snapLogic
Integrate Accelerate

THE DATA LAKE DATA INTEGRATION CHALLENGE

DAVID S. LINTHICUM

LINTHICUM RESEARCH

The reality is that data lakes require data integration solutions that can deal with structured and unstructured data, as well as deal with schema-less data storage. They also need to deal with streams of data that function in real time. In other words, a data lake requires a completely different approach to data integration, and it needs newer data integration technology to drive success.

Table of Contents

Executive Summary

The future for big data processing lies in the adoption of commercial Hadoop distributions (Cloudera, Hortonworks, MapR) and their supported deployments on AWS, Google, and MS Azure, along with the emergence of Databricks (Spark) and Datastax (Cassandra / Solr) implementations.

Of course, the macro use case for big data are data lakes. They are massive amounts of structured and unstructured data that do not carry the same restrictions as traditional data warehouses. They store everything, including every type of data, any volume, any scope of data that may be of use by enterprise data users, for any reason.

What's the value of the end result? The ability to find out the reliability of a supplier by simply scanning through PDF files of past invoices (unstructured), as compared to order data in a relational database (structured). Or, the ability to find inventory systems' usage statistics as related to cost data by comparing text-based log files (unstructured), with usage-based accounting that exists in a No-SQL database (semi-structured). The combinations and potential of this technology are endless, as well as the value that it can bring as enterprises now have access to any data, at any time, and in any pattern of use that's possible.

Despite the power and potential of data lakes, many enterprises continue to approach this technology and their new analytics infrastructure with the same data integration approaches and mechanisms they've used in the past. This may be an ESB (enterprise service bus), older data brokers, legacy ETL (extract, transform, load) tools, or even custom code to drive data integration. None of which work well.

The reality is that data lakes require data integration solutions that can deal with structured and unstructured data, as well as deal with schema-less data storage. They also need to deal with streams of data that function in real time. In other words, a data lake requires a completely different approach to data integration, and it needs newer data integration technology to drive success. This is the core topic of this paper.

Other issues that are consistent within data lakes include:

- Lack of highly-skilled resources to manage data in data lakes. Automation becomes the path to success, and not a reliance on people to deal with data complexities. Data integration technology must abstract the users away from the complexities, and we're looking to automate pretty much everything related to moving data from storage.
- Complexity of the big data ecosystem, including the lack of consistent schemas, the lack of schemas, and dealing with many patterns of data that constantly change within data lakes. Data integration technology must be able to deal with all types of data, in all types of ways.
- Constant change in big data technology, including new versions as well as new technology. This means that data lakes are constantly evolving, and data integration must keep up with the changes.

Key takeaways from this paper include:

