

Simplifying Apache[®] Spark[™] Operations with Databricks

Background

Apache Spark™ is being widely adopted as the general processing engine by organizations of all sizes for processing large-scale data workloads. To effectively run mission-critical production workloads on Spark, data teams spend significant effort and time creating and maintaining a complex technology stack that can:

- Deploy secure, highly available, and multi-tenant Spark clusters.
- Support developers, data engineers, and data scientists to explore data and build Spark applications.
- Put Spark applications in production and monitor their execution.

This whitepaper shows how Databricks enables developers, data engineering, and data science teams to focus on getting value out of their data instead of being bogged down with operational details by providing three sets of critical functionalities out-of-the-box for its users:

- Managing infrastructure
- Exploring data, writing code, and debugging applications
- Productionizing jobs



Managing Infrastructure

Building and deploying Spark clusters involve more than just data engineering and data sciences. A critical part of your project involves having to build up the infrastructure to operationalize your Spark deployment. This starts with a foundation of **containers or virtual machines (VMs)** allowing you to package your application and associated binaries and libraries to be easily patched, replicated, and distributed. It is also important to implement the necessary **security** mechanisms to ensure that the right users (authenticated) can access the right data (authorized). Production Spark deployments must ensure **high availability** and **multi-tenancy** so there is uninterrupted access to the data and its insights by its many concurrent users.

Infrastructure Management with Databricks

COMPONENT	CURRENT CHALLENGES	DATABRICKS SOLUTION
Container / Virtual Machines Maintenance	One of the fundamental building blocks for a Spark deployment is the need to build up the infrastructure to deploy your Spark development, test, and production environments. To do this, every developer, data engineer or data scientist will need to build up the DevOps knowledge and processes required to build up containers or virtual machines. These containers or VMs will require operational processes to patch the latest operating system and software updates and versions (e.g., latest Java / JVM, Python updates, Spark versions, Spark and R packages, etc.).	Automatically updates clusters and all of its underlying containers and virtual machines so you do not have to do this manually.
Security	The next fundamental building block is to establish the necessary security mechanisms to authenticate and authorize users. When implementing these required mechanisms, some potential issues that often arise range from building up the necessary virtual private clouds (VPCs), identity management, enterprise-grade encryption, and auditing. Fundamentally it is important to build the right mechanisms so that authenticated users are accessing only the data, metadata, and notebooks that they are authorized to access.	Built on industry-leading infrastructure, designed with best-in-class security features, and rigorously audited, Databricks connects to the data in your storage infrastructure while maintaining the highest standard of security. For more information, please refer to the Databricks Security Primer .

Infrastructure Management with Databricks (continued)

COMPONENT	CURRENT CHALLENGES	DATABRICKS SOLUTION
High Availability	Although Spark is highly distributed in the way it processes data, it employs a master / worker architecture where the master controls the scheduling workflow across the workers, which in turn executes the jobs - creating a single point of failure. If that master goes down, so does your service. To ensure continuous and uninterrupted service, high availability must be built into the systems. In most production environments, this will entail building multiple Spark masters backed by Zookeeper infrastructure with N workers, consuming dedicated resources. For Spark Streaming, persistent storage is necessary in addition to Zookeeper.	Databricks ensures your service is always up and running without the need to manage it yourself. If a worker instance is revoked or crashes, the Databricks cluster manager will relaunch it — transparent to the user.
Multi-tenant Cluster Management	As most Spark cluster deployments involve multiple concurrent users, it is important to build up security roles (e.g., IAM roles, access keys, etc.) so that users have the right authorization and authentication policies implemented to run their workloads. It will be important to allow for multiple notebooks and multiple users to attach to a single cluster; to set up an iPython-like environment that easily allows users to connect their notebooks to different clusters (with different Spark versions) while maintaining the infrastructure for the notebook layer.	A user-friendly, easy-to-use user interface simplifying the creation, restarting, and termination of multi-tenant Spark clusters.

Exploring Data, Writing Code, and Debugging Applications

Often developer, data engineering, and data science teams operate in siloed workspaces, making it very difficult to work efficiently and collaboratively. Hosted **notebooks** let them do so in an environment that is physically close to the data. But today, most notebooks do not have **version control** capabilities. This is particularly limiting when collaborating with others, auditing changes, going back and forth between versions, or working on multiple versions in parallel.

To diagnose bugs, developers and data scientists need convenient ways to understand the tasks and jobs being executed by the Apache Spark queries via the Spark UI. However, the Spark UI lacks interactivity and is therefore difficult to use during the development process. Moreover, once a Spark cluster is shut down, the Spark UI history is lost, which further complicates debugging.

Some developers augment their use of notebooks with advanced libraries that they develop using **IDEs**, such as IntelliJ or Eclipse. Packaging and uploading different iterations of their compiled libraries close to the data are cumbersome.

