DATA KITCHEN

The Seven Steps to Implement DataOps

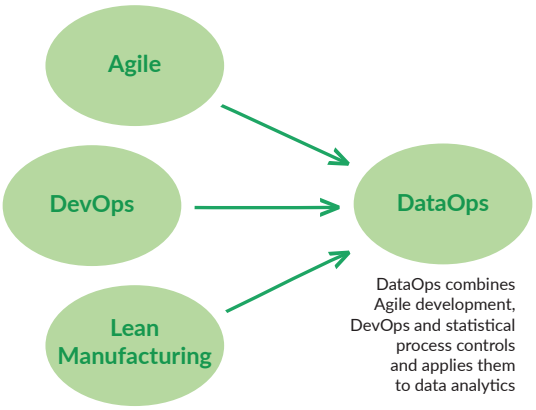
ABSTRACT

Data analytics teams challenged by inflexibility and poor quality have found that DataOps can address these and many other obstacles. DataOps includes tools and process improvements that enable faster, more responsive data analytics while maintaining a high level of quality and reliability. Data analytic teams can implement DataOps in seven simple steps: (1) Add data and logic tests, (2) Use a version control system, (3) Branch and merge, (4) Use multiple environments, (5) Reuse and containerize, (6) Parameterize your processing and (7) Use simple storage.

For a more detailed overview of DataOps, please see our companion white paper, "High-Velocity Data Analytics with DataOps."

INTRODUCTION

Data analytics has become business critical, but requirements quickly evolve and data-analytics teams that respond to these challenges in the traditional ways often end up facing disappointed users. DataOps offers a more effective approach that optimizes the productivity of the data analytics pipeline by an order of magnitude.

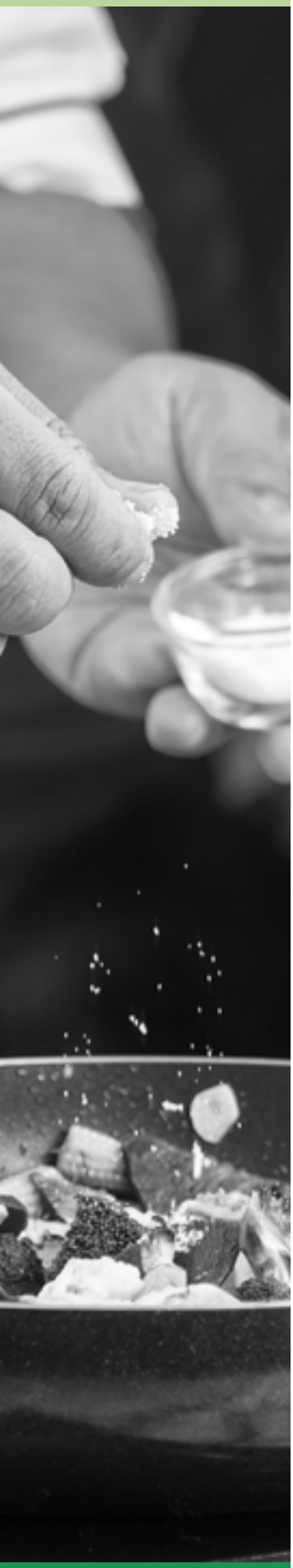


DataOps is a tools and process change that incorporates the speed of Agile software development, the responsiveness of DevOps, and the quality of statistical process controls (SPC) widely used in manufacturing.

Like Agile development, DataOps organizes the team and its processes around the goal of publishing releases to users every few days or even every few minutes. Each release contains working and valuable changes to the code base. Improvements are made and published quickly and feedback from users is incorporated into future releases as soon as possible. This approach is particularly good for non-sequential analytics development where requirements are quickly evolving.

Like DevOps, DataOps utilizes the automated provisioning of resources (infrastructure as code) and cloud services (platform as a service) to break down the barriers between IT, software development, quality assurance and other groups. The cloud provides a natural platform that allows individuals in each stage of development to create and define identical run-time environments. This minimizes errors, misunderstandings and delays.