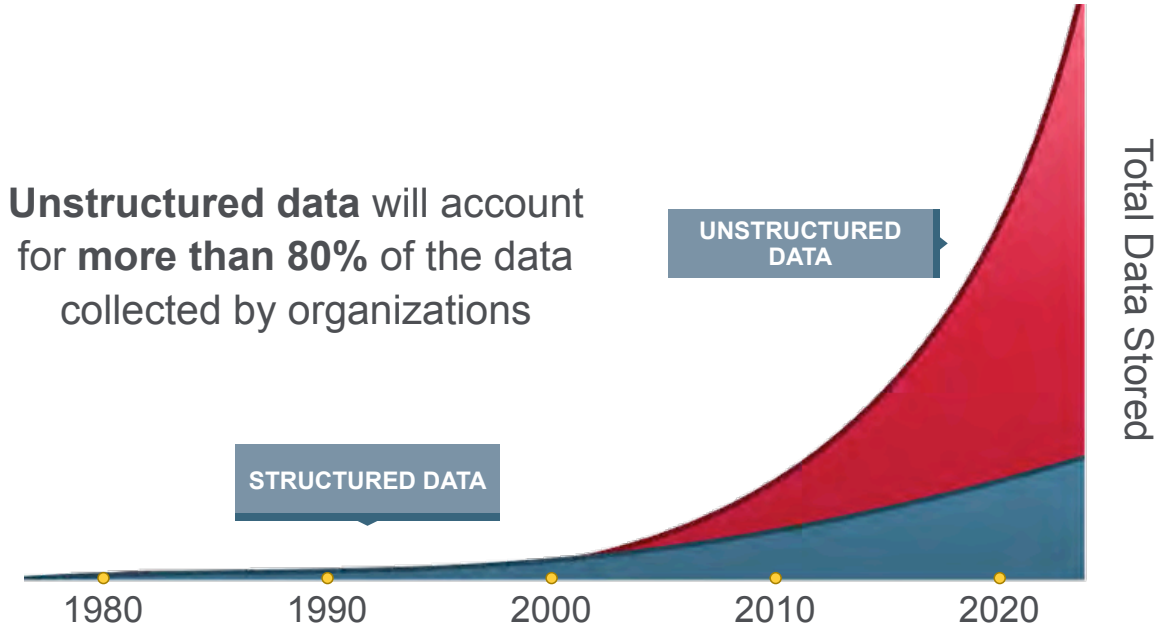
- Data lakes bring new value, as well as new complexity to enterprise data.
- Existing approaches to data integration, including ESBs, ETL, and other older approaches and technology, do not solve the core data integration needs of a data lake.
- The all-inclusive approach to data that data lakes use means that data integration needs technology to adapt to the growing complexity without requiring highly skilled IT talent.
- Data integration planning should become systemic to planning a data lake, or any other big data solution.
- Consider data governance and data security with your data lake and linked data integration solution.

Data Lake Use Cases

The core value of a data lake is that there are no longer restrictions on the type of data that you can store, as there were with traditional data warehouses. This means all data contained within an enterprise can now be leveraged. The value of this opportunity is a game changer.

As with the examples provided in the executive summary, we're looking at some meta-use cases that will provide the most value, including:

- ***The ability to compare data that's held in a well-defined schema, with unstructured or semi-structured data.*** This means that we now have the ability to make sense of data in PDF files, text files, CDFs, spreadsheets, and even multi-media binary files, which account for the largest portion of data within most enterprises. As you can see from Figure 1 below, unstructured data is outpacing structured data going forward, and the ability to leverage unstructured data becomes a priority for enterprises that are looking to understand all they can about their business as is, and in the future.



Source: Human-Computer Interaction & Knowledge Discovery in Complex Unstructured, Big Data

Figure 1: Unstructured data will continue to outpace structured data, and enterprises need to leverage data lakes to control both.

- ***The ability to leverage all data for predictive analytics to understand future trends of the business.*** Predictive analytics means that IT workers can determine future patterns from existing or past patterns of data. For instance, say we're looking to determine sales growth for a particular product. We would need to consider historical structured and unstructured data, such as recorded past sales, past social media data, and perhaps past key economic indicators. Using this data, we could determine future patterns by considering the correlation with known current data, and the likely resulting patterns based upon growing likely trends into the future. If done correctly, it's remarkably accurate.

- ***The ability to persist all data in a single location, and thus have a single source of truth.*** Data redundancy issues have often plagued enterprises. For instance, in how many places does the enterprise store customer data? The typical answer is "Many." As a result, you have a data consistency issue. When done right, the enterprise data lake should provide a single source of truth, in that the data is stored in a single logical location, and you can leverage specific data, such as customer data, and do so consistently.

Enterprise use cases number in the hundreds, when it comes to leveraging a data lake, including the emerging use of cloud computing, Internet of Things, and any other technology trend that is largely data driven. Indeed, most data lakes find a place on public clouds, and this trend will likely continue in the years to come.

According to a 2016 study by Nomura Research, "CIOs Are Prioritizing Big Data Analytics, Cloud Computing and Security In 2016/2017 Budget Cycles," 82% of CIOs cite security as the top driver for IT spending this year, followed by cloud computing (62%) and big data analytics (60%). Note that both data analytics (data lakes) and cloud computing are second and third only to security. We can assume that the need for security is driven by the desire to move to data analytics to the cloud. Figure 2 compares the top drivers of IT spending for respondents' companies.

