



Architecture Monthly

December 2021

Agriculture



EDITOR'S NOTE

[Amazon Web Services \(AWS\)](#) helps agriculture customers forecast supply and demand and create and maintain responsive, resilient food systems. This edition of Architecture Monthly focuses on the agriculture industry and their role in providing products to the world that are nutritious, healthy, accessible, affordable, and sustainable.

We'd like to thank our expert, Karen Hildebrand, Worldwide Tech Lead for Agriculture at AWS.

This month, we're also asking you to take a 10-question survey about your experiences with this magazine. The survey is hosted by an external company (Qualtrics), so the survey link doesn't lead to our website. Please note that AWS will own the data gathered from this survey, and we will not share the results we collect with survey respondents. Your responses to this survey will be subject to Amazon's Privacy Notice. Please take a few moments to give us your opinions.

[Take the survey](#)

Please give us your feedback! Include your comments on the [Amazon Kindle](#) page. You can [view past issues](#) and reach out to aws-architecture-monthly@amazon.com anytime with your questions and comments.

Bonnie McClure, Managing Editor

TABLE OF CONTENTS

- [Ask an Expert](#): Karen Hildebrand, PhD, Worldwide Tech Lead for Agriculture at AWS
- [Case Study](#): AGCO Lowers Costs, Boosts Speed, and Increases Retention Using Amazon Kinesis Services
- [Quick Start](#): Document Understanding Solution
- [Blog](#): Building a Controlled Environment Agriculture Platform
- [Case Study](#): ProRefrigeration Extends Their AWS IoT Capabilities to the Edge with CEI America
- [Quick Start](#): Genomics Tertiary Analysis and Machine Learning Using Amazon SageMaker
- [Solution](#): Improving Forecast Accuracy with Machine Learning
- [Blog](#): Building Machine Learning at the Edge Applications using Amazon SageMaker Edge Manager and IoT Greengrass v2
- [Blog](#): Processing satellite imagery with serverless architecture
- [Videos](#): The Future of Fresh Fruit Harvest: A Talk with FFRobotics, Seeing is Believing – the Rise of Data & Analytics in Agriculture, The Season for Vintner Innovation: Viticulture & More with E.J. Gallo Winery, Farming Innovation in Ireland: A Talk with Herdwatch, Voice Activated Agriculture, Amazon Prime – Clarkson’s Farm Season 1

NOTICES

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved.

ASK AN EXPERT

Karen Hildebrand, PhD, Worldwide Tech Lead for Agriculture at AWS

Agriculture and clouds have a long relationship. It has always been crucial to know when it's going to rain, and water is an important factor for all growing conditions. Modern agriculture technologies also rely on understanding the relationship between weather and climate, which means understanding data about what's happening — or what will happen — in the atmosphere. Even in controlled-environment agriculture, the temperature outside drives the combination of internal controls that are needed to maintain an ideal recipe for growth!

What are the general architecture pattern trends for agriculture in the cloud?

Feeding the world with nutrition that is healthy, accessible, affordable, and sustainable is a problem that no one company or technology can solve alone. It will require an ecosystem of innovative solutions that challenge the status quo. My customers are addressing this challenge by focusing on these areas:

1. **Forecasting** is critical to agricultural companies, from projecting supply/demand to consumer consumption patterns. To help customers enrich the data they gather, the [Registry of Open Data on AWS](#) hosts commonly used datasets for data science and forecasting. The [Open data on AWS supports sustainable agricultural practices and crop optimization](#) blog post provides a great example of collaboration between BASF, the US National Oceanic and Atmospheric Administration (NOAA), and the [Amazon Data Sustainability Initiative \(ASDI\)](#). They created hyper-specific ensemble weather models and made them available in Zarr and Parquet formats. This allows data scientists and machine learning (ML) engineers to use historical and future weather models in their artificial intelligence (AI)/ML algorithms.
2. **Resiliency** reflects our ability to respond to the unprecedented, including natural disasters like drought, fire, and flood. These events can change agricultural conditions, constrain supply chains and trade initiatives, and affect the movement and mobility of workers.

The [Improving Forecast Accuracy with ML solution](#) now incorporates the latest features in [Amazon Forecast Weather Index](#). If you're training a predictor using CNN-QR, DeepAR+, and Prophet algorithms, like predicting yield potential of a given commodity or the market price based on foot traffic to supermarkets, adding weather data can improve their accuracy.



[AWS Industry Insider - Agriculture - Dr. Karen Hildebrand](#)

Technology can help us adapt to a changing climate. [Computer vision](#) monitors farms, orchards, and processing facilities; Earth observation informs yield potential by analyzing satellite imagery and soil mapping; and the [Standardizing quantification of expression data at Corteva Agriscience with Nextflow and AWS Batch](#) blog post shows how customers can leverage partner solutions built on AWS to accelerate variety development for plant genomics.

3. **Innovation and collaboration** underpin the success of our current and future food systems. It starts with customers, particularly startups, having easy ways to build without compromising security and best practices. [Build on AWS](#) (part of [AWS Activate](#)) enables startups to use pre-built architectural guidance and solutions.

Offerings like [QnABot](#) are used by more seasoned agricultural customers to provide multi-language, multi-channel voice-driven interactions. These bots are able to help end users set up and maintain grow houses, field tools, etc. The [AAMOS SmallSat Toolkit on AWS](#) lets customers experiment and prototype [CubeSats](#), which offer connectivity to rural locations so that farmers can easily and quickly access and assess their data.

Collaboration takes many forms. By working with customers to assess their needs, AWS launched the [Service Workbench on AWS solution](#). With this service, researchers can quickly and securely stand up research environments and conduct experiments with peers from other institutions. For example, when the [University of Adelaide in Australia and the Plant Breeding and Acclimatization Institute](#)



[in Poland](#) partnered to analyze 48 wheat exomes and 18 whole barley genomes, they were able to complete their analysis in 6 hours. Normally, this analysis would take at least two weeks on the available local infrastructure.

When building an AWS architecture to solve business problems for agriculture customers, what are some of your considerations?

Architectural considerations in agriculture are all about assessing your customer's needs. The key questions I like to ask customers are:

1. What actions COULD you make (or are already making right now)?
2. What convinces you it was the right action to take?
3. If this action could be automated, how high would your confidence have to be that the decision was the right one?
4. How would you like to receive a recommended change in action?
5. Where would you be ideally when you receive an action?

Often the answers to these questions help us revise our solutions to better suit the customer's needs. Then we'll evaluate the

technical design considerations to make sure we aren't overlooking physical barriers and ensure we build in security best practices, resiliency, and cost efficiency.

Do you see different trends in agriculture in cloud versus on-premises?

Customers like [E. & J. Gallo](#), whose complex supply chain brings one in every three bottles of wine consumed in the world to consumers globally works with a partner to migrate their SAP workloads to the cloud. This ensures they can expand and provision excess capacity as needed, without the delay of on-premises hardware provisioning.

[Hortifrut](#), the biggest blueberry distributor in the world and whose Naturipe berries are often on my own kids' plates [have been successful running SAP](#) on the cloud to achieve higher availability and scalability.

Agricultural customers have also used the cloud to scale edge workloads for on-premises sensor data. [Perennia Food & Agriculture Inc.](#) built a tool that uses [Amazon SageMaker Edge Manager](#) to ingest data from various IoT edge devices, including power sensors, fuel sensors, GPS tracking, and water monitoring to having a clear understanding of their cost of production. [Grovet Technologies](#) used [AWS IoT Greengrass 2.0](#) to create a controlled environment agriculture solution for livestock feed rations, shortening the feed supply chain and ensuring high quality rations.

What's your outlook for agriculture, and what role will cloud play in future development efforts?

Agriculture is one of the world's oldest professions, but a farmer or producer typically has only 30-40 growing seasons in their career.

That's only 30-40 times in their lifetime to do different—and to put their livelihood—and the food supply on the line each time they do. That may work if you're in the 30th crop cycle. You have years of data gathered to know what to expect, but what if you're in year 2 or 5 and you're staring in the face of climate change? Your experience isn't likely to be enough.

As the supply chain for food is increasingly integrated, ensuring the primary producer, the processor, and ultimately the consumer have choice requires they have data they trust. The data indicates which variety of crop will grow best in a changing climate, what weed needs to be sprayed for increased precision and sustainability, or what the animal health and wellness scores are for a given dairy or swine herd. The backbone of our food supply becomes more resilient and adaptable when it runs on the cloud. It scales to changing conditions, enables scenario building, and encourages security in ways that are difficult if not impossible to achieve affordably.

ABOUT THE EXPERT



Dr. Karen Hildebrand is a fourth-generation farmer in Canada, and is deeply passionate about agriculture. She leads the Worldwide Agriculture

Solutions Architecture segment, working with global customers to maintain global food supply chains and create sustainable solutions for the agriculture industry. Prior to AWS, she held progressive leadership positions at several multi-national Fortune 500 companies.

CASE STUDY

AGCO Lowers Costs, Boosts Speed, and Increases Retention Using Amazon Kinesis Services

2021


[AGCO Corporation \(AGCO\)](#), a global provider of agricultural solutions, manufactures industrial farming equipment and cutting-edge digital products designed to revolutionize farms. Its goal of data-driven precision farming solutions means enabling farmers to manage sustainable, productive grain and livestock farms and to maximize yield and profitability. AGCO required modern technology to enable the scale of data tracking and storage necessary for smart farming. For that, it turned to Amazon Web Services (AWS).

The company began its AWS journey with a 2015 migration, and since 2019 it has been focusing on modernizing with microservices, automation, and serverless architecture. AGCO's former third-party black-box data system limited how the company could use its data. Crucially, it didn't have enough data stream retention time to account for the long

growth and investment cycles in farming. By replacing that system with an orchestration of AWS services, including three [Amazon Kinesis](#) services, the company could get timely insights and react quickly to new information. AGCO dramatically increased data retention time and data-retrieval speed, and it did so while reducing costs, which enables its digital-first strategy and serves as a cornerstone for future capabilities.

Replacing Its Legacy System with a Cost-Efficient Architecture

Farmers are increasingly using digital technologies to run their farms more sustainably, and AGCO is building products to help farmers manage their operations more efficiently and profitably. "We build farms where edge computing runs the show: the ventilation, the lighting, the heating. They're all hooked into the building controller and feeding the cloud," explains Paul Thornhill, product development manager at AGCO. "The cloud collects and analyzes the data, then sends the results back to the building, which adjusts the ventilation, lighting, and heating accordingly. It reports everything to the farmer." By automating that process, farmers can not only save time but also make



"Amazon Kinesis services provide the building blocks we need at a lower cost. We get a 1:3 or 1:4 return because they cover 60% or more of what we were paying before."