Data-analytics is a pipeline process much like software development and manufacturing. It executes a set of operations and attempts to produce a consistent output at a high level of quality. With DataOps this pipeline is highly automated. Statistical process controls are



used to test, monitor and manage the robustness and consistency of each stage of the data-analytics pipeline. With an automated test suite, changes can be quickly verified and approved supporting the continuous development and deployment of enhancements to data analytics.

Imagine the next time that the Vice President of Marketing requests a new customer segmentation, by tomorrow. With DataOps, the data-analytics team can respond 'yes' with complete confidence that the changes can be accomplished quickly, efficiently and robustly. How then does an organization implement DataOps? You may be surprised to learn that an analytics team can migrate to DataOps in seven simple steps.

STEP 1 - ADD DATA AND LOGIC TESTS

If you make a change to an analytic pipeline, how do you know that you did not break anything? Automated testing insures that a feature release is of high quality without requiring time-consuming, manual testing. The idea in DataOps is that every time a data-analytics team member makes a change, he or she adds a test for that change. Testing is added incrementally, with the addition of each feature, so testing gradually improves and quality is literally built in. In a big run, there could be hundreds of tests at each stage in the pipeline.

Adding tests in data analytics is analogous to the statistical process controls that are implemented in a manufacturing operations flow. Tests insure the integrity of the final output by verifying that work-in-progress (the results of intermediate steps in the pipeline) matches expectations. Testing can be applied to data, models and logic. The table below shows examples of tests in the data-analytics pipeline.

EXAMPLE TESTS

Inputs	**Verifying the inputs to an analytics processing stage** Count Verification - Check that row counts are in the right range, ... Conformity - US Zip5 codes are five digits, US phone numbers are 10 digits, ... History - The number of prospects always increases, ... Balance - Week over week, sales should not vary by more than 10%, ... Temporal Consistency - Transaction dates are in the past, end dates are later than start dates, ... Application Consistency - Body temperature is within a range around 98.6F/37C, ... Field Validation - All required fields are present, correctly entered, ...
Business Logic	**Checking that the data matches business assumptions** Customer Validation - Each customer should exist in a dimension table Data Validation - 90 percent of data should match entries in a dimension table
Output	**Checking the result of an operation, for example, a cross-product join** Completeness - Number of customer prospects should increase with time Range Verification - Number of physicians in the US is less than 1.5 million

For every step in the data-analytics pipeline, there should be at least one test. The philosophy is to start with simple tests and grow over time. Even a simple test will eventually catch an error before it is released out to the users. For example, just making sure that row counts are consistent throughout the process can be a very powerful test. One could easily make a mistake on a join, and make a cross product which fails to execute correctly. A simple row-count test would quickly catch that.

Tests can detect warnings in addition to errors. A warning might be triggered if data exceeds certain boundaries. For example, the number of customer transactions in a week may be OK if it is within 90% of its historical average. If the transaction level exceeds that, then a warning could be flagged. This might not be an error. It could be a seasonal occurrence for example, but the reason would require investigation. Once recognized and understood, the users of the data could be alerted.