Databricks' notebooks environment supports collaboration and version control

First, we need to featurize the Tweet text. ~~Here is that function below:~~

```
'''scala
object Utils {
  ...MLlib has a HashingTF class that does that:

  '''scala
  object Utils {
    ...

    val numFeatures = 1000
    val tf = new HashingTF(numFeatures)

    /**
     * Create feature vectors by turning each tweet into bigrams of
     * characters (an n-gram model) and then hashing those to a
     * length-1000 feature vector that we can pass to MLlib.
     * This is a common way to decrease the number of features in a
     * model while still getting excellent accuracy (otherwise every
     * pair of Unicode characters would potentially be a feature).
     */
    def featurize(s: String): Vector = {
      val n = 1000
      val result = new Array[Double](n)
      val bigrams = s.sliding(2).toArray
      for (h <- bigrams.map(_._hashCode % n)) {
        result(h) += 1.0 / bigrams.length
      }
      Vectors.sparse(n, result.zipWithIndex.filter(_._1 != 0)
    }.map(_._swap).tf.transform(s.sliding(2).toSeq)
    }
  }
```

July 13, 2:57 PM
● Administrator

July 13, 2:51 PM
● Administrator

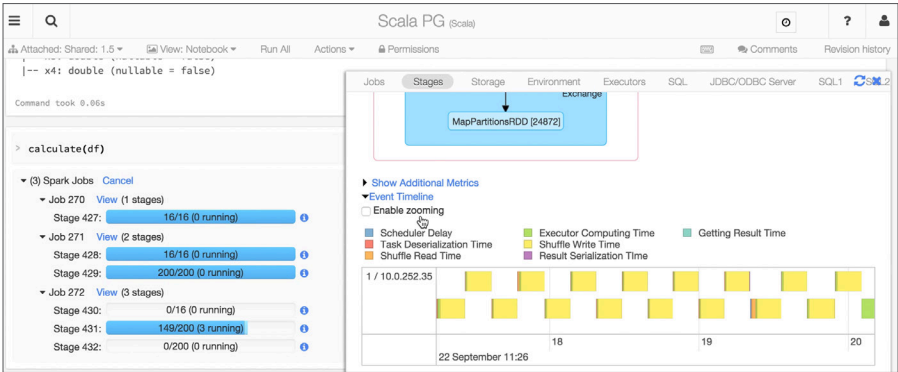
May 7, 5:44 PM
● Commit a8bb2be504
fixed twitter typos

September 23, 10:57 AM
● Commit ffb5d09537
Update the docs to reflect the HashingTF featurize
[Restore this revision](#)

September 5, 11:58 AM
● Commit b8b422a265
Better way to save tweets

September 4, 7:06 PM
● Commit d64963e7d0
Add the Twitter Streaming Langua...

Data Exploration and Building Apache Spark Applications with Databricks

COMPONENT	CURRENT CHALLENGES	DATABRICKS SOLUTION
Notebook Environment with Spark UI Integration	<p>A common solution to provide the interactivity for developers and data scientists is to build a notebook environment such as iPython and add custom functionality. From an operations perspective, this entails deploying a notebook UI service and a Spark History Server requiring the building and maintenance of even more infrastructure (e.g., web servers, databases, Python versions, JVM versions, Spark compatibilities, patching, etc.).</p>	<p>Progress Bars are included within the Databricks notebook so that a user can understand a Spark Job's execution progress in real time. This allows them to review Spark job progress directly next to the code that executed those jobs. For more information, refer to Easier Spark Code Debugging: Real-Time Progress Bar and Spark UI Integration in Databricks.</p>
<div><div></div><div><i>Spark UI Integration in Databricks</i></div></div>		
Source Version Control	<p>Most developers use source version control tools such as Git. Since most notebooks do not have version control capabilities, the developers would have to work with source version control manually.</p>	<p>GitHub integration is included in Databricks notebooks so data practitioners are able to manage and version notebooks in the same way as they write code.</p>
IDE Integration	<p>Many developers write Spark libraries in their preferred IDE and want to use these libraries in a notebook environment. Incorporating custom libraries in a custom notebook environment is a tedious manual process that slows down the development cycle.</p>	<p>Ability to use the SBT build tool to automatically compile, upload, and install libraries that are being developed, allowing users to quickly iterate on libraries that they are developing and Databricks notebooks that access those libraries. The SBT plugin can be used from most existing IDEs, such as IntelliJ. As noted in the blog post Making Databricks Better for Developers: IDE Integration, developers and data scientists can upload the libraries written in their favorite IDE to Databricks with a single click.</p>

Productionizing Jobs

In traditional environments, data scientists build models within a development environment. The completed models are then handed off and re-written to run in a production environment. The rewrite creates potential for bugs and prolongs the transition between prototyping and production. Organizations also have different trade-offs in the performance and cost of running jobs, so they need ways to customize the jobs configurations in ways that suit their unique needs. Once a job is in production, users want to be notified if a job fails, succeeds, or runs to detect problems early and minimize downtime.