When CIOs are asked how they are prioritizing big data analytics, cloud computing, and security in the 2016/2017 budgets.

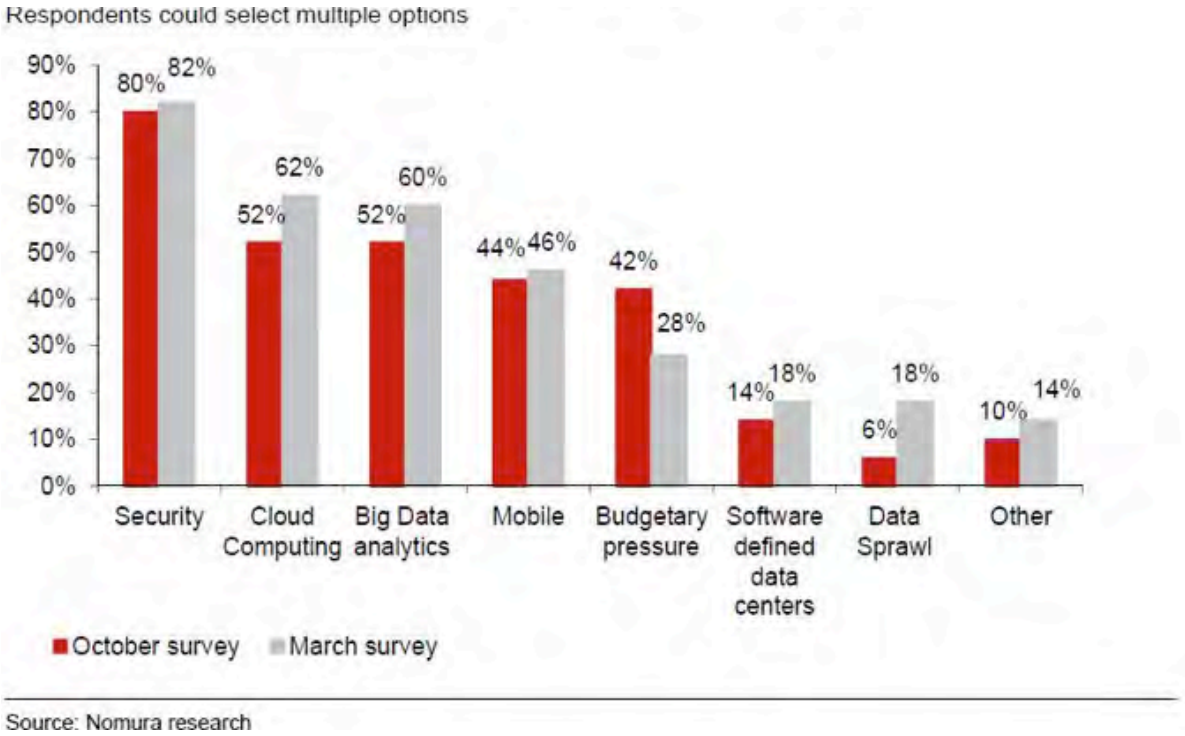


Figure 2: Massive budget amounts of budget are now being spent on both cloud computing

Traditional Approaches and Tools Don't Work!

Data lakes provide huge value, and yet its data integration needs are not being met. The reasons can be boiled down to a few core issues, including:

- Data needs to leave and arrive in the data lake in real-time. This means that we're moving away from a batch-oriented approach to streaming. Existing (or legacy) data integration solutions, such as ESBs and ETL tools, can't deal with real-time data, or can't handle the low-latency requirements of a data lake.

- Both unstructured and structured data needs to be shared, both consumed by the data lake, and produced by the data lake. Traditional data integration tools assume that the data has a structure, and these tools apply structure at the time of use within the data lake. We can no longer assume that data will use a consistent structure, and older tools are not built to deal with this kind of data.

