WEKA | aws

Sagemaker Reference Architecture

March 2023

Reference Architecture

Contents

Target Audience

This document is designed for a technical presales or engineering resources that would like to test or work with WEKA on AWS with Amazon's Sagemaker Local libraries in python 3. The idea of repeatable steps yielding predictable results is the key focus.

Executive Summary

Artificial intelligence's (AI) scope and breadth at this time have a trajectory toward consuming many of the commercial and enterprise software offerings that power operations, development, and our ability to identify risk given sufficiently well-curated data, scalable infrastructure, performance transparency, and lifecycle management of data.

Tagging key features in a photo or providing a barometer of customer satisfaction through sentiment analysis that was once considered dramatic has become table stakes for operational agility.

WEKA is uniquely positioned as a storage partner given the design patterns of the core architecture, the cloud-first but hybrid capable deployability, and native multi-cloud data mobility snap to object. In addition the ability to support multi-protocol access to all of the data, which eliminates the need to use multiple storage environments due to protocol requirements.

This paper will cover the different stages that are used including data validation and training using Sagemaker with WEKA as the data repository for the raw data that will be used to train the model.
Introduction

Amazon Web Services (AWS) provides a managed service called Sagemaker for all the lifecycle elements of building, deploying, and maintaining a wide variety of machine learning (AI/ML) models. The focus of the service is data first, providing data curation/hygiene tools, transport, and validation tools.

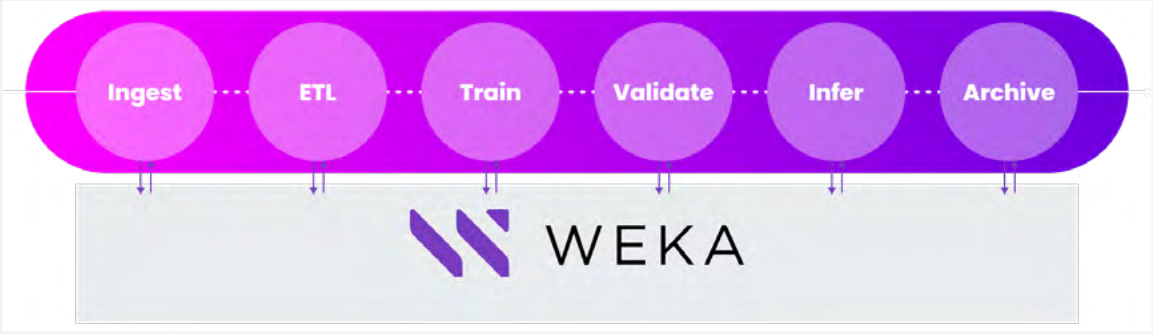
WEKA was founded on the idea that current storage solutions have only provided incremental improvements to legacy designs, allowing for a widening gap between compute performance and data storage performance. Storage remains a bottleneck to application performance, and with the continued densification of compute in areas such as GPU-based applications, has become even more problematic. In today's hyper-competitive market, organizations need flexible infrastructure; application workloads are becoming increasingly complex and data sets are continuing to grow unchecked, forcing enterprises to architect overly complicated and costly systems that reduce IT agility. As a result, important business insights remain locked away, out of reach of decision-makers.

IT organizations are adopting cloud technology for its fluid, on-demand scalability that supports diverse workloads at scale. However, while network and compute can be virtualized to operate at scale very effectively, storage remains largely isolated in silos based on system performance profiles. Consequently, organizations are forced to architect a storage system that is highly customized for their environment and workloads from building blocks that do not scale. The result is a storage solution that is complex, temperamental, expensive, and slow.

WEKA has built a software-only, high-performance file-based storage solution that is highly scalable and easy to deploy, configure, manage, and expand. The design philosophy behind the WEKA file system (WekaFS) was to create a single storage architecture that runs on-premises or in the public cloud with the performance of all-flash arrays, the simplicity and feature set of network-attached storage, and the scalability and economics of the cloud.

WEKA Advantages

WEKA has built a software-defined data platform that leverages cutting-edge cloud, compute, storage, and fast networking technologies to unleash the value of your data.



The WEKA® Data Platform delivers fast access to data when needed across the AI workflow. Our patented architecture scales to exabyte scale and eliminates the need for data copies, reducing operational complexity, enhancing pipeline efficiency, and increasing GPU utilization.

