Cribl

WHITE PAPER

The Observability Pipeline Buyer's Guide

Should You Build Your Own Observability Pipeline?

WHITE PAPER

The Observability Pipeline Buyer's Guide

Executive Summary

When looking for an observability pipeline, there are many things to consider before making a decision. Some will argue for open source solutions while others have solutions that are heavily tied to their existing vendors. While all solutions have their strengths and weaknesses, it's important to consider the short term and long term benefits of each type of solution.

Open source solutions might be low cost in the beginning, but they end up costly in the long run as they often require custom development for future integrations. Solutions tied into your existing vendors are not aimed to lower your costs. In fact, their goal is to create vendor lock-in.

Vendor agnostic solutions allow you to control where your data lives, allow you to reduce volume to control costs, and shorten the time to derive key insights from your data.

There are some key ideas to consider when looking at an observability pipeline. We've put them together in a handy checklist. As you research solutions, keep these in mind.

- *Protocol support*
- *Management system*
- *Supports both cloud and local instances*
- *Supports many destinations while having flexibility to add new ones*
- *Designed to reduce data costs*

VENDOR AGNOSTIC SOLUTIONS ALLOW YOU TO CONTROL WHERE YOUR DATA LIVES, ALLOW YOU TO REDUCE VOLUME TO CONTROL COSTS, AND SHORTEN THE TIME TO DERIVE KEY INSIGHTS FROM YOUR DATA.

54% OF ORGANIZATIONS ARE MOVING TOWARD A HYBRID IT ENVIRONMENT.

It’s said that you can only count on two things in this world: death and taxes. There’s a third to add to the list: the increasing volume of log data a company needs to analyze and store.

Many organizations adopt a host of new technologies and become more software-centric to keep up with business demand. Now more than ever, there’s an expectation of businesses to push new software, applications, services, features, and capabilities quickly to serve their customers better and keep up with the competition.

While more businesses embrace microservices architectures, some are going the container route like Kubernetes, and others are going serverless. Because each organization’s approach can vary, cloud-native technologies ultimately drive complexity.

In addition to those technologies, companies are still managing traditional systems as well. According to 451 Research, 54% of organizations move toward a hybrid IT environment that leverages both on-premises systems and off-premises cloud/hosted resources in an integrated fashion.

Each component of a complex, dynamic, distributed environment generates a set of operations data; data volumes are increasing rapidly. Implementing an observability pipeline can help organizations get increased visibility into their data.

Before we dive deep into an observability pipeline, we need to define observability. Observability is being able to answer the question of what’s happening in your environment without knowing what to look for ahead of time. Another way to look at it is to answer “how do I get the data out of my infrastructure and applications to observe, monitor properly, and secure their running state while minimizing overlap, wasted resources, and cost?”

In an observability pipeline, you are able to receive data from any source, collect from object storage, transform into any shape, and route it to any destination.

In all environments, there’s generally a need for a broad category of tools to solve these challenges: up/down monitoring, metrics or time-series databases (TSDB), log analytics, event streaming, security information and event management, user behavior analytics, and data lakes. We believe that many of these categories, such as monitoring, log analytics, SIEM, UBA, and others, fall under the broader category of observability.

In this Buyer’s Guide, we’re going to explore alternative build strategies for implementing an observability pipeline on top of other popular open-source projects like Fluentd, Logstash, or Apache NiFi, and compare that to an out-of-the-box approach. As you consider your organization’s needs, consider which course will provide a greater return on investment.

Building Your Own

There is much to be said for custom-engineering your own solution to this problem. The hope is it will be tightly integrated into your workflow. It will create career opportunities for engineers. You might think it'll be precisely what you need and nothing more. Since it'll be custom-built, there will be no rethinking of your internal workflows or policies.

If you're building your own observability pipeline, you will need several properties: protocol support for existing agents, an easily manageable system, and performance. Some people consider building their own observability pipeline on top of several open-source projects like Apache NiFi, Fluentd, or Logstash. Some have been successful at this, but for all of them, it's a costly journey that can involve major architectural shifts and building many things on top of the base open-source options.

We'll dig into some of the challenges below.

PROTOCOL SUPPORT

The first and most significant struggle with implementing an observability pipeline on your own is protocol support. Most organizations have anywhere from ten to hundreds of thousands of deployed agents already. Suppose your agents are proprietary, like the Splunk Universal Forwarder. In that case, you're looking at step 1 being a significant uplift to replace all your existing agents or, at a minimum, install a second agent collecting the same data.

Suppose your agents are already open source agents like Elasticsearch's Beats or Fluentd. In that case, you can probably implement Apache Kafka as an interim receiver to pre-preprocess before sending it onto Elastic. However, you're now adding a new stateful distributed system to your pipeline, which still does not solve the end-to-end use case. To solve the real problem, including transforming and processing the data, your pipeline isn't going to be simple. It's going to include multiple processing steps, with additional processing software like Fluentd pulling data off of topics and a second copy on a different topic.