Paul Thornhill

Product Development Manager, AGCO

sure that temperature, humidity, and other parameters are always set at the optimal levels to encourage growth without wasting money on electricity.

On its legacy system, AGCO couldn't enable farmers to compare growth cycles. It had a retention limit of 750 million records, which was only sufficient for a period of about 6 months—barely enough for even one 185-day cycle of bringing a pig from birth to market. Because of the 180,000 records each customer stores per day and the challenge of data retention, AGCO was actually spending more money than it was earning to onboard new customers.

AGCO needed technology that would collect all data from its smart farming equipment and then retain and analyze the data to provide insights and real-time machine diagnostics. Most options to accomplish that goal are expensive and time consuming, but Amazon Kinesis services can do the job by offering managed services and automatic scaling, seamlessly operating in AGCO's existing AWS infrastructure.

Reducing Cost and Improving Speed

In January 2020, AGCO went live with a serverless data pipeline built using three Amazon Kinesis services. Ingesting and storing data from the AGCO farming equipment is [Amazon Kinesis Data Streams \(Amazon KDS\)](#), a scalable and durable real-time data streaming service that continuously captures gigabytes of data per second from hundreds of thousands of sources. Amazon KDS can then feed equipment data into [Amazon Simple Storage Service \(Amazon S3\)](#), an object storage service. Also capturing equipment data is [Amazon Kinesis Data Firehose](#), a simple way to reliably

load streaming data into analytics services, data lakes, and data stores, which AGCO shapes for particular customer use cases. A downstream Amazon Kinesis Data Firehose makes data available for use with other AGCO products. Using these tools, AGCO's system moves 1,200 data points per minute and, when tested, could automatically scale to 10,000 data points per minute.

Once the data is ingested, it can either be stored or go directly to [Amazon Kinesis Data Analytics](#), which transforms and analyzes streaming data in real time using Apache Flink, an open-source analytics framework. Using this service results in valuable insights for farmers. "We want to give farmers data-driven knowledge about how the farm runs," says Thornhill. "Then we want to give them insights to improve the production process and more real-time diagnostic feedback about each machine and how to adjust it to match best practices and benchmarks."

Using Amazon Kinesis services, AGCO retains 1.5 billion records at a lower cost, resulting in 78 percent cost savings. It can consistently load that data to a screen in 600 ms—a process which previously took 8–30 seconds. AGCO also gained more data storage at a drastically reduced cost by using Amazon S3.

Another cost saver for AGCO was the ability to end its third-party contracts. "Even though there's now more labor on our side to process the data, it's a better ratio than when we were paying someone else to do it," Thornhill says. "Amazon Kinesis services provide the building blocks we need at a lower cost. We get a 1:3 or 1:4 return because they cover 60 percent or more of what we were paying before."

AGCO layers Amazon Kinesis services on top of the existing infrastructure to reduce its complexity and manage the underlying service.

As a result, AGCO only needs one person to manage the infrastructure rather than three to five. “Even with the amount of data that we’re currently moving and the complexity of its infrastructure, the pipeline has run nearly hands-free all year,” says Thornhill.

Increasing Productivity and Profitability for Farmers

AGCO plans to use AWS to build true edge communication through an Internet of Things core instead of a hand-built gateway. Using machine learning to analyze weather data, for example, would enable farms to adjust temperature and humidity in a barn minutes before a weather change rather than the usual 2 hours after. With the potential to adjust an environment without involving the farmer, this system would lead to increased production.

To further improve animal welfare, AGCO plans to develop a solution for real-time monitoring of sows. This will enable farmers to make quick, data-based adjustments to the sows’ environments to improve their well-being and increase productivity and profitability.

Using AWS, AGCO improved service for its agricultural customers by lowering costs and increasing data retention and retrieval speed. “Data collected and delivered to our farmers using AWS tools is driving better decisions and animal health,” Thornhill says. AGCO plans to



continue using AWS to adapt to the changing world of agriculture.

About AGCO Corporation

Headquartered in the United States and operating worldwide, AGCO Corporation is an innovator of agricultural solutions. Its goal is to create sustainable, productive grain and livestock farms through data.

Benefits of AWS

- Decreased costs by 78%
- Increased data retention to 1.5 billion records and growing
- Reduced data extraction time from 8–30 seconds to 600 milliseconds
- Manages data pipeline with one person rather than 3–5 people

[Read case study online](#)



QUICK START

Document Understanding Solution

What does this AWS Solutions Implementation do?

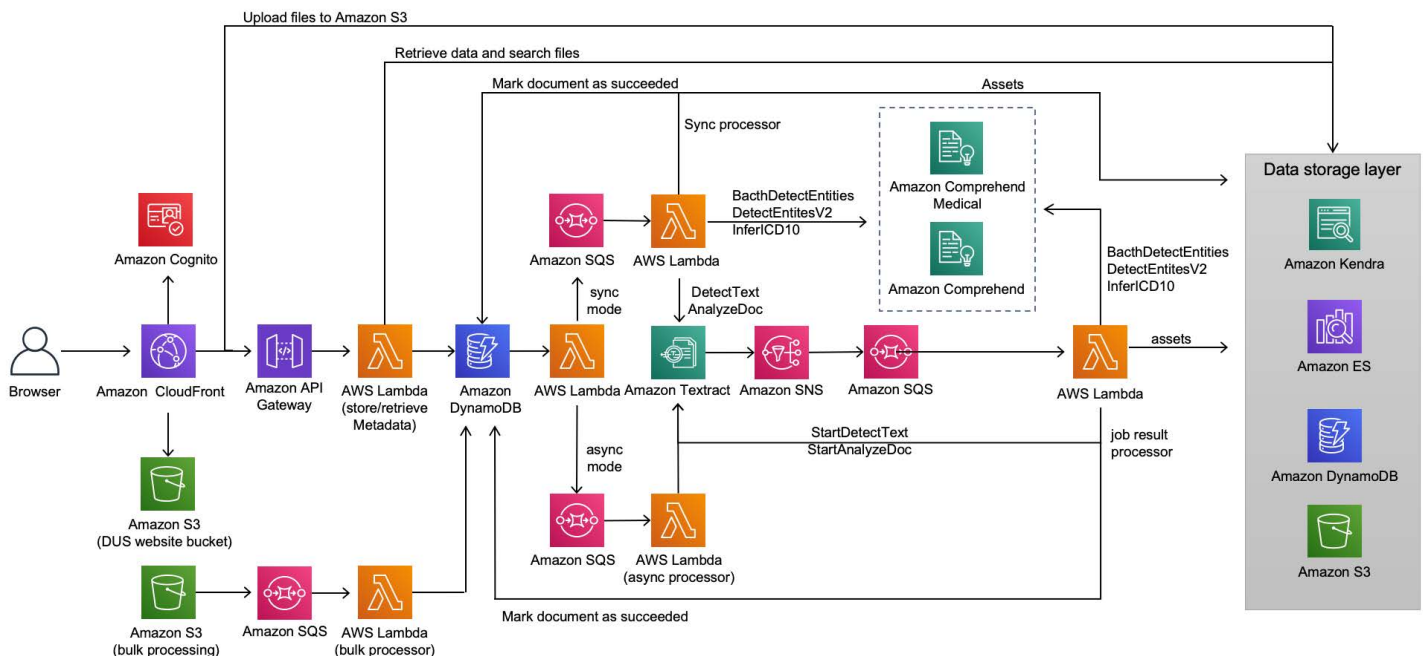
The Document Understanding Solution (DUS) delivers an easy-to-use web application that ingests and analyzes files, extracts text from documents, identifies structural data (tables, key value pairs), extracts critical information (entities), and creates smart search indexes from the data. Additionally, files can be uploaded directly to and analyzed files can be accessed from an Amazon Simple Storage Service (Amazon S3) bucket in your AWS account.

This solution uses [AWS artificial intelligence \(AI\)](#) services that address business problems that apply to various industry verticals:

- Search and discovery: Search for information across multiple scanned documents, PDFs, and images
- Compliance: Redact information from documents
- Workflow automation: Easily plugs into your existing upstream and downstream applications

AWS Solutions Implementation overview

The diagram below presents the architecture you can automatically deploy using the solution's implementation guide and accompanying AWS CloudFormation template.



Document Understanding Solution architecture

The [AWS CloudFormation](#) template deploys a static web application hosted on an Amazon S3 bucket and served by an [Amazon CloudFront](#) distribution. Users are authenticated using [Amazon Cognito](#). The web application interacts with the backend using an [Amazon API Gateway](#) API, supported by an [AWS Lambda](#) function. Documents are uploaded using either the web application, or directly to a dedicated Amazon S3 bucket for bulk processing. Document processing is initiated by the API, which triggers a Lambda function to add an entry to an [Amazon DynamoDB](#) table. The table triggers a second Lambda function that supervises the processing. The file format of the upload dictates the route for processing. [Amazon Textract](#) extracts text and structural information from the files. The extracted text is then passed to [Amazon Comprehend](#) and [Amazon Comprehend Medical](#) for further analysis.

The resulting analyses are stored in an Amazon S3 bucket and the metadata is stored in a DynamoDB database. Extracted information is used to index the document in [Amazon Elasticsearch Service \(Amazon ES\)](#) and, if enabled, in [Amazon Kendra](#).

[View full quick start online](#)

[View implementation guide](#)



Building a Controlled Environment Agriculture Platform

by Ashu Joshi

This post was co-written by Michael Wirig, Software Engineering Manager at Grōv Technologies.

A substantial percentage of the world's habitable land is used for livestock farming for dairy and meat production. The dairy industry has leveraged technology to gain insights that have led to drastic improvements and are continuing to accelerate. A gallon of milk in 2017 involved 30% less water, 21% less land, a 19% smaller carbon footprint, and 20% less manure than it did in 2007 ([US Dairy, 2019](#)). By focusing on smarter water usage and sustainable land usage, livestock farming can grow to provide sustainable and nutrient-dense food for consumers and livestock alike.

[Grōv Technologies \(Grōv\)](#) has pioneered the Olympus Tower Farm, a fully automated Controlled Environment Agriculture (CEA) system. Unique amongst vertical farming startups, Grōv is growing cattle feed to improve that sustainable use of land for livestock farming while increasing the economic margins for dairy and beef producers.

The challenges of CEA

The set of growing conditions for a CEA is called a "recipe," which is a combination of ingredients like temperature, humidity, light, carbon dioxide levels, and water. The optimal recipe is dynamic and is sensitive to its ingredients. Crops must be monitored in

near-real time, and CEAs should be able to self-correct in order to maintain the recipe. To build a system with these capabilities requires answers to the following questions:

- What parameters are needed to measure for indoor cattle feed production?
- What sensors enable the accuracy and price trade-offs at scale?
- Where do you place the sensors to ensure a consistent crop?
- How do you correlate the data from sensors to the nutrient value?