- The use of data aggregation (aka, data virtualization) means that, the data integration engine must often times interact with interfaces that are just an aggregation of data, and not a physical layer. While traditional data integration technology is good at dealing with well-structured data, data lakes mean that we're dealing with structures that are likely to constantly change, perhaps every day. Traditional tools are not built to deal with change, or place volatility into a domain.

- Traditional tools require that you maintain a technical staff that understands both the tool, as well as the source and target data sources. This means that you must spend on staff and on training because the use of traditional data integration tools is often laborious. This drives up cost of ownership, and drives down the agility around the use of data.

- In addition to unstructured data types, traditional tools also don't work and play well with newer data security and data governance systems (see Figure 3). The issue is that traditional tools are not designed from the ground up to work with data governance systems that track and control the changing of data inside of the data lake. Proper governance tools are vital to maintain data integrity and insure that the data is usable going forward, but unless the data integration tool is "governance aware," the data governance technology is useless. Same goes for security, considering that traditional tools typically don't work and play well with newer security approaches and tools, such as identity and access management (IAM).

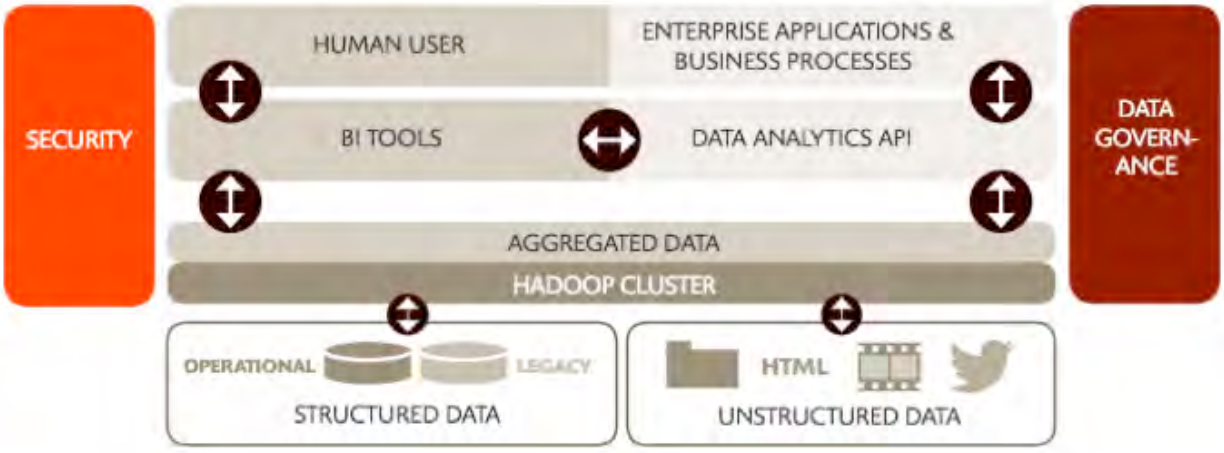


Figure 3: Data lakes are made up of many components, which makes data lakes very complex. Modern data integration tools must know how to deal with each layer.

Source: http://resources.idgenterprise.com/original/AST-0131518_BigData-Pivotal_v2.pdf

Custom Code

By now we should all understand that custom coding our integrations with source and target data is never a good idea. However, it's a practice that continues to this day.

The reasons for not coding interfaces and data movement between source and target systems hold true, no matter if we're dealing with a data lake or not. They include the fact that it's much more expensive to deal with custom integrations when you consider ongoing maintenance, with the core responsibility of troubleshooting integrations falling on developers, not data integration SMEs.

Management is another challenge, considering that integration must often change, and you'll end up spending an increasing amount of money to update and test data integration code. Moreover, there are so many points of integration that the tipping point for managing them is hit early and often.

Traditional Tools

ESBs provide an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit messaging through the use of services. For our purposes, you can consider ESBs as messaging systems that can internalize and externalize information via services, whereas in the past these message queuing systems leveraged proprietary APIs.

