC) addresses this by automating infrastructure provisioning and management, making it as scalable and flexible as the software itself.

Before IaC, setting up infrastructure manually was like building custom furniture without templates: fine for one piece, but impossible to scale without errors. Teams needed to evolve beyond configuring servers, networks, and storage by hand. IaC enables this by treating infrastructure as software**.** Devs can now write scripts to automatically provision servers (setting up all project prerequisites), load balancers (coordinating network and server traffic), and provide resource scaling (dynamically adjusting resource allocation).

In our web application example, when the team deploys a new feature like a search bar through CI/CD, IaC tools automatically update the infrastructure to keep the software and the hardware in sync. IaC tools like Terraform, Chef, and Ansible prevent configuration drift and ensure consistency across environments by automating setups across development, testing, and production. By abstracting away the complexities, IaC makes advanced infrastructure accessible to developers without requiring deep expertise in hardware management.

4. Embrace a Culture of Innovation

Peter Drucker once said, "Culture eats strategy for breakfast." This is especially true in the context of DevOps, where a cultural shift was the key driver of its global adoption, transcending management strategies**.** While tools and processes play a significant role, DevOps would not have succeeded without developers themselves embracing a culture of innovation. They took it upon themselves to find solutions best suited to their needs, proactively driving changes that benefited their workflows and productivity.

DataOps: Taking a Page Out of the DevOps Playbook

Data teams today are contending with challenges strikingly similar to those that Dev and Ops teams faced 20 years ago: siloed resources, slow development cycles, stagnant innovation, and costly, error-prone manual processes. With the pressures of growing data volumes, the complexities of unstructured data, and increasing demands for AI insights, all too familiar cracks are appearing in workflows. The obstacles and the notorious finger-pointing that DevOps once overcame are now hindering data teams.

Source: Maggie Hays, Now's the time to tackle Data Ownership, 2022

Data differs from code in several ways. It has inherent variability, is subject to strict regulations like the General Data Protection Regulation (GDPR) in the European Union and various privacy laws across the United States, and involves a diverse set of stakeholders with different needs. Despite these differences, the core principles of DevOps can still offer valuable guidance. So, why not take a page from the DevOps playbook? Startups in the DataOps landscape are adapting these principles to fit data teams' unique needs.

In the following sections, we explore four DataOps practices that can help teams unlock their full potential, drawing on proven strategies from DevOps: 1) foster collaboration with data contracts; 2) implement rapid feedback with CI/CD for data; 3) abstract away complexity with data products; and 4) embrace innovation with an asset-oriented data approach.

1. Foster Collaboration with Data Contracts

Think of data contracts as blueprints for data usage within an organization. They are formal agreements between data producers and consumers that define schemas, data types, quality standards, and access protocols. By establishing clear guidelines, data contracts help organizations adapt quickly to changing data needs while ensuring stakeholder alignment – fostering collaboration, accountability, and automation across teams much like Agile development has done for software projects.

As the volume of sensitive data is projected to grow five-fold by 2028, data contracts become crucial for maintaining data integrity and adhering to stringent regulatory requirements.[15] Companies like Gable are leveraging data contracts to create a centralized

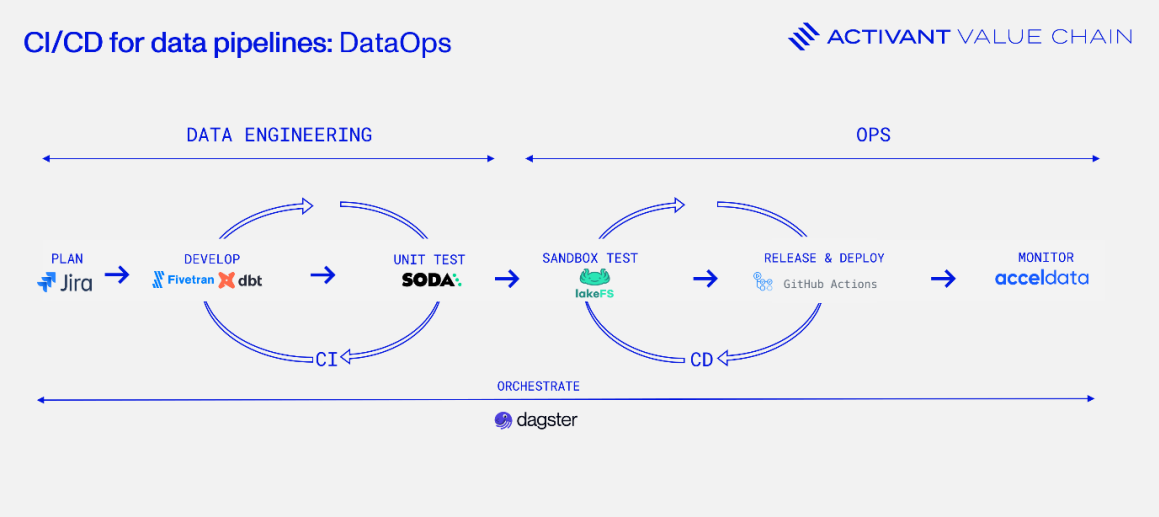
platform for data version control and collaboration, effectively making them "GitHub for data". This approach allows multiple stakeholders to work on data assets concurrently, with clear visibility into changes and history, just as developers have with code on GitHub.