To progress from a passively monitored system to a self-correcting, autonomous one, the CEA platform also needs to address:

- How to maintain optimum crop conditions
- How the system can learn and adapt to new seed varieties
- How to communicate key business drivers such as yield and dry matter percentage

Grōv partnered with [AWS Professional Services \(AWS ProServe\)](#) to build a digital CEA platform addressing the challenges posed above.



Olympus Tower - Grōv Technologies

Tower automation and edge platform

The Olympus Tower is instrumented for measuring recipe ingredients by combining the mechanical, electrical, and domain expertise of the Grōv team with the IoT edge and sensor expertise of the AWS ProServe team. The teams identified a primary set of features such as height, weight, and evenness of the growth to be measured at multiple stages within the Tower. Sensors were also added to measure secondary features such as water level, water pH, temperature, humidity, and carbon dioxide.

The teams designed and developed a purpose-built modular and industrial sensor station. Each sensor station has sensors for direct measurement of the features identified. The sensor stations are extended to support indirect measurement of features using a combination of Computer Vision and Machine Learning (CV/ML).

The trays with the growing cattle feed circulate through the Olympus Tower. A growth cycle starts on a tray with seeding, circulates through the tower over the cycle, and returns to the starting position to be harvested. The sensor station at the seeding location on the Olympus Tower tags each new growth cycle in a tray with a unique “Grow ID.” As trays pass by, each sensor station in the Tower collects the feature data. The firmware, jointly developed for the sensor station, uses [AWS IoT SDK](#) to stream the sensor data along with the Grow ID and metadata that’s specific to the sensor station. This information is sent every five minutes to an on-site edge gateway powered by [AWS IoT Greengrass](#). Dedicated [AWS Lambda](#) functions manage the lifecycle of the Grow IDs and the sensor data processing on the edge.

The Grōv team developed AWS Greengrass Lambda functions running at the edge to ingest critical metrics from the operation automation software running the Olympus Towers. This information provides the ability to not just monitor the operational efficiency, but to provide the hooks to control the feedback loop.

The two sources of data were augmented with site-level data by installing sensor stations at the building level or site level to capture environmental data such as weather and energy consumption of the Towers.

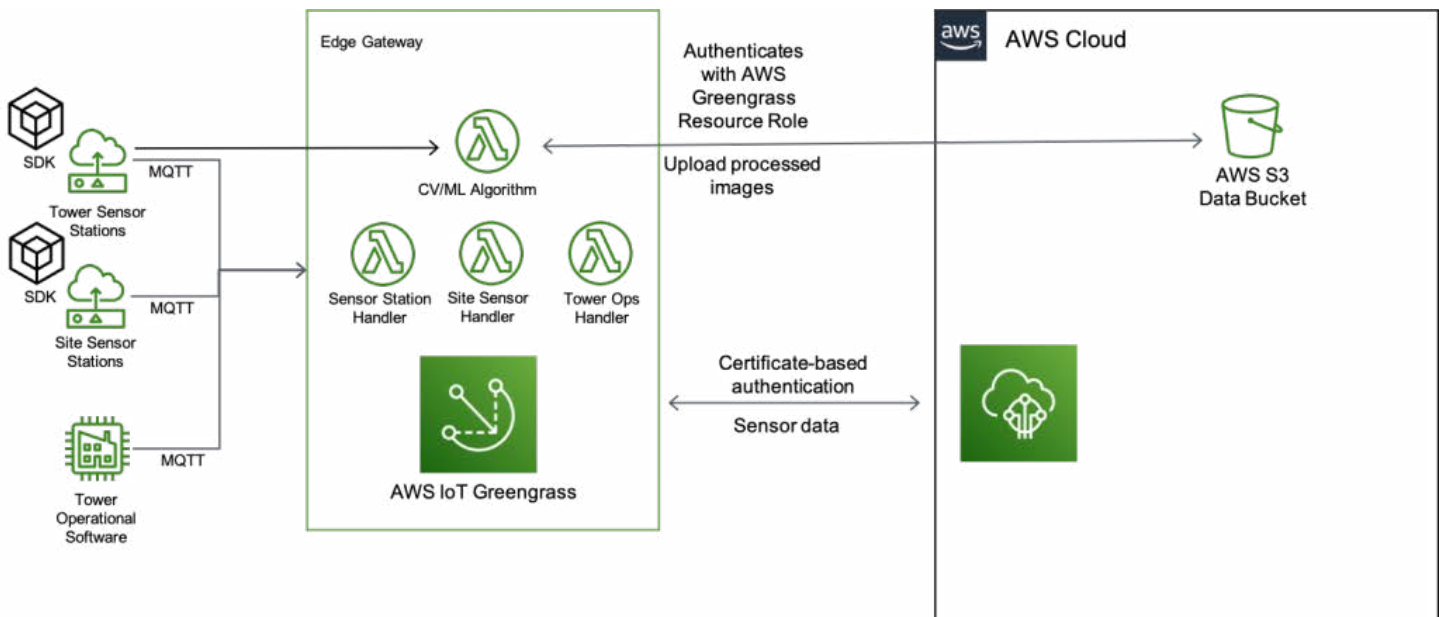
All three sources of data are streamed to AWS IoT Greengrass and are processed by AWS Lambda functions. The edge software also fuses the data and correlates all categories of data together. This enables two major actions for the Grōv team – operational capability in real-time at the edge and enhanced data streamed into the cloud.

Cloud pipeline/platform: analytics and visualization

As the data is streamed to [AWS IoT Core](#) via AWS IoT Greengrass, [AWS IoT rules](#) are used to route ingested data to store in [Amazon Simple Storage Service \(Amazon S3\)](#) and [Amazon DynamoDB](#). The data pipeline also includes [Amazon Kinesis Data Streams](#) for batching and additional processing on the incoming data.

A [ReactJS](#)-based dashboard application is powered using [Amazon API Gateway](#) and AWS Lambda functions to report relevant metrics such as daily yield and machine uptime.

A data pipeline is deployed to analyze data using [Amazon QuickSight](#). [AWS Glue](#) is used to create a dataset from the data stored in Amazon S3. [Amazon Athena](#) is used to query the dataset to make it available to Amazon



Gröv Technologies - Architecture

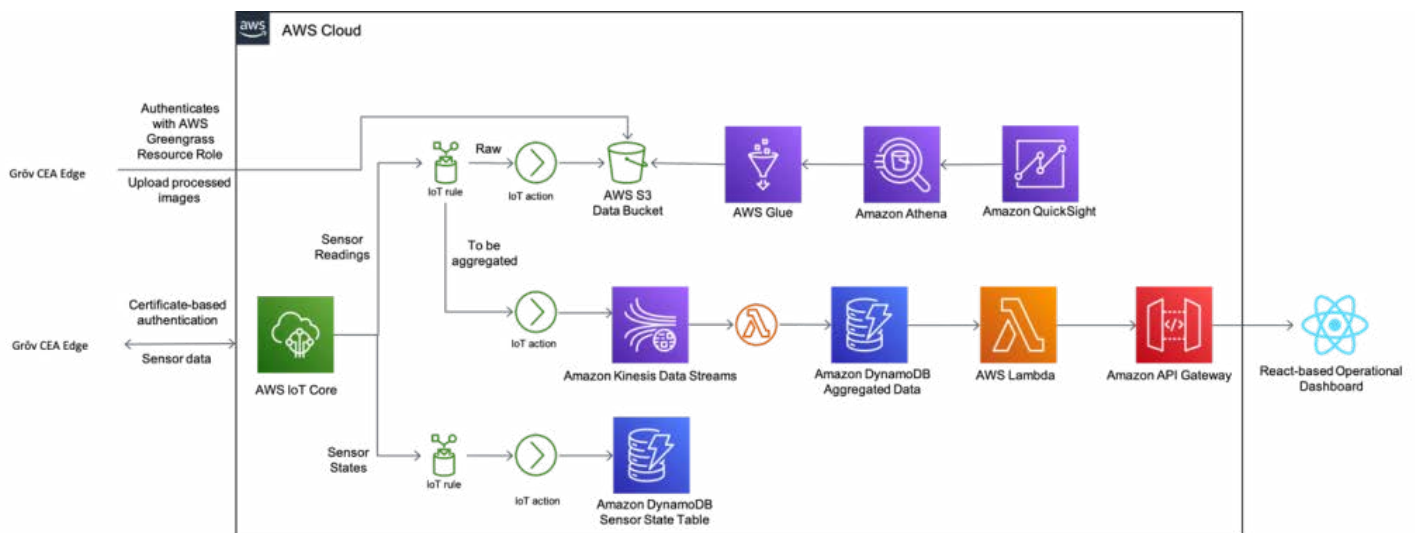
QuickSight. This provides the extended Gröv tech team of research scientists the ability to perform a series of what-if analyses on the data coming in from the Tower Systems beyond what is available in the react-based dashboard.

Completing the data-driven loop

Now that the data has been collected from all sources and stored it in a data lake architecture, the Gröv CEA platform

established a strong foundation for harnessing the insights and delivering the customer outcomes using machine learning.

The integrated and fused data from the edge (sourced from the Olympus Tower instrumentation, Olympus automation software data, and site-level data) is co-related to the lab analysis performed by Gröv Research Center (GRC). Harvest samples are routinely collected and sent to the lab, which



Data pipeline - Gröv Technologies

performs wet chemistry and microbiological analysis. Trays sent as samples to the lab are associated with the results of the analysis with the sensor data by corresponding Grow IDs. This serves as a mechanism for labeling and correlating the recipe data with the parameters used by dairy and beef producers – dry matter percentage, micro and macronutrients, and the presence of myco-toxins.

Grōv has chosen [Amazon SageMaker](#) to build a machine learning pipeline on its comprehensive data set, which will enable fine tuning the growing protocols in near real-time. Historical data collection unlocks machine learning use cases for future detection of anomalous sensors readings and sensor health monitoring, as well.

Because the solution is flexible, the Grōv team plans to integrate data from animal studies on their health and feed efficiency

into the CEA platform. Machine learning on the data from animal studies will enhance the tuning of recipe ingredients that impact the animals' health. This will give the farmer an unprecedented view of the impact of feed nutrition on the end product and consumer.

Conclusion

Grōv Technologies and AWS ProServe have built a strong foundation for an extensible and scalable architecture for a CEA platform that will nourish animals for better health and yield, produce healthier foods and to enable continued research into dairy production, rumination and animal health to empower sustainable farming practices.