The problem is that all ESBs are different, and thus ESBs don't provide consistent integration patterns that can be leveraged by IT. To compound matters, there is no single definition of ESB, thus many middleware systems that call themselves an ESB have very different patterns from vendor to vendor.

The most common use of an ESB is as an information message and movement engine, where the information flows from one peer to another, using a queuing type of infrastructure. Some ESBs go beyond basic queuing to provide many of the services found in traditional integration servers, including transformation, routing, flow control, and even process integration and orchestration.

ESBs fail to provide good integration for data lakes for the following reasons:

- Messaging, by definition, is highly structured. Thus most ESBs are not built to deal with unstructured data, and thus can't deal with all of the data stored within a data lake.
- ESBs are high latency messaging engines, and are not built to deal with streaming data, or the use of data around a real time requirement.
- ESBs require that data be pushed to them, and they are not built to extract data around events, and this must be done programmatically.
- ESBs are more service-oriented than data-oriented, and thus don't focus as much on the data, but on services that interact with the data.

Backing up for a bit, ESBs really became a way to label JMS-based middleware, which was service-enabled as "ESBs." Built around the interest in SOA (services oriented architecture), almost all of vendors that had anything resembling an integration server, message-oriented middleware, or a message broker, re-labeled their product as an "ESB." Thus, you had dozens of ESBs out there, all different, and all supporting the notion of SOA and data integration in different ways.

Another traditional approach to integration is ***extract, transform, and load (ETL).*** ETL was designed to deal with larger volumes of data, typically in support of traditional data warehouses and data marts. ETL was designed around batch data processing, typically moving and transforming large amounts of data from one data store, such as a transactional database, to another database, such as a large relational database that is used as a data warehouse.

In other words, ETL has the same limitations around dealing with unstructured data as ESBs. This dependence upon a structure means that ETL engines are useless when dealing with data lakes.

ETL tools only focused on data that had to be copied and changed. Emerging data lakes use large amounts of data by largely leaving data in place, and instead access and transform data where it sits, no matter if it maintains a structure or not, and no matter where it's located.

Moreover, traditional tools, including ESB and ETL, lack support for JavaScript Object Notation (JSON). JSON is a lightweight data interchange format that continues to emerge as the common approach that will allow data integration technology to handle tabular, unstructured, and hierarchical data at the same time. As we progress, the role of JSON will become even more strategic to emerging data integration approaches.

Creating a Data Integration Strategy

The best way to approach data integration around the use of a data lake is to take a stepwise approach. This means looking at all aspects of your business and technology requirements, and focus on what data will be stored in the data lake, what the data means, and how it will be leveraged by data analyses and transactional processes.

To make this more understandable, we've defined 5 major steps:

Step 1: Define business requirements

List all of the business reasons for building a data lake, and place dollar amounts around the value to be generated. For example:

> ***Enhanced inventory depletion management using predictive analytics that will generate $10-15 million dollars in additional gross revenue in 2017.***

This is where you make the business case for the data lake, as well as all of the surrounding technology (data integration). It should list all values to be generated by the data lake, and the value of removing inefficiencies as well.

Step 2: Define macro-use cases

What are the higher level tasks that you'll use the data lake for, and what value will those tasks generate? Don't focus too much on the tactical issues at this step, such as specific systems and data sets, but the higher-level patterns of use. You can decompose those higher-level tasks down later when you define the data lake in more detail.

Step 3: Define the data attributes, structured and unstructured

Define the data as best you can, but not to the point where you create a structure. For instance, within the invoice PDFs, what data is relevant to analytics? Much of the data will be self-described as existing or having existed within a database. However, considering what we saw earlier in Figure 1, most of the data will exist in raw states, and therefore there will have no native structure. The idea within this step is to define a temporary template for the data. We'll use this template to create temporary structures that will be leveraged when the data accessed. These structures can be changed as our needs change in how we want to view the data.see Figure 4 (temporary structure)).