This shared framework of data contracts not only ensures that sensitive information is managed correctly but also enhances transparency and accountability across teams. As a result, data management becomes more collaborative and efficient, enabling organizations to harness their most valuable asset: data.

2. Implement Rapid Feedback with CI/CD for Data

Data pipelines are in serious need of an update through automation. The traditional "dev-stage-prod" process can't keep up with the speed and complexity of today's data needs. In this setup, teams develop in a dev environment, test in a staging environment, and deploy to production. But staging is a full replica of production, making tests slow and costly, taking countless hours and thousands of dollars to run. Isolating changes becomes a major hurdle, as multiple contributors test simultaneously, making it difficult to pinpoint the root cause of issues.

To meet the demand for rapid feedback, faster deployment, and automation, DataOps teams should adopt CI/CD practices from DevOps. Here's how data teams can apply these principles to their pipelines:



1. **Plan:** Plan projects to align stakeholders and establish roadmaps, just like in DevOps.
2. **Develop**: ETL code created using tools like Fivetran and dbt should be stored in GitHub to allow for version control, rather than being kept locally. Storing code locally often results in a siloed workflow that slows development and hinders collaboration.
3. **Testing:** Replace manual checks with frequent automated unit and integration tests for every project change:

a) **Unit tests**: Testing tools like Soda and Great Expectations can be used to handle data validation and quality checks with minimal manual intervention. These tests should run automatically whenever new code is committed, ensuring immediate feedback.
b) **Sandbox**: "Ephemeral environments" or "pipeline sandboxes" can be spun up to run integration testing with tools like lakeFS and Snowflake's zero-copy clones. This allows integration testing without affecting live data, thereby enhancing efficiency, reducing costs, and simplifying issue isolation compared to the traditional "dev-stage-prod" approach.

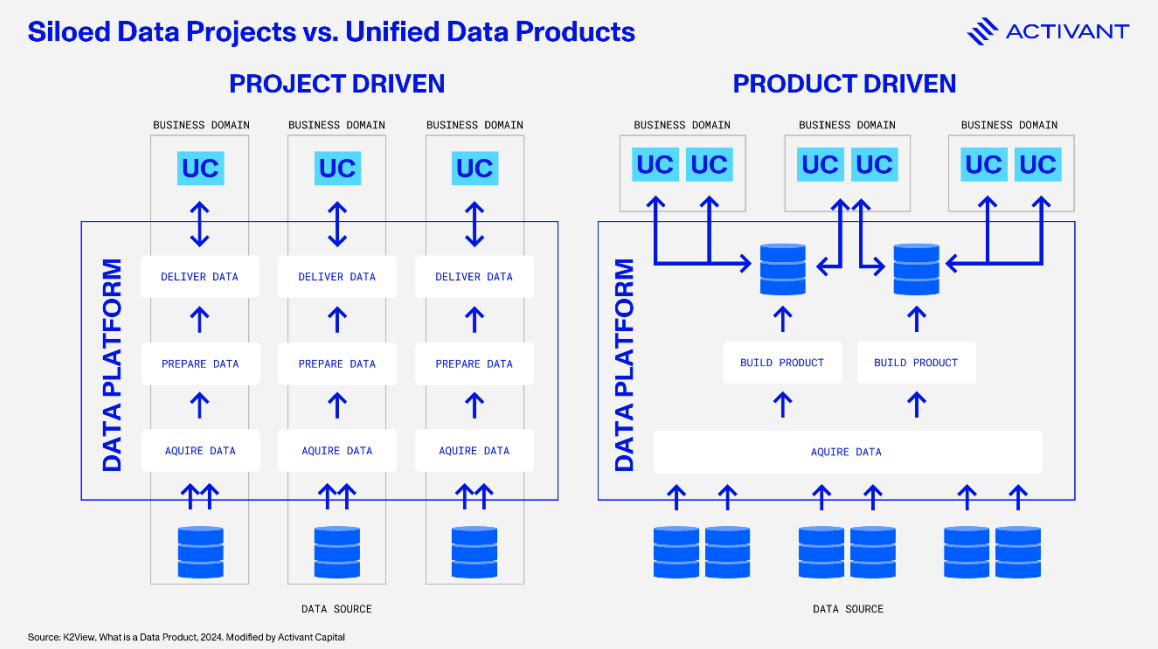
4. **Release & deploy**: Tested code for data can also be released and deployed to production with tools like Jenkins or GitHub Actions mirroring the process seen in DevOps.
5. **Monitor**: Real-time monitoring tools like Acceldata are crucial for tracking runtime errors and pipeline failures. Implementing these tools enables faster issue detection and resolution, ensuring the health of data pipelines.
6. **Orchestrate:** Recognize that data pipelines often involve complex, data-dependent operations requiring precise timing and sequencing. Use orchestrators like Prefect and Airflow to manage task sequencing, and consider tools like Dagster that treat data as assets to manage data flow and dependencies. These orchestrators can also integrate with IaC tools such as AWS CloudFormation and Terraform to provision underlying infrastructure, reducing bottlenecks between data and IT teams.