[Read blog online](#)



CASE STUDY

ProRefrigeration Extends Their AWS IoT Capabilities to the Edge with CEI America

The Challenge

ProRefrigeration has been manufacturing industrial chillers for over 30 years with specialization across several industries, including dairy, craft beverage (breweries and wineries), food processing, and extracting. Delivering reliable products and services has enabled ProRefrigeration to cultivate and sustain relationships with thousands of customers around the world. In 2018, ProRefrigeration extended their offering by employing Internet of Things (IoT) solutions which enabled them to stream telemetry data from their chillers to monitor performance, isolate anomalies, and prevent downtime using the company's PROElliot IoT application, running on Amazon Web Services (AWS).

PROElliot was announced at IoT World in Santa Clara in 2017 and later the Craft Brewing Conference in Nashville, Tennessee. As an early adopter of Industrial IoT (IIoT), ProRefrigeration disrupted the chiller market. Yet they also realized that many of their customers' use cases were in remote areas with limited connectivity, leaving them reliant on WiFi connection(s). This limited ProRefrigeration's ability to extend the solution to emerging cold chain use cases, including dairy, because they require constant connectivity and fidelity to ensure the milk stays at or below 40 degrees F throughout the production process. This is a critical

measurement, as the bacteria load in milk doubles every 20 minutes at or above this temperature. This helps confirm that the product offered for consumption to the public is not only safer but also of a higher quality with less bacterial load. This also makes it possible for dairy farmers to charge a higher premium (5-10%) for their product, while reducing the producer's liability, which has resulted in less expensive insurance premiums for them.

Tracking the temperature and status of milk and dairy products has historically been done manually, using a government mandated circular paper chart, for 50-plus years. Sensing a modernization opportunity that could help farmers increase efficiency and reduce costs, ProRefrigeration, digitally replicated the chart in PROElliot to automate the collection and reporting of data. This is achieved by leveraging telemetry data from the chiller systems, heat exchangers, milk storage tanks, and other related assets. This automates the process of getting this crucial data from the equipment into the cloud and provides real-time results that can accurately identify when immediate action is necessary.

Understanding that they had created a potentially revolutionary IIoT service for dairy farmers and producers, ProRefrigeration realized they would need help extending their PROElliot solution to the Edge specifically for cold chain use cases that require the system to operate in areas with limited connectivity. Demonstrating its intrinsic value to customers, and compelling them to adopt it, would be a key step.

For PROELLiot to address these emerging requirements ProRefrigeration enlisted the help of, CEI America, of Pittsburgh, Pennsylvania, to turn their vision into reality.

This meant delivering PROELIot with device capabilities that went beyond just providing access to real-time data but that could also operate in environments with intermittent connectivity by filtering messaging, data queuing, and sending local alerts if an anomaly was detected or an asset was trending towards failure.

This not only extended PROELIot for dairy use cases, it also positioned them to assist with emerging cold chain opportunities, including tracking COVID-19 vaccinations. This ensures the safe and secure distribution from the point of origin to where the vaccines are most needed to expedite the process of getting everyone on the road to recovery.

The Solution

Having gained an understanding of this request's details, and a clear definition of the expectations, CEI focused on leveraging native AWS IoT services, such as AWS IoT Greengrass. Integrating these components enables PROELIot to reliably ingest and deliver real-time data and reporting on the status of the ProRefrigeration chillers in use on dairy farms. This updated solution requires the deployment of sensors into holding tanks, on milk trucks, as well as in other core holding and transportation infrastructures. Once in place and operational, the sensors send telemetry data to AWS IoT Greengrass, AWS IoT Events, and other AWS IIoT services to track, monitor, report, and send immediate alerts if the milk or dairy product temperature is trending higher.

In addition, CEI's Edge solution enables PROELIot to track and ingest data over the course of the entire process. It does this by first capturing pertinent real-time data on the status of the dairy products, then

applies relevant third-party information (such as weather reports), before incorporating Artificial Intelligence (AI) to help accurately predict potential anomalies and trends.

The value of this solution is amplified by its capability to enable the system filter and store telemetry data offline while concurrently monitoring for cellular connectivity to subsequently send the offline data to AWS. This results in the quality of the milk and dairy products being accurately tracked throughout the entire process (from milking to distribution), even if WiFi or cellular connectivity is not available, and a real-time connection to AWS cannot be made.

Extending PROELIot's IIoT capabilities to the Edge has made it possible for dairy farmers to monitor the temperatures of their products more effectively. Using the built-in alerts and alarm notifications, farmers instantly learn when their cooling systems are trending the wrong way before they are adversely impacted. This helps to not only protect and increase the value of desired milk fat, but also ensures safe, healthy delivery of milk and dairy products for sale to consumers.

The Benefits

Implementation of this evolving solution has helped dairy farmers and distributors become more efficient. A practical example is demonstrated by auto triggering the milk tank agitator to engage when the milk truck enters the farm, while simultaneously sending a report back to the milk hauler with the current Cold Chain status and milk temperature. This enables dispatchers to schedule their pickups and deliveries more efficiently while reducing the time spent waiting for product to be loaded. As a result, the drivers can go to more

farms and pickup additional product over the course of days, months, and years.

Incorporating reliable AWS IoT and AI capabilities into their solution has enabled CEI to help PROElliot effectively leverage its edge functionality and more accurately evaluate key data, isolate potential trends, and more quickly identify the emergence of patterns that could affect business outcomes. The scalability and durability of the AWS IoT services ensure that thousands, if not millions, of devices can be scaled on demand, without concern for exceeding limits on the number of devices. Moreover, the scalability and elasticity of AWS compute and storage enables the ongoing ingestion of large data volumes which can be scaled up for immediate analysis and accelerates the process of detecting and isolating patterns and anomalies. Once completed, these can then be scaled back down to reduce costs and improve overall business outcomes.

AWS IoT—and specifically edge—technologies are critical to CEI's ability to build and deploy cold chain solutions that operate in environments where connectivity is not always reliable. These services continuously enhance both the solution and the pace at which new edge capabilities can be incorporated to add value. In the end, this helps reduce equipment downtime and associated costs, while ensuring the viability and value of the end-product.

Added Efficiency Helps Increase Productivity

As the value of this solution is realized, it becomes evident that CEI's use of AWS services has helped increase the proliferation of PROElliot's use among the dairy farming community. The constant monitoring has enabled farmers to charge an additional 5-10% for their "cold chain verified" products. By capturing this critical IoT temperature and

other data in real time, an average/large dairy farm (one that produces an estimated \$28,000 per day in milk), can increase the value of that same milk by \$1,400-\$2,800 per day. This estimation would generate an additional \$42K-\$82K for the same product/commodity, which transitions to \$500K/\$1M in additional annual revenue for the producer.

"It has been exciting to watch PRO as we develop and transform from an equipment manufacturer into a technology company that builds equipment!"

Jim VanderGissen Jr

Improve Data Access with Edge Functionality Results in Real Data Costs

The combination of CEI and AWS is delivering impressive cost savings benefits, as well as adding revenue to the bottom line, for ProRefrigeration's customers. The PROElliot Cold Chain Verification Platform will drive a 30% reduction in pre-cooling water, delivering **annual savings over \$100,000**, and leaving more than 10 MILLION gallons of water in the ground. Combined with herd health, the PROElliot Cold Chain Verification is vital to reaching the Premium Quality Bonus payments that adds \$.10 per CWT of Milk, **another \$87,600 in annual revenue**. When additional sanitation savings are included for a single 3,000 head farm, the **total estimated cost benefit is just under \$200K, annually**.

[Read case study online](#)





QUICK START

Genomics Tertiary Analysis and Machine Learning Using Amazon SageMaker

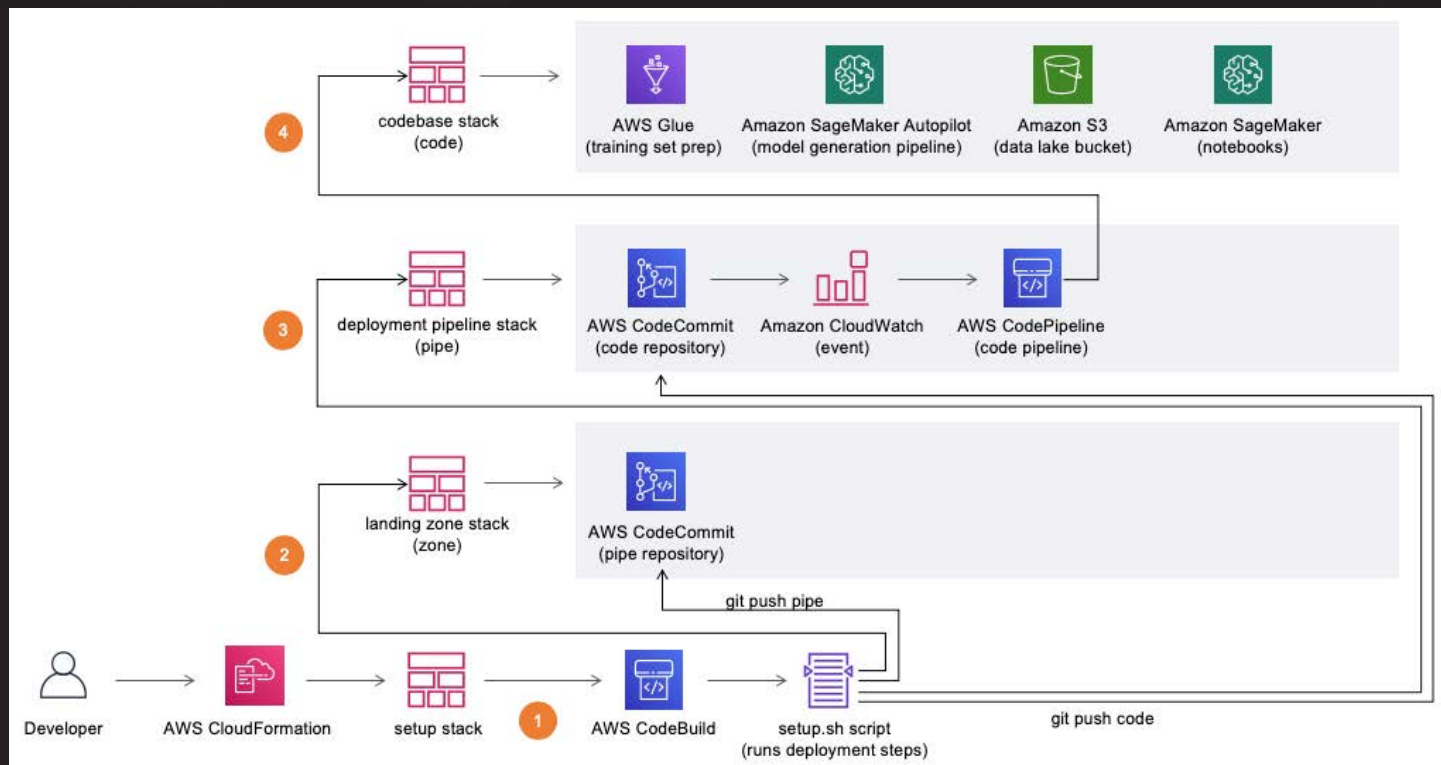
What does this AWS Solutions Implementation do?