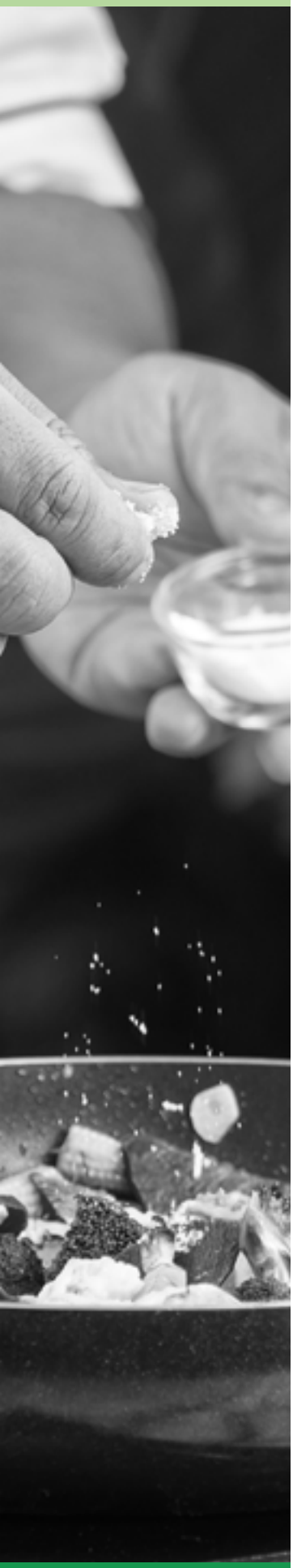
DataOps is not about being perfect. In fact, it acknowledges that code is imperfect. It's natural that a data-analytics team will make a best effort, yet still miss something. If so, they can determine the cause of the issue and add a test so that it never happens again. In a rapid release environment, a fix can quickly propagate out to the users.

With a suite of tests in place, DataOps allows you to move fast because you can make changes and quickly rerun the test suite. If the changes pass the tests, then the data-analytics team member can be confident and release it. The knowledge is built into the system and the process stays under control. Tests catch potential errors and warnings before they are released so the quality remains high.

STEP 2 - USE A VERSION CONTROL SYSTEM

There are many processing steps that turn raw data into useful information for stakeholders. To be valuable, data must progress through these steps, linked together in some way, with the ultimate goal of producing a data-analytics output. Data may be preprocessed, cleaned, checked, transformed, combined, analyzed, and reported. Conceptually, the data-analysis pipeline is a set of stages implemented using a variety of tools including ETL tools, data science tools, self service data prep tools, reporting tools, visualization tools and more. The stages may be executed serially, but many stages can be parallelized. The pipeline is deterministic because the pipeline stages are defined by scripts, source code, algorithms, html, configuration files, parameter files, containers and other files. All of these items are essentially just code. Code controls the entire data-analytics pipeline from end to end in a reproducible fashion.

The artifacts (files) that make this reproducibility possible are usually subject to continuous improvement. Like other software projects, the source files associated with the data pipeline should be maintained in a version control (source control) system such as Git. A version control tool helps teams of individuals organize and manage the changes and revisions to code. It also keeps code in a known repository and facilitates disaster recovery. However, the most important benefit of version control relates to a process change that it facilitates. It allows data-analytics team members to branch and merge.



STEP 3 - BRANCH AND MERGE

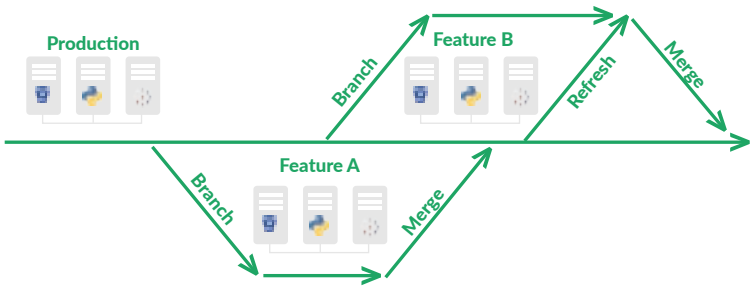
In a typical software project, developers are continuously updating various code source files. If a developer wants to work on a feature, he or she pulls a copy of all relevant code from the version control tool and starts to develop changes on a local copy. This local copy is called a branch. This approach can help data-analytics teams maintain many coding changes to the data-analytics pipeline in parallel. When the changes to a branch are complete and tested, the code from the branch is merged back into the trunk, where the code came from.