Typical Big Data System Architecture

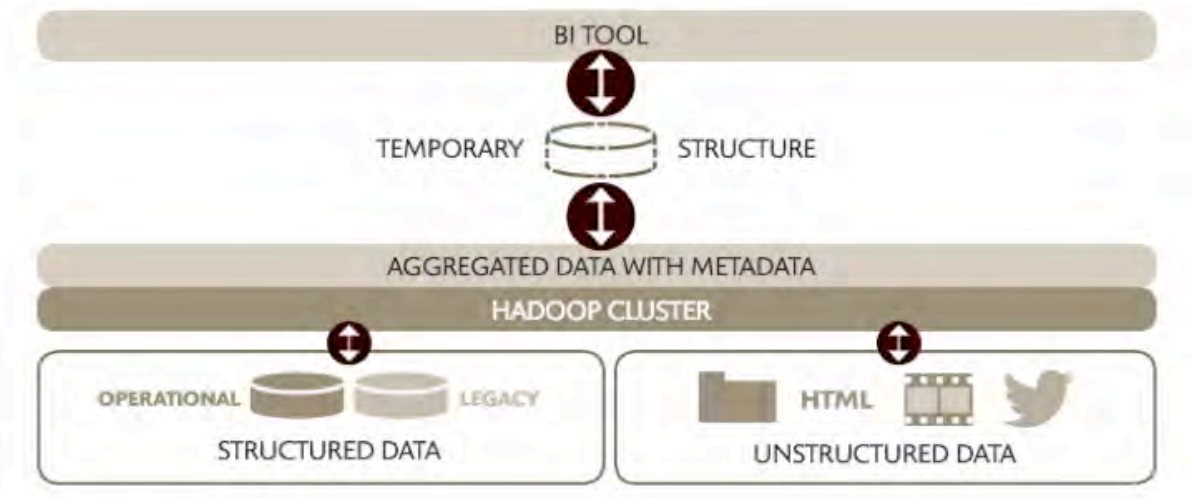


Figure 4: When defining data within a data lake, and defining data integration as well, there is no requirement to force a structure.

This approach flies in the face of traditional database design tactics, where structure is everything, and even traditional data integration tactics, where you need to force a structure for the data as well. There is no need to force a structure when leveraging a data lake. Indeed, it can be defined by use, as seen in Figure 4. This may be where you create a metadata layer that can provide definition around each data attribute.

Step 4: Define security, governance, performance, and other meta requirements

Define all of the technology and approaches needed to insure the data lake is secure, monitored, tracked, and can enforce policies that will define limits on use. Consider what security approaches and technology is correct, as well as data governance. Other concepts that come into play at this step include performance, operations, integration with DevOps, and other items that may be related exclusively to your problem domain.

Step 5: Define technology requirements, prepare to move to selection and implementation

Define the data integration pattern that meets the requirements that emerged in the previous steps. It's unnecessary to pick the technology in this step, but look at the patterns that will narrow down the number of tools to a manageable few. Those who cast a wide net in this step may find that there are too many tools to evaluate, and that it's obvious that most of the tools won't be a fit. For instance, ETL and ESB, as described above.

Technology Selection

In many instances, new data integration approaches are the best choice. They are purpose-built to take advantage of concepts such as late binding, and the ability to declare schema(s) when reading the source data vs. having schemas defined prior to the read.

New data integration approaches can work around the issues of dealing with large amounts of data that, these days, does not support a native hierarchical structure, including most unstructured data. What's more, modern technologies that handle this type of data to support applications are accustomed to declaring a structure upon access, and not having databases that are dependent upon a structure.