The Genomics Tertiary Analysis and Machine Learning Using Amazon SageMaker solution creates a platform in the AWS Cloud that can be used to build machine learning models on genomic datasets using AWS managed services. We define *tertiary analysis* to be the interpretation of genomic variants and assigning meaning to them. This solution provides a broad platform for genomic machine learning in AWS, using variant classification as an example of a scientifically meaningful problem that can be solved using this platform. In the example, we solve the specific challenge of competing clinical definitions when examining genomic variants. Our example is based on the following [Kaggle challenge](#). We create a model to predict if a variant annotated in [ClinVar](#) has a conflicting classification or not. A model that can predict the existence of a conflicting classification for a variant can save valuable time that researchers have to spend looking for such conflicts.

This solution demonstrates how to 1) automate the preparation of a genomics machine learning training dataset, 2) develop genomics machine learning model training and deployment pipelines and, 3) generate predictions and evaluate model performance using test data. These steps can be repeated or edited by users for their specific use cases.

AWS Solutions Implementation overview

The diagram below presents the architecture you can automatically deploy using the solution's implementation guide and accompanying AWS CloudFormation template.



Genomics Tertiary Analysis and Machine Learning Using Amazon SageMaker solution architecture

The [AWS CloudFormation](#) template creates four CloudFormation stacks in your AWS account including a *setup* stack to install the solution. The other stacks include a landing zone (*zone*) stack containing the common solution resources and artifacts; a deployment pipeline (*pipe*) stack defining the solution's [continuous integration](#) and [continuous delivery](#) (CI/CD) pipeline; and a code base (*code*) stack providing the ETL scripts, jobs, crawlers, a data catalog, and notebook resources.

The solution's *setup* stack creates an [AWS CodeBuild](#) project containing the *setup.sh* script. This script creates the remaining CloudFormation stacks and provides the source code for both the [AWS CodeCommit](#) *pipe* repository and the *code* repository.

The landing zone (*zone*) stack creates the CodeCommit *pipe* repository. After the landing zone (*zone*) stack completes its setup, the *setup.sh* script pushes source code to the CodeCommit *pipe* repository.

The [AWS CodePipeline](#) *code* pipeline deploys the code base (*code*) CloudFormation stack. The resources deployed in your account include [Amazon Simple Storage Service \(Amazon S3\)](#) buckets for storing object access logs, build artifacts, and data; CodeCommit repositories for source code; an AWS CodeBuild project for building code artifacts (for example, third-party libraries used for data processing); a CodePipeline pipeline for automating builds and deployment of resources; example [AWS Glue](#) jobs; and an [Amazon SageMaker](#) Jupyter notebook instance. The example code includes the resources needed to quickly develop machine learning models using genomics data and generate predictions.

[View quick start online](#)

[View implementation guide online](#)

SOLUTION

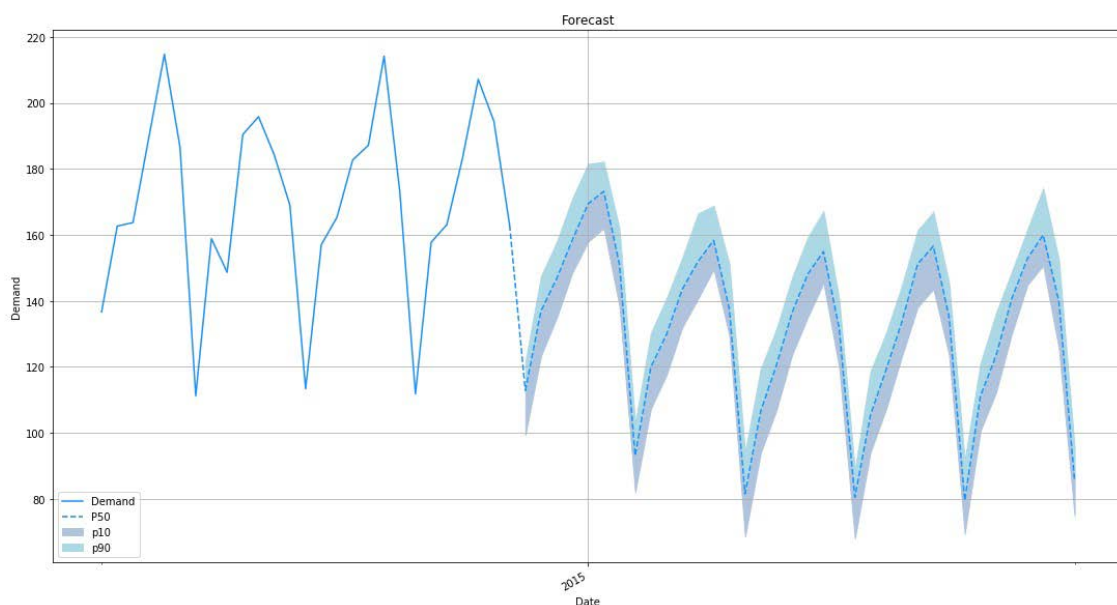
Improving Forecast Accuracy with Machine Learning

What does this AWS Solutions Implementation do?

The Improving Forecast Accuracy with Machine Learning solution generates, tests, compares, and iterates on Amazon Forecast forecasts. The solution automatically produces forecasts and generates visualization dashboards for Amazon QuickSight or Amazon SageMaker Jupyter Notebooks—providing a quick, easy, drag-and-drop interface that displays time series input and forecasted output. Forecasting can be applied to predict retail inventory demand, supply-chain planning, workforce status, web traffic forecasting, and more.

Forecasts can be compared across dimensions (for example, retail store location) or item-level metadata (for example, product brand, size, and color). You can use this data for the following:

- **Optimize existing forecasts:** Save time and retain compatibility with your legacy tools, or gain insight into over- and under-provisioning, with the p50 forecast.
- **Meet variable customer demand:** Provide high levels of customer satisfaction with the p90 forecast, where the true value is expected to be lower than the predicted value 90% of the time.
- **Avoid over-provisioning:** Save on costs and avoid over-provisioning with the p10 forecast, where the true future demand value is expected to be lower than the predicted value only 10% of the time.

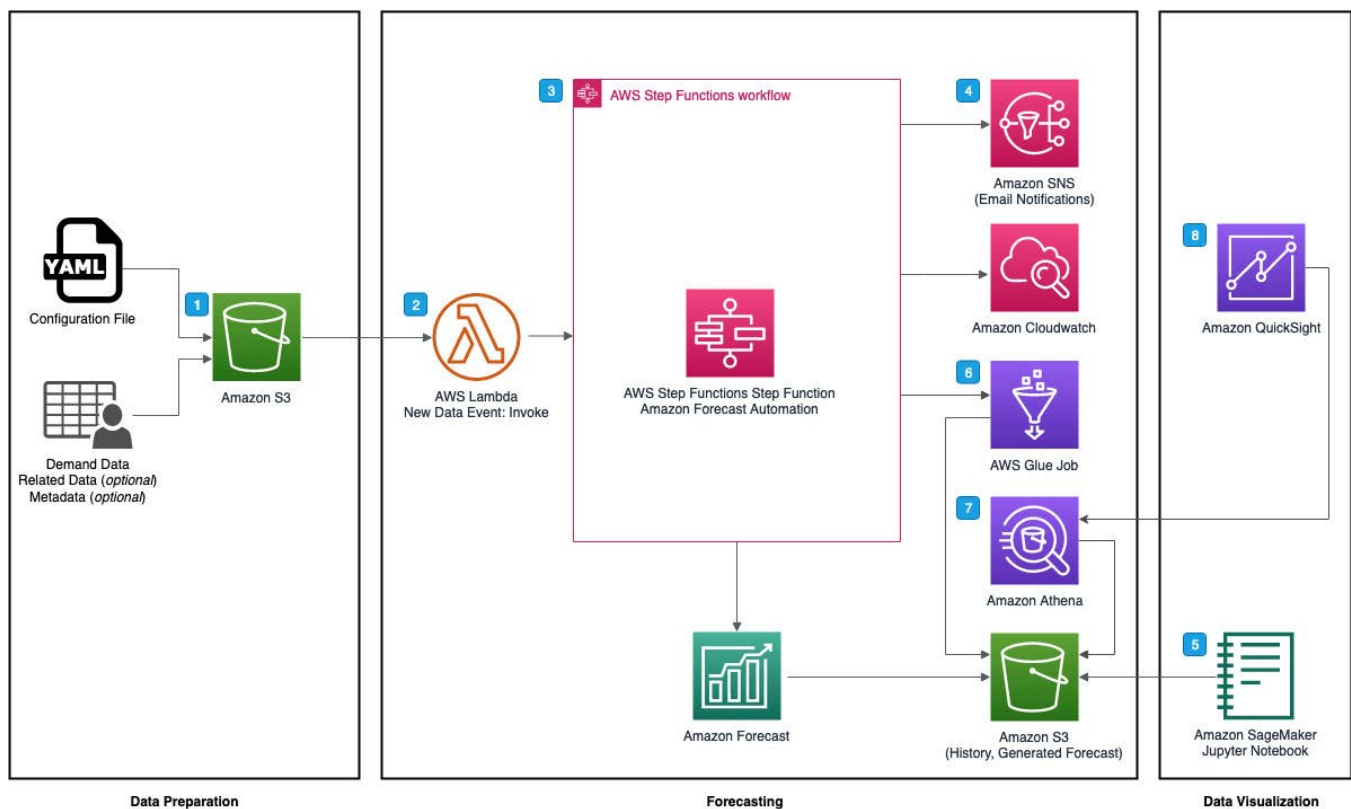


Benefits

- **Automated processes:** Streamline the process of ingesting, modeling, and forecasting multiple experiments through the automation of Amazon Forecast.
- **Secure deployment:** Provide a secure one-click deployment using an AWS CloudFormation template developed with the AWS Well-Architected Framework methodologies.
- **Proactive monitoring:** Easily monitor forecasts by emailing users when successes and failures occur.
- **Automated visualization:** Facilitate collaboration and experimentation by combining your input data and forecast output in an Amazon QuickSight Analysis or Jupyter Notebook.

AWS Solutions Implementation overview

Deploying this solution with the default parameters builds the following serverless environment in the AWS Cloud.