With WEKA, a single data platform supports all popular data access methods, including the POSIX file system, NFS, SMB, S3, and GPU Direct Storage (for direct data movement between GPUs and storage). In addition, because the platform is software-defined, it enables both on-prem and cloud deployment.

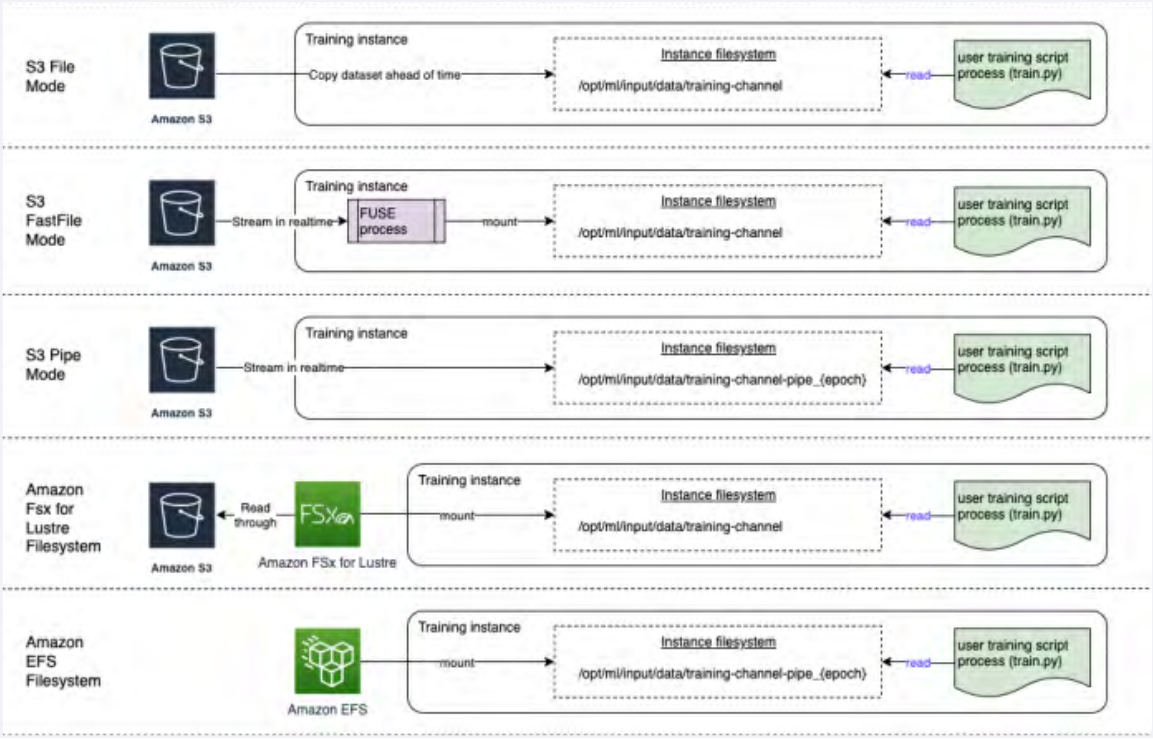
Additional benefits include:

- Move and back up data easily. Advanced data management capabilities simplify backup data to the cloud or move data quickly and efficiently between clouds to meet operational needs.
- Tier automatically. WEKA can automatically tier cold data to low-cost object storage (on-prem or cloud) for better economics. The differentiating value is that all of the data remains in the WEKA namespace, and metadata stays on the flash tier for fast access.
- Ensure security. The WEKA Data Platform was designed to ensure the security of your data with advanced authentication, in-flight and at-rest encryption, and flexible key management.

Read more in the WEKA Data Platform architecture white paper.

WEKA Sagemaker Solution Design

Amazon Sagemaker has a wide range of access approaches for ingesting data for processing.



Each is useful for different requirements, for our solution validation purposes, we will be using Amazon Sagemaker Local Mode configuration to facilitate the entire pipeline using the data on the WEKA system on the cloud. This configuration is closest to Amazon Fsx for Lustre Filesystem in the above diagram.

Environment configuration

The diagram below details the AWS environment that we used.

The WEKA system was composed of **i3en.2xlarge** instances

The client hosting the GPU (NVIDIA A10G) was an g5.12xlarge instance with a WEKA POSIX client deployed on the same VPC as the Weka cluster.