PROGRAMMING YOUR PIPELINE

The second major challenge we see with building an observability pipeline on top of a generic stream processing engine is how much work is left to the administrator. Systems like Apache NiFi are very flexible. You can use them to transport even binary data from the edge. You can use them as a passthrough, routing arbitrary bytes from one Kafka topic to another. From the Apache NiFi documentation, "Apache NiFi is a dataflow system based on the concepts of flow-based programming." NiFi is a graphical environment, but most generic stream processing engines provide a programming environment to their end-users. Programming a stream processing engine requires end-users to work at a much lower level. Extreme flexibility comes at the cost of requiring the system's user to reimplement many things that are done out of the box in logging systems.

BUILDING YOUR OWN MANAGEMENT

When thinking of implementing a system, most projects and products tend to focus on the day one problem: “how does the user get this up and running?” However, over the lifetime of the solution, day two questions like the following are left to the system’s implementer:

- *How does the user make changes safely and reliably?*
- *How does the user know if the system is performing correctly?*
- *How does the user troubleshoot this system?*

Log processing tools like Fluentd and Logstash are okay on day 1. Downloading them on your laptop, throwing some data through, and building an initial configuration is pretty easy to do, but is it scalable and long-lasting?

Managing configurations and configuration changes require a significant investment on top of your base systems like Fluentd or Logstash. Organizations building their own observability pipelines successfully have created their own unit testing frameworks for configurations and intense code and configuration review processes. These processes weren’t invented for no reason. Before implementing these processes, minor configuration errors would regularly break production. This process also involves implementing your own continuous deployment pipeline to roll changes safely to production.

On top of CI/CD, which needs to be built, monitoring comes next. Suppose data is coming in delayed or the system slows down for any reason. In that case, metrics need to be available to determine which system is causing back pressure or which particular set of configurations might be the culprit. One wrong regular expression can destroy the performance of a data processing system. Building up rich monitoring dashboards are a considerable part of the Day 2 cost of building your own system.

Lastly, when data is not being processed in how the end-users are expecting, the operators of the system need to be able to reproduce data processing so they can validate why a given configuration produces a given output. There is no environment like production. Attempting to take a configuration and an environment where the given conditions reproduce the problem can be hugely time-consuming. Tools like Fluentd and Logstash provide little introspection, data capture, or troubleshooting capabilities.

PERFORMANCE

When moving data at multi-terabyte and multi-petabyte daily volumes, a 20% improvement in processing speed can impact the tens to hundreds of thousands of dollars a year in infrastructure costs. Fluentd and Logstash have had long-known performance challenges. Generic systems like Apache NiFi are not designed for petabyte/day scale, and building your own system can be very costly from an infrastructure perspective at scale.

In logging systems like Splunk, Elasticsearch, etc., there is a fair amount of work handled for you by combining the shipper and the storage tier, like a Splunk Forwarder or Elasticsearch Beats, and Elasticsearch or a Splunk Indexer. The shipper picks up a byte stream off of disk, and it may do lightweight parsing on the endpoint to break that byte stream up into events, or it may ship raw bytes off and do event breaking at the centralized tier. Either way, from the user’s perspective, an event’s concept is the base unit of work. If you’re trying to put something like Apache NiFi in the middle of your ingestion pipeline, it is incumbent upon the developer to do things like event breaking. This situation is possible, but it adds a lot of work. To build your own observability pipeline on NiFi or a similar system, you will need developers to implement event breaking for you, which can put a strain on your team's resources.

An Alternative: The Out-of-the-Box Solution

LEVERAGING AN OUT-OF-THE-BOX OBSERVABILITY PIPELINE LIKE LOGSTREAM MAKES IT EASY FOR TEAMS TO CHOOSE THEIR DATA MANAGEMENT STRATEGY, IMPLEMENT IT QUICKLY, AND REDUCE INFRASTRUCTURE COSTS.

As outlined above, building your own observability pipeline comes with some challenges, including insufficient protocol support, increased workload for the administrator, difficulty troubleshooting and managing the configuration(s), and rising infrastructure costs. An out-of-the-box observability solution can help relieve these stressors. An out-of-the-box solution like Cribl LogStream is enterprise-ready with a turn-key deployment and easy manageability.

Another feature to consider is role based access control (RBAC). Use RBAC to manage teams' access to data, so they can only receive what they need to do their job. RBAC helps protect sensitive data and can also streamline how different categories of data are allocated.

An observability pipeline can provide configuration validation within the product, complete with built-in distributed management and monitoring. These solutions also typically offer rich data capture capabilities to help teams troubleshoot data flow in production, relieving protocol support pain points.

A no-code configurable solution that works natively on events can help too. Because most sources and destinations also deal with events, an observability pipeline enables organizations to handle event breaking natively and intuitively directly in their observability tool when sent a raw byte stream.

With a tool like LogStream, a developer skill set isn't required to work with observability data, which eases administrator workload and allows you to troubleshoot and manage your configuration(s) much more straightforwardly. Tools should work for you out of the box. New updates should just happen.