Branching and merging can be a major productivity boost for data analytics because it allows teams to make changes to the same source code files in parallel without slowing each other down. Each individual team member has control of his or her work environment. They can run their own tests, make changes, take risks and experiment. If they wish, they can discard their changes and start over. Another key to allowing team members to work well in parallel relates to providing them with an isolated machine environment.

STEP 4 - USE MULTIPLE ENVIRONMENTS

Every data-analytics team member has their own development tools on their own laptop. Version control tools allow team members to work on their own private copy of the source code while still staying coordinated with the rest of the team. In data analytics, a team member can't be productive unless they also have a copy of the data that they need. Most use cases can be covered in less than a Terabyte (TB). Historically, disk space has been prohibitively expensive, but today, at less than $25 per TB per month (cloud storage), costs are now less significant than the opportunity cost of a team member's time. If the data set is still too large, then a team member can take only the subset of data that is needed. Often the team member only needs a representative copy of the data for testing or developing one set of features.

When many team members work on the production database – it can lead to conflicts. A database engineer changing a schema may break reports. A data scientist developing a new model might get confused as new data flows in. Giving team members their own Environment isolates the rest of the organization from being impacted by their work. The diagram to the right shows how version control, branching and merging, and multiple environments work together.



STEP 5 - REUSE & CONTAINERIZE

Another team productivity tool is the ability to reuse and containerize code. Each step in the data-analytics pipeline is the output of the prior stage and the input to the next stage. It is cumbersome to work with an entire data-analytics pipeline as one monolith, so it is common to break it down into smaller components. It's easiest for other team members to reuse smaller components if they can be segmented or containerized. One popular container technology is Docker.

Some steps in the data-analytics pipeline are messy and complicated. For example, one operation might call a custom tool, run a python script, use FTP and other specialized logic. This operation might be hard to set up because it requires a specific set of tools, and difficult to create because it requires a specific skill set. This scenario is another common use case for creating a container. Once the code is placed in a container, it is much easier to use by other programmers who aren't familiar with the custom tools inside the container, but know how to use the container's external interfaces. It is also easier to deploy that code to each environment.

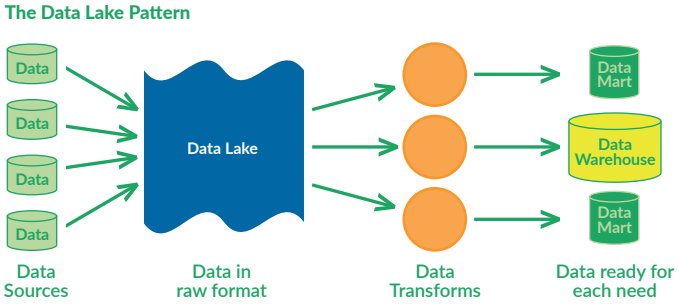
STEP 6 - PARAMETERIZE YOUR PROCESSING

There are cases when the data-analytic pipeline needs to be flexible enough to incorporate different run-time conditions. Which version of the raw data should be used? Is the data directed to production or testing? Should records be filtered according to some criterion (such as private health care data)? Should a specific set of processing steps in the workflow be included or not? To increase development velocity, these options need to be built into the pipeline. A robust pipeline design will allow the engineer or analyst to invoke or specify these options using parameters. In software development, a parameter is some information (e.g. a name, a number, an option) that is passed to a program that affects the way that it operates. If the data-analytic pipeline is designed with the right flexibility, it will be ready to accommodate different run-time circumstances.