Improving Forecast Accuracy with Machine Learning solution architecture

The [AWS CloudFormation](#) template deploys the resources required to automate your [Amazon Forecast](#) usage and deployments. Based on the capabilities of the solution, the architecture is

divided into three parts: data preparation, forecasting, and data visualization. The template includes the following components:

1. An [Amazon Simple Storage Service \(Amazon S3\)](#) bucket for Amazon Forecast configuration where you specify configuration settings for your dataset groups, dataset predictors, and forecasts, as well as the datasets themselves.
2. An [Amazon S3 event notification](#) that triggers when new datasets are uploaded to the related Amazon S3 bucket.
3. An Improving Forecast Accuracy with Machine Learning [AWS Step Functions](#) state machine. This combines a series of [AWS Lambda](#) functions that build, train, and deploy your Machine Learning (ML) models in Amazon Forecast. All AWS Step Functions log to [Amazon CloudWatch](#).
4. An [Amazon Simple Notification Service \(Amazon SNS\)](#) email subscription that notifies administrative users with the results of the AWS Step Functions.
5. An [Amazon SageMaker](#) notebook instance that data scientists and developers can use to prepare and process data, and evaluate Forecast output.
6. An [AWS Glue](#) job combines raw forecast input data, metadata, predictor backtest exports, and forecast exports into an aggregated view of your forecasts.
7. [Amazon Athena](#) can be used to query your forecast output using standard SQL queries.
8. [Amazon QuickSight](#) analyses can be created on a per-forecast basis to provide users with forecast output visualization across hierarchies and categories of forecasted items, as well as item level accuracy metrics. Dashboards can be created from these analyses and shared within your organization.

[View solution online](#)

[View implementation guide](#)



Building Machine Learning at the Edge Applications using Amazon SageMaker Edge Manager and IoT Greengrass v2

by Pavan Kumar Sunder and Jon Slominski

Running machine learning (ML) models at the edge can be a powerful enhancement for Internet of Things (IoT) solutions that must perform inference without a constant connection back to the cloud. Although there are numerous ways to train ML models for countless applications, effectively optimizing and deploying these models for IoT devices can present many obstacles.

Some popular questions include: How can ML models be packaged for deployment across a fleet of devices? How can an ML model be optimized for specific edge device hardware? How can we efficiently get inference feedback back to the cloud? What ML libraries do we need to install on our storage-constrained IoT devices?

In this post, we show how you can integrate [Amazon SageMaker Edge Manager](#) and [AWS IoT Greengrass](#) to build robust ML applications that are targeted specifically for edge use cases. AWS IoT Greengrass is an open-source edge runtime that we can use to build, deploy, and manage edge applications across a fleet of devices. Edge Manager can optimize and package our ML models for specific device targets, and provides an integration capability for inference in edge applications via gRPC.

Solution overview

We take a use case in the agriculture industry as an example. Today's agriculture customers are looking for solutions to monitor, track, and count livestock in the most remote areas when transferred from the nursery, weaning, growth to finish, and market locations. These solutions must be run at the edge for connectivity, latency, and cost reasons. To solve this problem, we show you how to use the [Amazon SageMaker](#) built-in object detection model and deploy it on edge devices like NVIDIA Nano, TX2, and Xavier via AWS IoT Greengrass and run inferences on them. You can then use these detections as input to tracking algorithms that help track and monitor animals.

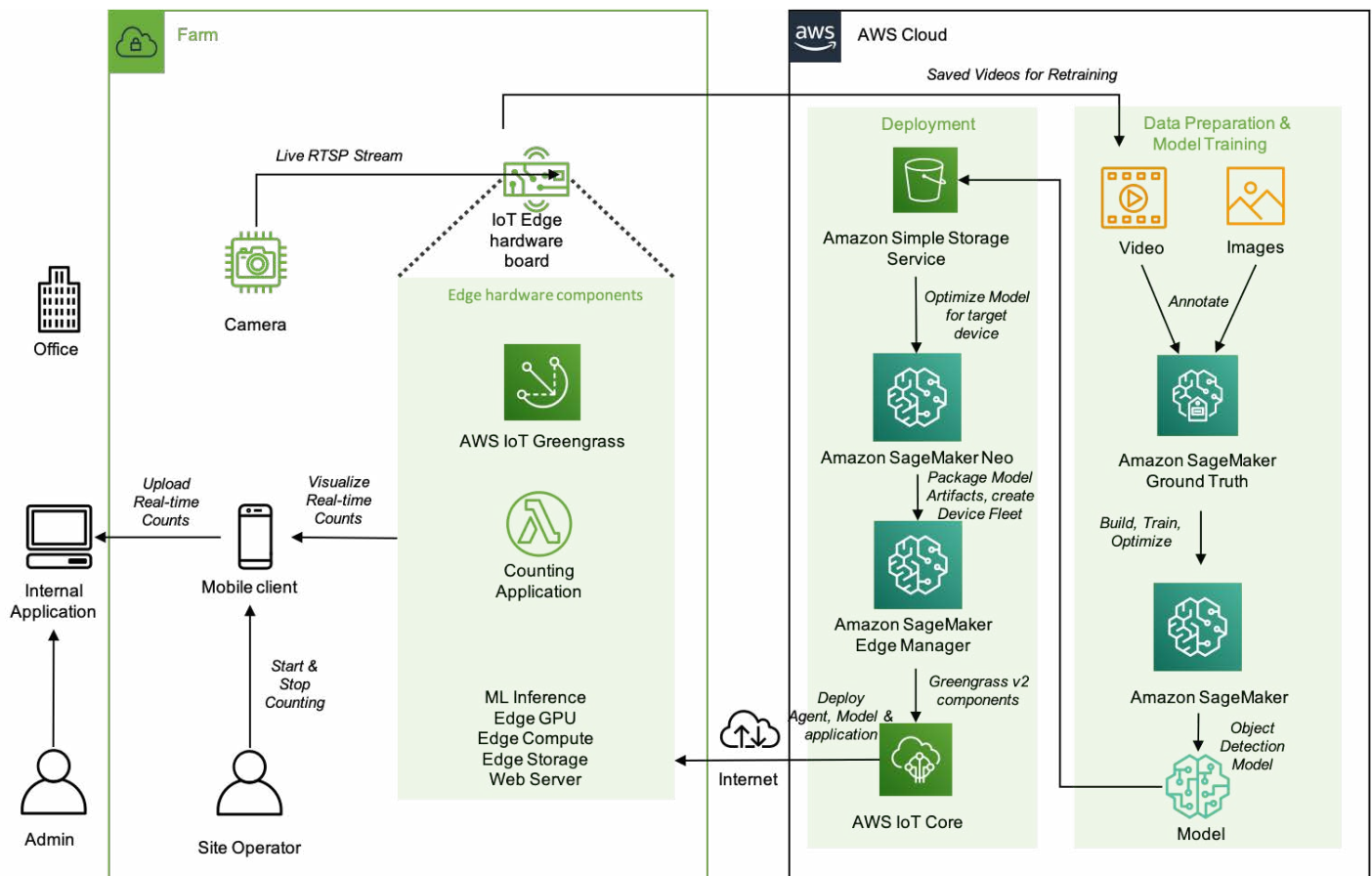
One of the applications of tracking animals is to count them. Counting pigs can be hard; they move quickly, they turn around, they all look the same! Three AWS experts tried to count pigs manually, and all three got different answers. Computer vision and ML at the edge can increase efficiency and accuracy for livestock management. The most important customer benefit is getting consistent, accurate, near-real-time livestock counts to support sound economic decisions such as feed and weight management that can optimize revenue gain or reduce revenue loss. Secondly, reducing or eliminating manual counting tasks allows workers to focus on higher-value tasks such as animal

care. This increases operational efficiency and product quality. Apart from the agriculture industry, this has applications in monitoring wildlife as well.

We look at how to set up an edge device, (in this case, a NVIDIA Jetson Xavier) with AWS IoT Greengrass. After we have trained the model, we deploy it to this device. We then run inference at the edge to count how many animals are in a given image. You can feed a live camera stream to the system, where you can use object detection outputs combined with a tracker to count animals in real time. We go over the following sections to take you through creating this application:

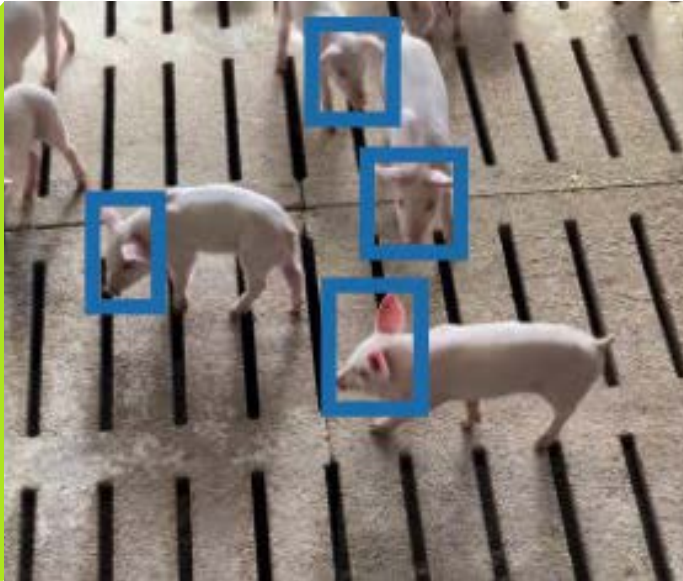
1. Prepare your dataset.
2. Use the Amazon SageMaker built-in object detection model.
3. Optimize the model for the edge device.
4. Package the model for the edge.
5. Deploy the models to the edge.
6. Build an AWS IoT Greengrass application for running inference and counting using an [AWS Lambda](#) function.
7. Set up the edge device.
8. Run the application at the edge to perform inference.

The following diagram shows a high-level architecture of the components that reside on the farm and how they interact with the AWS services in the cloud.



Prepare the dataset

For this post, we gather videos of the livestock from multiple farms with enough lighting and diverse floors. When you collect videos for training your model, use a ceiling-mounted camera that covers the whole alley where the livestock are transferred. Split those videos into frames, and use [Amazon SageMaker Ground Truth](#) to create annotations with the help of [Amazon Mechanical Turk](#). The following is an example of an annotated pigs image.



Example notebooks available for dataset creation are on the [GitHub repo](#). You can use data augmentation techniques to increase your dataset for training.