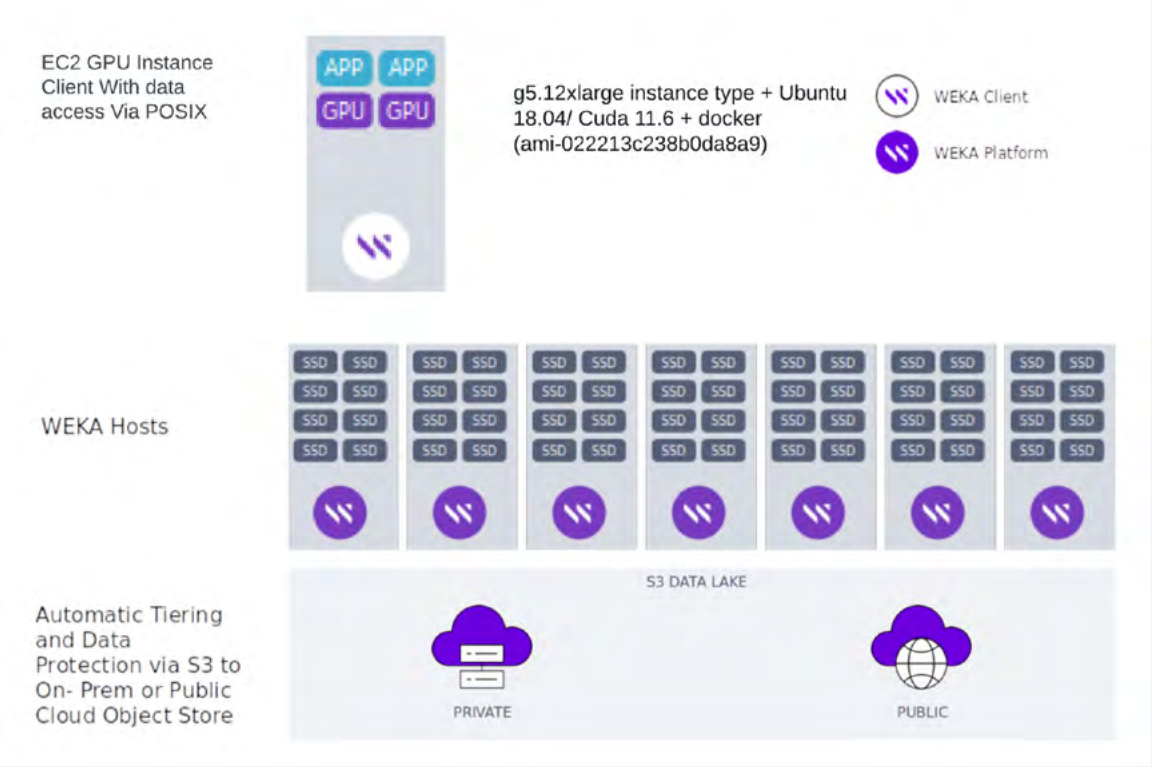


FIG. 1 WEKA AWS cluster and GPU instance used in the testing.

WEKA and S3 file with local NVMe storage Comparison

Test 1 – Perform similar steps when using Sagemaker with WEKA and S3 with Local NVMe

For the first test, our approach was to take an example python jupyter notebook which performs the following steps.

1. Download an augmented CIFAR10 dataset from a local web server
2. Perform an md5 hash operation to validate integrity on the downloaded tarball archive
3. Extract the dataset to a local filesystem which is placed on the local instance NVMe.
4. Perform an md5 hash operation to validate individual files containing training data. (1283 files)
5. Read the data from the pickled file format into memory
6. Read the metadata about the individual records to provide effective classification
7. Upload the uncompressed pickled records to an S3 standard bucket for Sagemaker distributed session management.
8. Read a random batch of 4 images from the data set
9. Train using the data set using a single local A10G Nvidia GPU on the provisioned instance

We then recorded how long each step took to execute for later comparison with the WEKA system.

This allowed us to compare all elements of data preparation and training for an Amazon Sagemaker job that uses S3 and a local filesystem, which is one of the default high-performance fashions Sagemaker recommends.

Test 2 – Remove the download step from the WEKA testing (redundant)

The second test shows how the pipeline looks like when we removed the download, extraction, and additional operation associated with moving data when running the test on the WEKA system. Since we analyzed the data in place on a WEKA filesystem without the need to download it to another storage tier this better simulated a real environment.