Running Production Jobs with Databricks

COMPONENT	CURRENT CHALLENGES	DATABRICKS SOLUTION
Production Job Scheduler	To put Spark code in production, a scheduler is needed by writing custom scripts and integrating them with existing workflow managers, such as Luigi or Oozie. Furthermore, such schedulers run standalone jobs; they don't easily run notebooks to interactively develop code, debug, and investigate data.	Databricks lets you launch or use any specified Spark version, supports standalone jobs, JARs, as well as running notebooks as jobs. Users can seamlessly transition between interactive exploration and production —allowing them to iterate and improve without spending time to rewrite and move code between different systems.

☰

🔍

?

👤

[All Jobs](#)

Salesforce Leads Job

Task: Notebook at `/summit-demos/Salesforce Leads with Machine Learning, Spark SQL, and UDFs` - [Change](#) / [Remove](#)
 Libraries: [Add](#)

Cluster: 90 GB Spot, Spark 1.5 [Edit](#)
Schedule: Every day at 2:00am (US/Pacific) [Edit](#) / [Remove](#)
[Advanced ▶](#)

Active runs

Run	Start Time	Launched	Duration	Status
No active runs. Run Now				

Completed runs

[Latest successful run \(refreshes automatically\)](#) - [View as a dashboard](#)

[Previous 20](#)

Run	Start Time	Launched	Duration	Status
Run 4	2015-10-15 02:00:00	By scheduler	7m 8s	Succeeded
Run 3	2015-10-14 02:00:00	By scheduler	9m 0s	Succeeded
Run 2	2015-10-13 02:00:00	By scheduler	6m 42s	Succeeded
Run 1	2015-10-12 13:39:56	Manually	1m 40s	Succeeded

[Next 20](#)

“Salesforce Leads” notebook is now part of a Salesforce Leads job that is executed on a 90GB spot cluster executed every day at 2 am PT

Running Production Jobs with Databricks (continued)

COMPONENT	CURRENT CHALLENGES	DATABRICKS SOLUTION
Notification Support	Monitoring the progress of Spark jobs requires significant effort to build custom scripts and integrate tools for the most basic features such as being alerted to a Spark job failure. Without automatic notification, monitoring production jobs is time-consuming and requires manual intervention.	Production jobs on Databricks can be configured to send an email to a set of users whenever they complete or fail, allowing important jobs to run without manual intervention.

Email Alerts

Enter addresses to email when a run starts, succeeds, or encounters an error.

On start

Emails (comma-separated)

On success

Emails (comma-separated)

On error

Emails (comma-separated)

Cancel

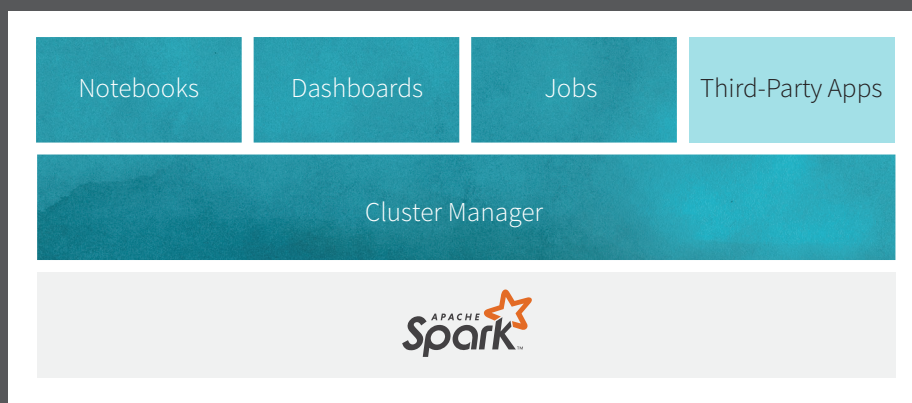
OK

Notification support

For more information, refer to [The Easiest Way to Run Spark Jobs](#).

Conclusion

With Apache Spark, data practitioners have a fast and powerful execution engine for processing large-scale data workloads. Yet, there are many operational processes that must be deployed and maintained instead of the focus being on making sense of the data. Tasks ranging from deploying containers and VMs to securing the environment are fundamental to your infrastructure. Building out high availability and multi-tenant clusters allow you to have multi-user, concurrent and uninterrupted access to your data and analysis. Providing easy-to-use interfaces allow data practitioners to code and debug in their language of choice. Easy integration with GitHub and popular IDEs allow you to manage, version control, and review the code and models being created. With the Databricks Jobs infrastructure, that same notebook can easily be deployed as a production job with minimal configuration and support.



Databricks incorporates many features that handle the operational tasks of your Spark deployments with minimal guidance from your data engineers and data scientists. By simplifying the operational management of your Spark clusters, data teams get to focus on building data products instead of maintaining infrastructure and managing operations.

For more information on Databricks, please reference the [Databricks Product Page](#) or the [Databricks Feature Primer](#)