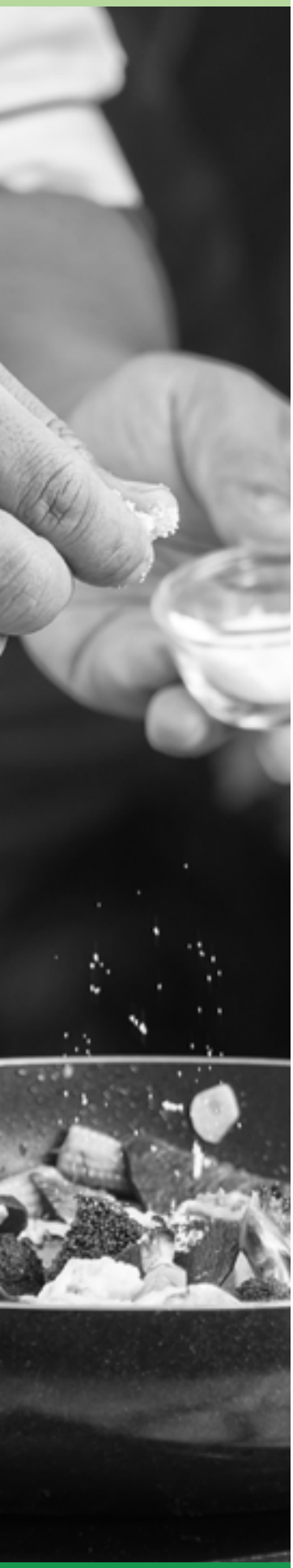
For example, imagine a pharmaceutical company that obtains prescription data from a 3rd party company. The data is incomplete, so the data producer uses algorithms to fill in those gaps. In the course of improving their product, the data producer develops a different algorithm to the fill in the gaps. The data has the same shape (rows and columns), but certain fields are modified using the new algorithm. With the correct built-in parameters, an engineer or analyst can easily build a parallel data mart with the new algorithm and have both the old and new versions accessible through a parameter change.

STEP 7: USE SIMPLE STORAGE

Third party vendors provide simple highly reliable storage that can scale on demand. This simple storage is an ideal way to create data repositories, data lakes, data warehouses and data marts for analytics. It provides backup and restore service, disaster recovery and makes it very easy to create and destroy copies of a dataset. It is often possible to create a copy of a multi-terabyte database and be up and running in around 10 minutes. Cloud storage vendors offer all this convenience and service at a relatively low cost.



Simple storage has made data lakes cost effective. Data lakes utilize simple storage to retain data in its raw or original format. When the structure of the data is optimized for a specific use case, it is populated in a data mart or data warehouse. Having been purpose built and



serving active users, data marts and data warehouses are often difficult to modify without impacting users. The ability to go back to the original data lake makes it much simpler to generate new data marts and data warehouses that are adapted to new requirements. This is often done by starting with the original data lake and modifying the code that created the data warehouse. Utilizing data lakes is a key component of achieving high-velocity DataOps.

CALL TO ACTION

DataOps can accelerate the ability of data-analytics teams to create and publish new analytics to users. It requires an Agile mindset and must also be supported by an automated platform which incorporates existing tools into a DataOps development pipeline as summarized by the "Seven Steps" above. A well-designed data-analytics pipeline can serve as a competitive advantage for data-driven organizations. For more information on the benefits of DataOps, please see our white paper "High-Velocity Data Analytics with DataOps."

For more information about how you can sharply improve the responsiveness of your data analytics, please contact info@datakitchen.io or see www.datakitchen.io.

ABOUT DATAKITCHEN

DataKitchen, Inc. enables analytic teams to deliver value quickly, with high quality, using the tools that they love. DataKitchen provides the world's first DataOps platform for data-driven enterprises, enabling them to support data analytics that can be quickly and robustly adapted to meet evolving requirements. DataKitchen is leading the DataOps movement to incorporate Agile Software Development, DevOps, and manufacturing based statistical process control into analytics and data management. DataKitchen is headquartered in Cambridge, Massachusetts.

RESOURCES

The Agile Manifesto

http://agilemanifesto.org/

Scrum Guides

http://www.scrumguides.org

Wikipedia DevOps

https://en.wikipedia.org/wiki/DevOps

Statistical Process Control

https://en.wikipedia.org/wiki/Statistical_process_control