NOTE: This step is important since with WEKA being the filesystem it already contains the data that will be used for the training therefore there is no need to further download, also the WEKA will be used while residing on the WEKA filesystem, and not copied to a local NVMe. With the S3 plus local NVMe option even if the data is on the S3 bucket it still needs to be downloaded to the local NVMe of the instance in order to be processed.

This test allowed us to understand the time savings in a real pipeline for exploring WEKA resident data instead of copying from an S3 bucket or external web server.

This second test performs the same steps as the 1st test with the change of performing steps 1,2,3, only on the S3 with local NVMe testing.

1. Download an augmented CIFAR10 dataset from a local web server - S3 with local NVMe configuration only
2. Do an md5 hash operation to validate integrity on the downloaded tarball archive - S3 with local NVMe configuration only
3. Extract the dataset to Filesystem - performed only S3 with local NVMe
4. Do an md5 hash operation to validate individual files containing training data
5. Read data from the pickled data into memory
6. Read metadata about the individual records to provide effective classification
7. Upload the uncompressed pickled records to an S3 standard bucket for Sagemaker distributed session management
8. Read a random batch of 4 images from the data set
9. Train using the data set using a single local A10G Nvidia GPU on the provisioned instance

Between these two tests, the goal of this paper is to clearly show the functionality and capabilities of an Amazon Sagemaker Local Mode instance with WEKA storage, as we have seen them in our testing.

We will take key patterns from the Amazon Fsx for Lustre filesystems, where the primary copy of data resides

To provide differentiation against other vendors, a single client is attached to a minimum WEKA cluster.

The GPU instance that we used for the testing is

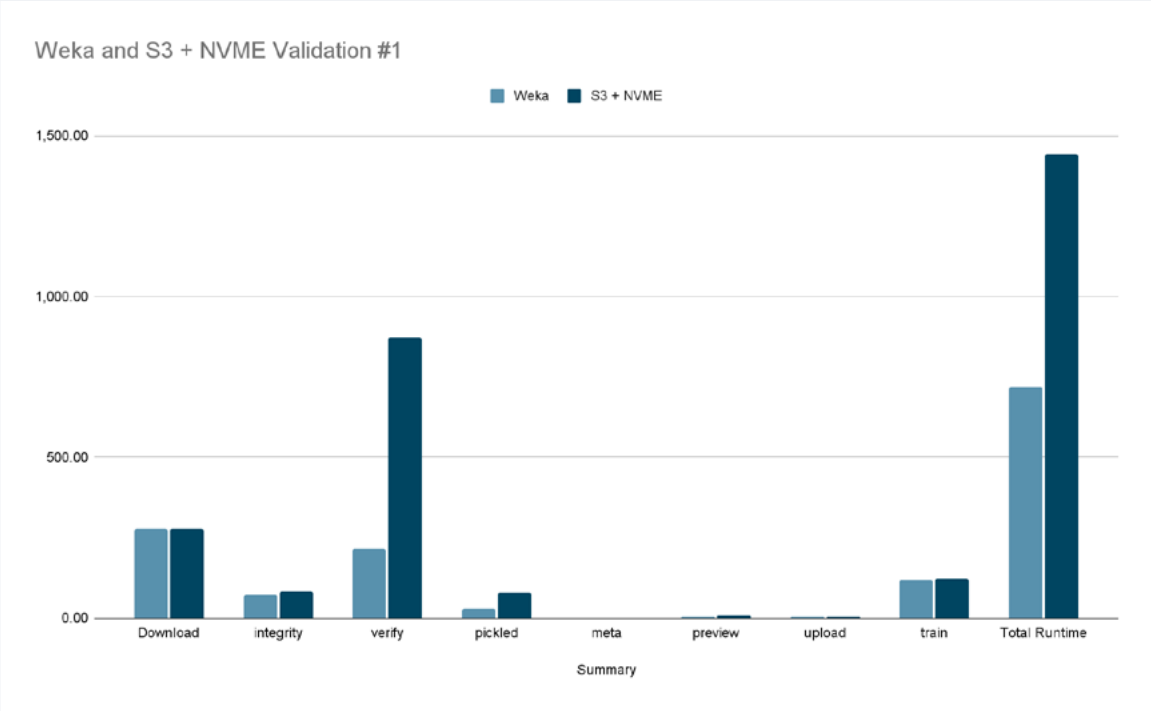
- EC2 Instance type g5.12xlarge
- OS - Ubuntu 18.04 9 (ami-022213c238b0da8a9)
- CUDA version - 11.6
- GPUs - 4 GPUs - only one is used for training
- Networking - 25GbE networking
- Local storage for S3 Sagemaker comparison - Three volumes for local storage to compare contrast what local NVMe technology performance looks like as a control element, a typical client in the industry standard cloud deployment.