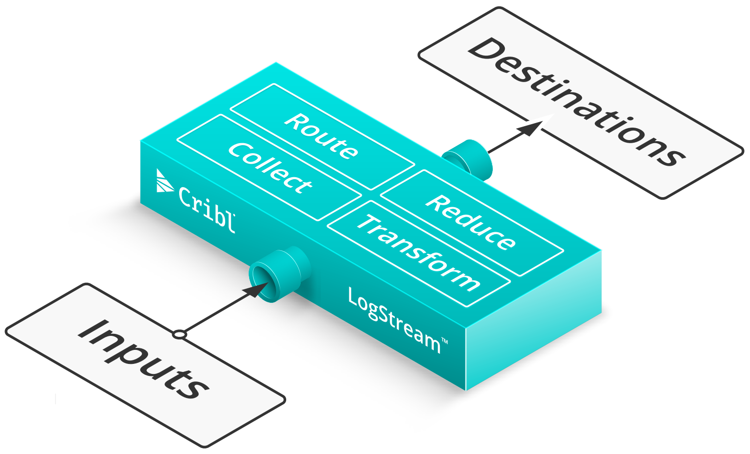
Additionally, an out-of-the-box observability pipeline can help significantly cut infrastructure costs while increasing performance. A good observability pipeline should give you the ability to choose your business's best data management strategy. Depending on your business goals, that can mean implementing several tactics, including but not limited to:

- *Sending data that benefits from needle in a haystack search performance to an indexed analytics system*
- *Routing metrics data to a time-series database for fast dashboarding and initial investigations*
- *Placing raw data in cheap storage like S3 for optimized retrieval of subsets of that raw data*

Leveraging an out-of-the-box observability pipeline like LogStream makes it easy for teams to choose their data management strategy, implement it quickly, and reduce infrastructure costs. Using your observability tool, you can decide which data you need to send to an analytics tool to analyze now, which logs can be aggregated into metrics, which data should be stored and analyzed later if required, and which elements of data should be dropped altogether. The combination of cheap storage and increased performance presents a significant return on investment.

Vendor Lock In: Avoid at All Costs

With an out-of-the-box solution like LogStream, a vendor is working to lower your data costs while easing access burden. Existing vendor solutions are designed to keep you locked into their platform, where a dedicated observability pipeline vendor is working to give you the most flexible access with the most destinations.



Key questions to ask yourself before choosing a vendor or solution type is do you want to future-proof your ingestion pipeline and be able to deliver data anywhere? Do you want to be able to deliver data even to potentially competitor products with your existing logging system?

When you choose to use a "feature" of your existing logging system as an observability pipeline, you'll be locked into that vendor for everything in the future and whatever price increases come with that. It's likely that the compute infrastructure costs of your existing vendor will cost more than an out-of-the-box solution.

IT'S CRUCIAL TO CONSIDER MANY FACTORS WHEN BUILDING AN OBSERVABILITY PIPELINE, INCLUDING PROTOCOL SUPPORT FOR EXISTING AGENTS, AN EASILY MANAGEABLE SYSTEM, AND PERFORMANCE.

Final Thoughts

An observability pipeline is a new concept that is solving key problems that organizations face today. We have seen some successful custom-built implementations, but more often solving a business problem requires a full out-of-box solution. Most existing custom solutions require a lot of integration and building effort on the part of their users to really address all of their challenges. Custom solutions will require continual investment as well. The upfront fee is only building the foundation. As new destinations are needed, new investment will have to be made. If your budget doesn't allow for those new investments, you now have a product that is expensive, doesn't meet your needs, and now you're locked into using.

Organizations that are going down the path of building their own solution should prepare for a considerable investment of time, talent, and money. Most organizations are investing in 2-5 resources from 3-12 months, depending on the needs of their requirements. A build option starts at around $100,000 and up to $1 million or more for the first year. In addition to the initial build, a continual investment of a scrum of 1-3 developers to write transformation code puts the sustained maintenance in the hundreds of thousands a year.

Building a solution can satisfy your requirements in the short term, but is the investment into undifferentiated infrastructure and processes worth it? Not only can an out-of-the-box observability solution like LogStream meet your organization's needs, but it can also scale with you, giving your team the freedom to decide what the organization's data management strategy looks like moving forward. An out-of-the-box solution is prepared for your organization's needs today, but also growing to meet tomorrow's needs.

As you weigh building your own observability pipeline on top of an open-source project, it's crucial to consider many factors, including protocol support for existing agents, an easily manageable system, and performance. While building your own pipeline can often meet your organization's exact needs now, they may do it inefficiently, straining your team's resources and increasing costs. On the other hand, an out-of-the-box observability solution like LogStream requires an investment but can result in better performance, reduced infrastructure costs, and the ability to shift your data management approach as your business challenges evolve. Cribl's LogStream, with its out-of-the-box approach, you'll have a scalable solution that provides a near immediate return on investment.

FINAL POINTS

- *Do you want to future-proof your ingestion pipeline and be able to deliver data anywhere?*
- *Would you like to affordably retain years of data while lowering your costs?*