Other items that are typically desirable when selecting data integration technology for your data lake includes

1. Simple and Complex Orchestration Use Cases
2. Pre-Built Connectors
3. Java SDK
4. Cloud Connector Development Kit
5. Ease of Use Allowing Anyone to Leverage the Tool

6. Advanced Monitoring and Operations Support
7. Empower the “Citizen Developer”
8. The Ability to Support Mobile Devices
9. The Ability to Support IoT Devices and Sensors
10. REST-based Interfaces
11. Parity Between On-premises and Cloud Deployments
12. Elastic Scale Out Architecture
13. Multi-tenancy
14. Native Support for Hierarchical and Relational Data (JSON)
15. Event-based or Real-time Integration
16. Big Data Integration Using Data Lake Technology
17. High Availability
18. Standards-based Security
19. End-to-end Audit Trails
20. Geo-redundancy

You can use the 20 points as the foundation of your own requirements. However, make sure to consider your own use of the data lake concept, and have a good understanding of everything we’ve asked you to do in steps 1 through 5 above.

This process can do a few things to help insure success. First, you should understand your data lake requirements with enough detail to select the right approach, architecture, and correct data integration technology to support the data lake. Second, you can narrow the field of technology under consideration, thus compressing the time it will take to get from the need to a working data lake with a sound data integration approach and technology bound to it.

Call to Action

There are compelling reasons to use a data lake, as we’ve defined in this paper. However, the call to action is that you consider the specific value that a data lake will bring to your enterprise, and look specifically at the value concept we’ve presented in this paper.

The action is to take the time to define what this technology means to your organization. In many cases, it could be a competitive advantage that raises the business to new levels, with the new ability to see all of the data. We’re able to use that data to predict the future, as well as understand what’s going wrong in the business right now, and how to fix it. Finally, we’ll have the ability to automate much of the this process by binding the intelligence that we can abstract from the data with live business processes that can automatically change the course of the business using almost perfect information.

Summary

Key items presented in this paper include:

- Data lakes bring new value, as well as new complexity to enterprise data. While there is risk in implementing any new technology, including a data lake, this seems like a good idea most of the time, considering the upside benefit.
- Existing approaches to data integration, including ESBs, ETL, and other older approaches and technology, do not address the core data integration needs of a data lake. While many have based their data integration approaches and technology on these legacy technologies, their use is coming to a quick end.
- Data lakes' all-inclusive approach to data means that data integration needs technology to adapt to the growing complexity without requiring highly skilled IT talent. We need technology that is easy to use by the rank and file, and does not require that "data scientists" be involved with building and changing data integration for a data lake.
- Data integration planning should become systemic to planning a data lake, or any other big data solution. As we've seen in the steps presented above, there is much to think about in the planning stages of a data lake, and defining the proper data integration solution is a key to success.
- Make sure to consider data governance and data security with your data lake and linked data integration solution. See the checklists above, as to that you need to consider when building data integration solutions for a data lake.
- Picking the right technology is critical. You need to find technology that both meets your requirements, and provides the core solutions patterns you need, as well as tactical capabilities that are relevant to leveraging a data lake.

About the Author

David S. Linthicum is an internationally recognized industry expert and thought leader in the world of cloud computing and the author or co-author of 15 books on computing, including the bestselling Enterprise Application Integration, and his latest book, Cloud Computing and SOA Convergence. He is a blogger for InfoWorld, Intelligent Enterprise, eBizq.net, and Forbes, and he conducts his own podcast, the Cloud Computing Podcast. His industry experience includes tenure as the CTO and CEO of several successful software companies, and upper-level management positions in Fortune 100 companies. In addition, Linthicum was an associate professor of computer science for eight years and continues to lecture at major technical colleges and universities.

About SnapLogic

SnapLogic is the industry's first unified data and application integration platform as a service (iPaaS). The SnapLogic Elastic Integration Platform enables enterprises to connect to any source, at any speed, anywhere — whether on premises, in the cloud or in hybrid environments. The easy-to-use platform empowers self-service integrators, eliminates information silos, and provides a smooth onramp to big data. Founded by data industry veteran Gaurav Dhillon and backed by investors including Andreessen Horowitz, Ignition Partners, Microsoft Corp., and Silver Lake Waterman, SnapLogic is helping companies across the Global 2000 to connect faster. Learn more at www.snaplogic.com.