Code for these experiments was developed from this repository here, the code is available here in a WEKA private repository.

Test #1 Results

The following chart shows how long multiple segments of a job took with the runtime in seconds, lower is better:

Comparing between WEKA and S3 + NVME



	WEKA	S3 + NVMe
Download	275.55	275.55
integrity	71.76	82.93
verify	214.04	872.40
pickled	28.08	77.93
meta	0.01	0.00
preview	4.59	6.82
upload	5.12	5.33
train	120.47	123.50
Total Runtime	**719.62**	**1,444.47**

Test #1 Conclusion

The overall pipeline on WEKA **was 2X faster** than compared to the S3+Local NVMe option (which is one of the standard high-performance options for Sagemaker on AWS).

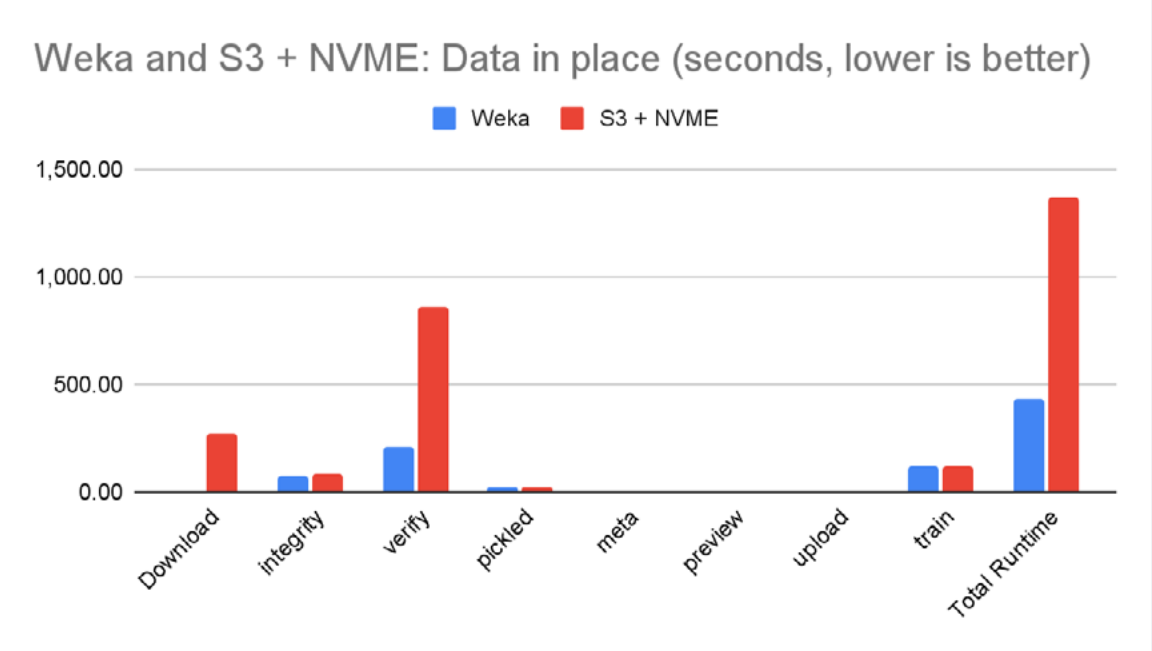
The Verify step on WEKA **was ~4X faster** compared to the S3+Local NVME option.

With these metrics, providing a similar pattern to the first experiment, we can confidently conclude that the WekaFS performance is at least at parity for ingesting a new dataset, verifying integrity for the tarball archive and the contents of the archive, and driving a training pipeline.