Build a livestock detection model

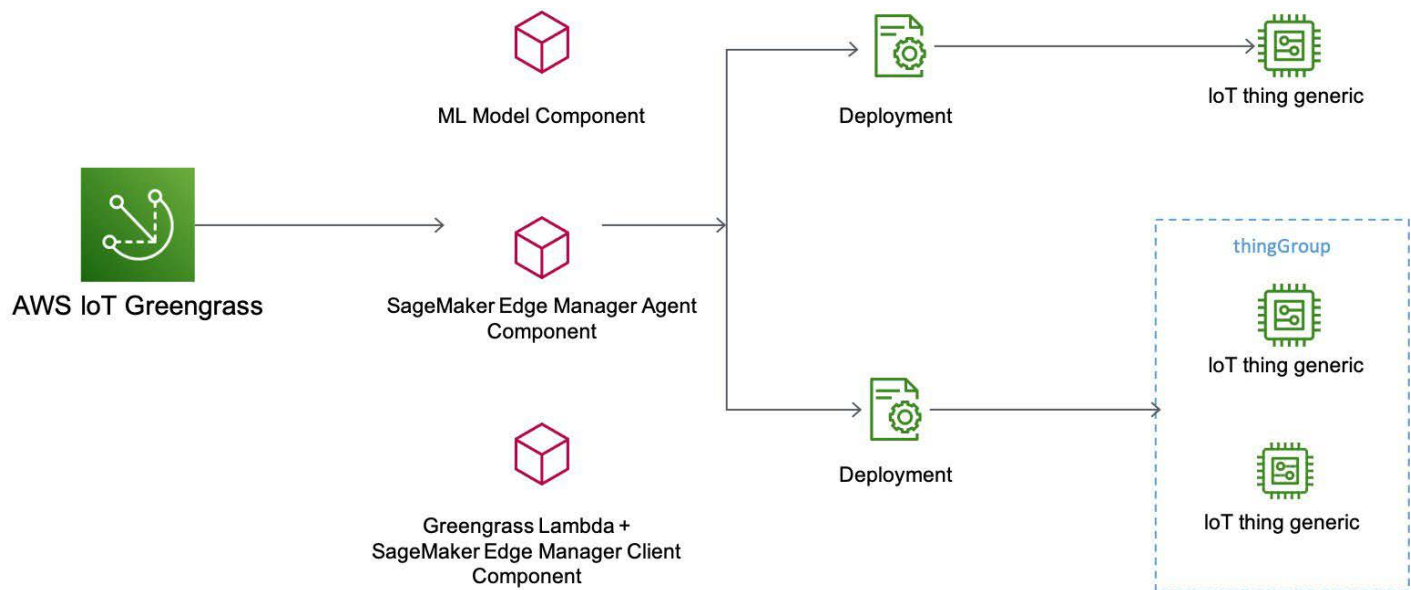
We use the SageMaker built-in object detection model and train it on a dataset of pigs. One of the biggest challenges in livestock is crowding. To make the model learn such scenes, we recommend experimenting with bounding boxes around the heads instead of entire bodies.

When we have the annotated dataset, we can use the built-in object detection model that uses

the Single Shot multibox Detector ([SSD](#)) algorithm. The following example [notebook](#) illustrates how to do this, and you can pass in your livestock dataset to create an ML model.

Optimize the model for the edge

We use [Amazon SageMaker Neo](#) to optimize the model to the target device—in this case, the Jetson Xavier. Neo automatically optimizes ML models for inference on cloud instances and edge devices to run faster with no loss in accuracy. You start with an ML model already built with DarkNet, Keras, MXNet, PyTorch, TensorFlow, TensorFlow-Lite, ONNX, or XGBoost and trained in SageMaker or anywhere else. Then you choose your target hardware platform, which can be a SageMaker hosting instance or an edge device based on processors from Ambarella, Apple, ARM, Intel, MediaTek, Nvidia, NXP, Qualcomm, RockChip, Texas Instruments, or Xilinx. With a single click, Neo optimizes the trained model and compiles it into an executable. The compiler uses an ML model to apply the performance optimizations that extract the best available performance for your model on the cloud instance or edge device. You then deploy the model as a SageMaker endpoint or on supported edge devices and start making predictions.



Deployments using Greengrass v2.0

Package the model for the edge

After you optimize the model for the edge device, we can use Edge Manager to optimize, secure, monitor, and maintain ML models on fleets of smart cameras, robots, personal computers, and mobile devices.

Edge Manager provides a software agent that runs on edge devices. The agent comes with an ML model optimized with Neo automatically, so you don't need Neo runtime installed on your devices to take advantage of the model optimizations. The agent also can collect prediction data and send a sample of the data to the cloud for monitoring, labeling, and retraining so you can keep models accurate over time. You can view all the data on the Edge Manager dashboard, which reports on the operation of deployed models.

Because Edge Manager enables you to manage models separately from the rest of the application, you can update the model and the application independently, reducing costly downtime and service disruptions. Edge Manager also cryptographically signs your models so you can verify that it wasn't tampered with as it moves from the cloud to edge devices.

Deploy the models to the edge

You can then deploy the packaged models and their business applications that use these models using AWS IoT Greengrass, an open-source IoT edge runtime and cloud service that lets you quickly and easily build intelligent device software.

AWS IoT Greengrass Version 2 is a new major version release of AWS IoT Greengrass. You can add or remove pre-built software components based on your use cases, configured specifically for your target device's CPU, GPU, and memory resources. For example, you can choose to include only prebuilt AWS IoT Greengrass components, such as stream manager, when you need to process data

streams with your application. When you want to perform ML inference locally on your devices, you can also include ML components, such as the public Edge Manager component provided by AWS, or a custom component containing your ML model. By decoupling your ML model and inference client code, you can quickly swap out different model versions without having to update application code or re-deploy your entire solution. The following [GitHub repository](#) shows an example of how to easily deploy an ML model, Edge Manager, and AWS IoT Greengrass Lambda function using AWS IoT Greengrass V2 custom components.

In this example, we deploy three components, as illustrated in the following diagram: the public Edge Manager component, a custom component that contains our Python inference client code, and another custom component wrapping our ML model.

The Edge Manager component downloads, installs, and runs an Edge Manager binary agent specific to our OS and platform architecture. When the agent is running, a gRPC-based service enables clients to manage models through a collection of APIs. With requests to the APIs, the client can load, unload, and describe models; run predictions with raw bytes or a SharedMemoryHandle of a multi-dimensional tensor array; and upload input and output tensors to the cloud. Clients can easily communicate with these APIs using the proto file available as part of the Edge Manager release artifacts.

Build an AWS IoT Greengrass V2 application using a Lambda function

To run your business logic at the edge, we package the code as an [AWS IoT Greengrass Lambda function](#) that runs indefinitely on the edge device. For example, the following [Lambda function](#) counts the number of livestock in a still image. You can also extend this to do object tracking in videos using tracking algorithms like correlation tracker, CSRT, GOTURN, KCF, and so on. See the code online: <https://amzn.to/AWS-Ag-Edge-Mgr-GG2>

Set up the edge device

For this demonstration, we use an NVIDIA Jetson Xavier NX development kit as our target edge device. To get our device communicating with AWS, we installed AWS IoT Greengrass V2 runtime [software over SSH](#), which doesn't require the device to have local AWS credentials, or the [AWS Command Line Interface \(AWS CLI\)](#) installed. Another major benefit of AWS IoT Greengrass V2 is that device provisioning to [AWS IoT Core](#) is built in.

[Read full blog post online](#)



BLOG

Processing satellite imagery with serverless architecture

by James Beswick

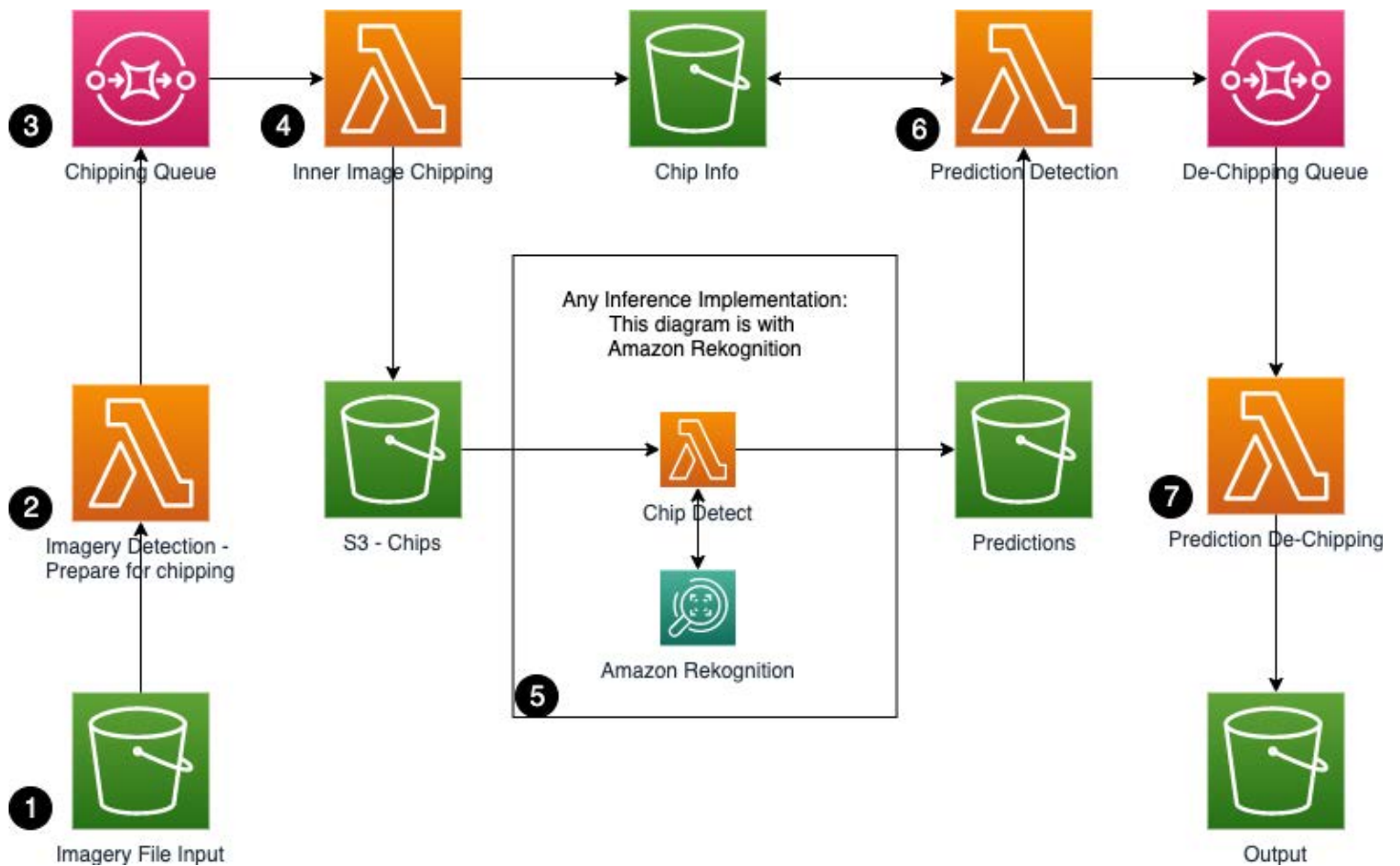
This post was written by Justin Downes, Machine Learning Consultant.