Adopting these practices may require an initial investment of time and resources, but the payoff is significant. Users report an 80% reduction in pipeline development time, translating to a 5x ROI through workflow automation.[16,17] These improvements reduce costly errors, speed up deployment, and ensure high-quality, reliable data pipelines, which are critical for data-driven decision making.

3. Abstract Away Complexity with Data Products

Under the DevOps model, developers do not need to be experts in infrastructure provisioning, they leverage IaC to abstract away the complexity. However, most data teams apply a ***project-driven*** approach to data, which often requires that data analysts are experts in the pipeline engineering that sits below their analytics dashboards. This approach keeps data siloed and limits collaboration across teams — only 20% of data leaders believe their platforms support data sharing across the entire organization, meaning most data remains underutilized.[18]

DataOps drives the shift to a ***product-driven*** approach, where data is treated as a well-designed product tailored to end users, emphasizing quality, usability, and ease of access. Instead of handling each data request as a separate project per use case (UC), companies make analytics-ready "data products" available through a centralized data hub or marketplace. This internal "store" lets analysts easily browse, select, and access the data they need, offering a seamless experience akin to online shopping and eliminating the need for lengthy preparation processes. Treating data as a product also makes it easier to scale as the core logic for each product resides within a platform and can be reused across teams.



Looking ahead, we believe this transformation will reshape data management, with most enterprises adopting the Data-as-a-Product approach in the next 5-10 years. Companies like RightData, Ascend, and Y42 are leading the way by designing data products that align perfectly with DataOps principles, automating workflows, speeding up time-to-market, and reducing silos, all while making data more reliable and accessible.

4. Embrace Innovation with an Asset-Oriented Approach

DataOps can achieve greater efficiency by adopting an innovative asset-oriented approach instead of the traditional workflow-oriented methods seen in DevOps. Most modern data stack tools, from ingestion to visualization — such as Snowflake, Monte Carlo, and Fivetran — are asset-oriented, ensuring seamless compatibility across the stack.

However, standard orchestrators like Airflow adopt a workflow-oriented approach around sequential tasks (step A to step B), diminishing transparency and control at each stage. This creates observability gaps and complicates data quality and governance, making it harder to optimize and manage data effectively.

Embracing an asset-oriented orchestration tool like Dagster allows teams to define, track, and manage assets with greater granularity, providing a unified view of the data stack. This strategy effectively addresses the complexities of modern data environments by enhancing compatibility across the data stack, thereby empowering DataOps teams to maintain high standards of quality and governance.

Final Thoughts

In much the same way that DevOps transformed the software landscape, growing into a $20 billion industry, so DataOps has the potential to become a cornerstone of modern data management. The parallels are clear and the opportunity is immense. We're excited by the companies leading this transformation: Gable is advancing data contracts; Soda and lakeFS are pushing the boundaries of data quality; Acceldata is driving advancements in monitoring; Prefect is streamlining orchestration; RightData, Ascend, and Y42 are innovating in data products; and Dagster is spearheading the implementation of an asset-oriented approach to data management.

The playbook is ready, and the infrastructure is in place. Now it's time for data teams to take the field and execute.

End Notes

[1] IDC, Worldwide DevOps Software Tools Forecast, 2021 - 2025, 2021

[2] Okta, Businesses at Work, 2022

[3] Ibid

[4] Saagie, The ultimate guide to DataOps, 2019

[5] Forrester, DataOps Can Build The Foundation For Your Generative AI Ambitions, 2024

[6] Gartner, Gen AI Seminar attended by Activant Research, 2024

[7] Ibid

[8] Ibid

[9] IDC, Addressing the Challenge of Adopting and Scaling DevOps, 2020

[10] Google Cloud, The ROI of DevOps Transformation, 2020

[11] Ibid

[12] Ibid

[13] Ibid

[14] Ibid

[15] Rubrik, The State of Data Security, 2023. *Statistics reflect 5000+ Rubrik customers across 67 countries*

[16] Enterprise Strategy Group, Analyzing the Economic Impact of Ascend Data Automation, 2023

[17] Senior Data Engineer at Clearcover, Tegus Interview, 2022

[18] IDC, Unlock Data Value by Enabling Data Product Sharing, 2024