Test #2 Results

The following chart shows how long multiple segments of a job took with the runtime in seconds, lower is better:

NOTE: As data is resident on WekaFS, the download element for the S3 + NVMe is the only one tracked. In this scenario it took Sagemaker 4 minutes to download the data from the S3 bucket to the local NVMe - as the size of the data increases this time will increase as well. With WekaFS this will always remain 0 seconds since the data is already resident and does not need to use local NVMe or AWS S3 storage.



	WEKA	S3 + NVMe
Download	0	275.55
integrity	77.98	81.01
verify	206.48	858.26
pickled	23.89	23.09
meta	0.01	0.02
preview	4.11	4.46
upload	5.44	5.33
train	117.45	117.42
Total Runtime	**435.37**	**1,365.13**

Test #2 Conclusion

We can see that eliminating the stage of downloading the data to the WEKA NVMe tier shortens the overall runtime on the WEKA system. Compared to S3+NVMe WekaFS **is more than 3X Faster than S3+NVME.**

By using WekaFS to ingest a new dataset, verify the integrity of the tarball archive as well as the contents of the archive, and drive a training pipeline, we can confidently conclude that using WekaFS improves the entire pipeline runtime and can even shorten it significantly.

NOTE: Removing the download step of data from a data source is a key performance concern for the overall runtime of the job.

Using a local webserver to load the dataset, a local instance of apache took on average over four minutes. For larger archives being streamed in from S3, this will vary, driving an even more aggressive WEKA advantage.

Appendix 1: Preparation steps for a single client for all tests

GPU Instance

details specified above

WEKA Client Install

```
curl http://172.31.9.140:14000/dist/v1/install | sudo sh
```

Creating a mount point

```
mkdir /mnt/weka
```

Mount WEKA filesystem

```
sudo mount -t wekafs 172.31.9.140/default /mnt/weka
```

Obtain blog entry setup shell script

```
wget https://raw.githubusercontent.com/aws-samples/amazon-sagemaker-local-mode/main/blog/pytorch_cnn_
cifar10/setup.sh
```

Run setup + install required python3 libraries

```
bash setup.sh
pip install -U sagemaker
pip install jupyterlab
pip install torch
pip install pipreqs
pip install --upgrade pip
pip install torchvision
pip install matplotlib
pip install 'sagemaker[local]'
pip install ipywidgets
```

Configure docker external pull access

```
docker login
docker run --runtime=nvidia --rm nvidia/cuda nvidia-smi
cd amazon-sagemaker-local-mode/
cd blog
```

Enter s3 bucket test for boto inside python notebook

```
mkdir /home/ubuntu/.aws
cd /home/ubuntu
vi ~/.aws/credentials
chown -R ubuntu .aws
chgrp -R ubuntu .aws
```

Enter s3 bucket test for S3cmd

```
apt install s3aws
apt install s3cmd
s3cmd --configure
s3cmd ls s3://wekaiosa
s3cmd --recursive put data/ s3://wekaiosa/data/
s3cmd  delete s3://wekaiosa/data
```

Allow docker access to the config.json file as ubuntu uid

```
chmod a+r /home/ubuntu/.docker/config.json
chmod 777 /home/ubuntu/.docker/config.json
chown -R ubuntu /home/ubuntu/.docker
docker stats
```

Execution of scripts

From the notes in the repository

```
pip3 install -r requirements.txt
```

Run control script to validate environment:

```
python3 Control-2-pytorch_local_mode_cifar10.py
```

If this script works, you are ready to test for both the S3 + NVMe and WEKA performance tests:

```
python3 S3_pytorch_local_mode_cifar10_working_s3-only-new-loader-extended-2.0.py
python3 Weka_mnt_pytorch_local_mode_cifar10_working_s3_on_weka-new-loader-extended-2.0.py
```

This generates log files for each script with timings. Parsing the data logs with the shell scripts allows you to create timings for each segment of the job.

Final thoughts on access to data

Without access to key data assets, small scoops, and digests only provide tiny insights, however, when attached to a petabyte scale platform, the real strength of the solution enhances not just time to results, but expedites access to key features previously undiscovered.

Sagemaker Local Mode has the potential for producing impressive outcomes via connecting a rich library deployed in docker containers, but it is enhanced when attached to a massive amount of data on the WEKA AI-optimized data platform.

Utilizing Sagemaker Local Mode with WekaFS accelerates the data exploration significantly allowing researchers to train and learn faster than alternative methods while using a simpler and less expensive environment.

WKA348-01 03/2023