The amount of satellite imagery publicly available is growing and images from satellites tend to be large. Architectures for processing those images for machine learning must scale to meet this demand. Since many machine learning models need smaller images of a fixed size to make predictions, these images are broken into smaller sections in a process known as chipping.

This post explains a serverless approach to chipping images and sending the results to an inference engine for predictions. These predictions use the smaller sections of the

large image and must be projected back to the original image. This architecture automatically handles these projections and allows you to perform any post processing needed. This extensible architecture allows for the inclusion of additional pre- and post-processing functions, such as color correction or saving images of detected objects.

The main use case of this framework is to prepare imagery for inference pipelines. I include a reference implementation that uses [Amazon Rekognition](#) for inference. The output processing functions parse Amazon



Rekognition's specific prediction format. I note where to make changes if you use your own prediction parsing format.

While the pipeline is geared toward inference, the chipping and preprocessing functions could be used to prepare imagery for model training. Only the first half of the solution needs to be deployed (see the *Deploy with AWS SAM* section) and you can add any preprocessing functions you need.

Prerequisites

For this example, you must have an AWS account and a role with sufficient access to create resources in the following services:

- [AWS Lambda](#)
- [Amazon Simple Storage Service](#)
- [Amazon Simple Queue Service \(SQS\)](#)

1. Imagery files are stored in an S3 bucket.
2. The S3 objects invoke a Lambda function, which adds object references to a queue.
 1. A separate reference is added to the queue for each factor of parallelism that is desired. Since large images can take time to chip, this provides a mechanism to use multiple Lambda functions to work on the same image.
 2. For each factor of parallelism that you want to chip the images in, a duplicate reference is added to the queue. Each duplicate message invokes another Lambda function to chip the image. This is done in parallel by skipping over sections handled by other functions.

3. The SQS queue maintains references for imagery files.
4. The image chipping Lambda function retrieves references from the queue and chips images.
 1. Chips are written to the chip bucket.
 2. Chip information is written to the chip information bucket.
5. Inference is run on the chips and the results are written to the predictions bucket. You can use any inference engine you prefer.
6. The prediction detection Lambda function detects new predictions and retrieves the correct chip information from the chip info bucket.
 1. This Lambda function assumes that a reference to the chip information file is in the prediction.
 2. Individual predictions and chip information are paired and put onto the de-chipping queue.
7. Prediction de-chipping Lambda function converts chip coordinates back into original imagery coordinate space and puts results in the output bucket.

Deploying with AWS SAM

Download the code used in this post from this [GitHub repo](#). To deploy only the chipping portion of the architecture, comment out the following sections in the template.yaml file: DeChipper, PredictionDetection, and DeChippingQueue.

This example includes an [AWS Serverless Application Model \(AWS SAM\)](#) template to make it easier to deploy to your own

account. You can reuse components for your own solution. This approach promotes loose coupling and integrates with other components.

To deploy this solution, you must have the [AWS SAM CLI](#) installed. After cloning the [code repository](#):

1. Navigate to the repository:

```
cd Imagery-processing-blog
```
2. Build the AWS SAM application:

```
sam build
```
3. Deploy the AWS SAM application:

```
sam deploy --guided
```

The Lambda function that uses Amazon Rekognition for inference is commented out to reduce the risk of unintended charges. To use Amazon Rekognition, uncomment the section "RekognitionInference" in the template.yaml file. To use your own inference endpoint, keep this section commented and deploy a separate Lambda function to read image chips from the chip bucket.

Implementation details

This serverless design is scalable with no need to provision infrastructure and allows you to control the throughput of images. SQS controls the throughput of artifacts that are persisted in the system. The following sections describe the components in the solution and explain how they can be configured to fit your needs.

S3 buckets

The user interacts with some S3 buckets to drop initial imagery into the pipeline, retrieve image chips, and retrieve final prediction output.

Other buckets are used for internal processing. They stage information for further processing in the application. The chip info bucket stores information about each image chip so that predictions can be translated back into the original, larger, image's pixel space.

To do much of the post-processing of a prediction, information about where the original image chip came from must be stored somewhere. In this solution that information is stored in an S3 bucket, the ChipInfo bucket. If another solution is used to store chip information, such as DynamoDB, then care must be taken such that a given prediction can be matched to the correct record containing the chip's information. The predictions bucket contains the output of our inference model, in this case Amazon Rekognition's object detection.

SQS queues

The SQS queues decouple functions in the pre- and post-processing of the imagery. You can extend this by adding additional queues that are processed by other functions. The Lambda functions that precede each queue can publish to multiple queues. Or you can chain multiple queues together with Lambda functions processing data between them. This allows parallel processing of information or processing of information that needs to be sequential.

The chipping queue contains references to images that must be chipped. The chipping Lambda function pulls messages off the chipping queue and partitions that image. To chip an image in parallel, each chipping function must know which parts of the image to save and which to skip. This is enabled by having multiple messages for a single image on the chipping queue where each message tells the chipping function how to chip the larger image.

The second queue, the de-chipping queue, takes the results of merging the predictions and chip information messages and stages them for any post processing. This implementation projects those predictions back into the original image space but one could add any other functions they choose.

Lambda functions

Detection Lambda function: This is triggered by a new image being put into the imagery S3 bucket. For each new object, it detects an entry is pushed on to the SQS queue specified by the OUTPUT_QUEUE parameter and if the value for DUPLICATE_CHIPPERS is greater than 1 then that many entries are pushed onto the queue. This is done so that the chipper Lambda function can be instantiated multiple times to chip the image in parallel.

Chipping Lambda function: This processes larger images into smaller chips. The size of the smaller chip and the stride that the chipping function takes between chips are customizable through environment variables. This function will also skip alternating chips if there are parallel chipping functions. The chip itself will be saved to an S3 bucket and the information about the chip (original image, where the chip was taken, etc.) will be written to a different S3 bucket.

Prediction Lambda function: This detects predictions that have been made on the chips produced by the chipping Lambda function. It then merges these predictions with the chip's information contained in the chip information bucket. The example provided with this post shows an implementation based on Amazon Rekognition's object detection output, but this could easily be modified to parse any other inference output. After merging, this Lambda function then takes this information and pushes each item onto an SQS queue.

De-chipping Lambda function: This Lambda function processes messages from the PREDICTION_QUEUE, performs processing, and stores the results in an S3 bucket. In this implementation, the post processing projects the prediction coordinates from the chip's pixel space back into the original, larger image's pixel space. Other functions that could be applied here would be to chip out each detected object or to forward to some follow-on ML model.

Conclusion

This post shows how to deploy an imagery processing pipeline in the AWS Cloud. It is decoupled to allow both pre and post-processing extensions to be integrated into the pipeline more easily. Visit the [code repository](#) for further information.

This architecture uses [Amazon Rekognition](#). To use your own inference engine, such as a [SageMaker endpoint](#), you must modify a couple of functions. The chip detection Lambda function must be modified to send images to your endpoint instead of Amazon Rekognition. The Lambda function that detects the predictions must parse the output of your endpoint correctly to match the chip information files in the chip info bucket.

For more serverless learning resources, visit [Serverless Land](#).

[Read blog post online](#)



VIDEOS

The Future of Fresh Fruit Harvest: A Talk with FFRobotics

Robotic harvesting of high value fruits is a high tech solution to the challenges of orchard and vineyard growers around the globe. Join us as we speak to FFRobotics about how they view AI/ML at the edge, and how robotics are making us think differently about computer vision.



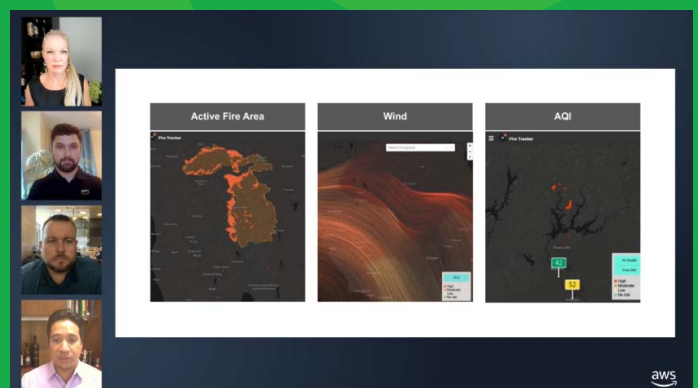
Seeing is Believing - the Rise of Data & Analytics in Agriculture

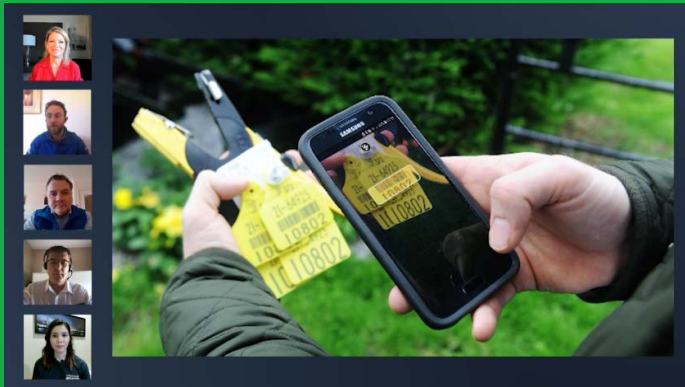
What would you do if you could scale computer vision almost infinitely? What questions could you start to answer if you had a thousand eyes instead of two to your farm, your orchard, your processing facility? In this session, learn how machines are scaling human vision to take on some of the largest challenges in agriculture today.



The Season for Vintner Innovation: Viticulture & More with E.J. Gallo Winery

It's always the right season for celebrating diversity in Agriculture! In this episode, our industry expert hosts Karen Hildebrand and SA John Marciniak raise a glass to innovation with leaders from E.J. Gallo Winery: Ryan Barr and Robert Barrios. Join us as they share some amazing insights about viticulture and their work with AWS ML Labs and SAP Migration.





Farming Innovation in Ireland: A Talk with Herdwatch

How many data points are collected on milk before it gets to your glass? Find out in this episode, as our AWS host Karen Hildebrand and special guests from Herdwatch discuss building the #1 farm management app in Ireland. Herdwatch is scaling globally on AWS, and we'll go behind the scenes to see how they are driving change and innovation.



Voice Activated Agriculture

Hands-free interfaces have special appeal in the field, where conditions may not be suitable to fumbling fingers and devices. In this session, hear how innovators are using voice and audio to help growers and farmers get on with their jobs with minimum hassle.



Amazon Prime - Clarkson's Farm Season 1

An intense, arduous and frequently hilarious year in the life of Britain's most unlikely farmer, Jeremy Clarkson. Join Jeremy and his rag-tag band of agricultural associates as they face-up to a backdrop of unhelpful weather, disobedient animals, unresponsive crops and an unexpected pandemic. This is Jeremy Clarkson as you've never seen him before.