SAMANTHA SERVO, KAYNE RODRIGO,
& JON BONSO

AWS CERTIFIED

MACHINE LEARNING ASSOCIATE

MLA-C01 EXAM

Tutorials Dojo Study Guide

TABLE OF CONTENTS

INTRODUCTION

Today's world is experiencing technological innovations daily, particularly in AI and ML, where businesses are integrating different fields. This transformation has been made easier due to the flexibility and scalability of the Cloud. AWS, the leading company in cloud-implemented technologies, has AI and ML services that aim to assist companies in maximizing the use of the available tools and technologies. The AWS Machine Learning Associate certification MLA-C01 is designed for individuals who want to demonstrate their skills in building and deploying machine learning applications on the Cloud. This certification serves as proof of expertise in data preparation, the application of machine learning algorithms, and the deployment of models on cloud infrastructure. Additionally, it highlights proficiency in implementing CI/CD pipelines, monitoring models and infrastructure, and ensuring the security of systems in a machine-learning environment.

As companies increasingly adopt artificial intelligence, there is a growing demand for professionals skilled in machine learning to develop and maintain advanced systems. Earning the AWS Certified Machine Learning Engineer – Associate certification identifies you as a qualified practitioner capable of helping organizations implement machine learning technologies on a larger scale. This certification showcases your skills and opens up better job opportunities as employers seek cloud-based machine learning expertise. Accomplishing a certification in machine learning on AWS is a difficult venture to undertake. However, it brings a plethora of benefits. The AWS Certified Machine Learning Engineer – Associate certification is a stepping stone toward more excellent machine learning and artificial intelligence achievement. For those aspiring to be data scientists, machine learning engineers, or hold any leadership role surrounding AI, this certification becomes essential in achieving and paving pathways to success.

This Study Guide eBook has been crafted to walk you through the preparation for the AWS Certified Machine Learning Engineer – Associate (MLA-C01) certification. It contains the most essential concepts, examination domains, sample questions, and other materials valuable for your success. We shall start by providing an overview of the exam, including the domains, question types, scoring criteria, and benefits of passing the examination. The content of this guide corresponds with the official MLA-C01 exam guide, which includes every essential subtopic in each exam domain. Together, we will tackle the details of technical ML modeling, data transformation, implementation, and system watching, which are the components of AWS machine learning solutions.

Note: We recommend using this study guide alongside hands-on experience and practice exams to reinforce your knowledge and prepare for the exam. These resources will help you assess your strengths, identify areas for improvement, and build confidence. This comprehensive study plan will equip you for the exam and open up future opportunities in machine learning.

AWS CERTIFIED MACHINE LEARNING ENGINEER ASSOCIATE (MLA-C01) OVERVIEW

Amazon Web Services began its Global Certification Program in 2013 with the primary purpose of validating the technical skills and knowledge of IT Professionals in building secure and reliable cloud-based applications using the AWS Cloud. In April 2013, AWS launched its first-ever AWS Certification test called the AWS Certified Solutions Architect Associate exam. This was followed by the AWS Certified SysOps Administrator and AWS Certified Developer Associate exams.

Amazon has been continuously expanding and updating its certification program year after year. They launched a series of Professional and Specialty-level certifications that cover various topics like DevOps, machine learning, data analytics, advanced networking, and many others. As the number of AWS services increases, a new and updated version of the AWS certification exam is released regularly to reflect the recent service changes and include the new knowledge areas.

In August 2024, AWS launched the AWS Certified Machine Learning Associate (MLA-C01) exam, an associate-level certification for individuals who are familiar with AI/ML technologies on AWS. The AWS Certified Machine Learning - Associate (MLA-C01) exam is designed for individuals with at least one year of experience using AWS services like Amazon SageMaker and other ML engineering tools. It evaluates the ability to build, deploy, and operationalize ML solutions within the AWS Cloud. This certification is ideal for roles like backend software developers, DevOps engineers, data engineers, and MLOps engineers.

The exam consists of 65 questions and has a duration of 130 minutes. It includes a combination of multiple-choice, multiple-response questions, ordering, matching type, and case study questions. The exam cost is 150 USD, and it can be taken either at a local Pearson VUE testing center or online with remote proctoring.

This certification validates the skills needed to implement ML workloads in production and to manage their lifecycle. It helps professionals stand out in the competitive job market and positions them for in-demand roles in machine learning, a field expected to grow significantly.

By earning the AWS Certified Machine Learning Engineer – Associate credential, you demonstrate expertise in key areas, such as data ingestion and preparation, model training and deployment, and using AWS services for continuous integration and delivery (CI/CD). It also demonstrates your ability to monitor and secure machine learning systems effectively.

Individuals who unfortunately did not pass the AWS exam must wait for 14 days before they are allowed to retake the exam. There is no hard limit on the number of exam attempts, so you can try again and again until you pass the exam. Take note that on each attempt, the full registration price of the exam must be paid.

Your AWS Certification Account will have a record of your complete exam results within 5 business days of completing your exam. The score report contains a table of your performance for each exam domain, which indicates whether you met the competency level required for these domains or not. AWS uses a compensatory scoring model, which means that you do not necessarily need to pass each and every individual section.

You will pass this exam as long as you get an overall score of 720 across 4 domains. Each section has a specific score weighting that translates to the number of questions; hence, some sections have more questions than others. Your Score Performance table highlights your strengths and weaknesses that you need to improve on.

Exam Details

The AWS Certified Machine Learning Engineer – Associate (MLA-C01) exam is designed for individuals who can effectively demonstrate a comprehensive understanding of machine learning (ML) concepts, model development, deployment, and operationalization using AWS services and tools. This certification is ideal for professionals involved in building, deploying, and managing ML solutions within the AWS Cloud, regardless of their specific job role.

The exam includes various question types, such as multiple-choice questions, where you select one correct response out of four options, and multiple-response questions, where you choose two or more correct responses out of five or more options. Additionally, it features ordering questions that require arranging responses in the correct order to complete a specified task, matching questions that involve pairing responses to a set of prompts correctly, and case study questions based on a scenario with two or more related questions. Each question is evaluated separately, and you receive credit for each correct response. You have the flexibility to take the exam either via online proctoring or at a testing center near you.

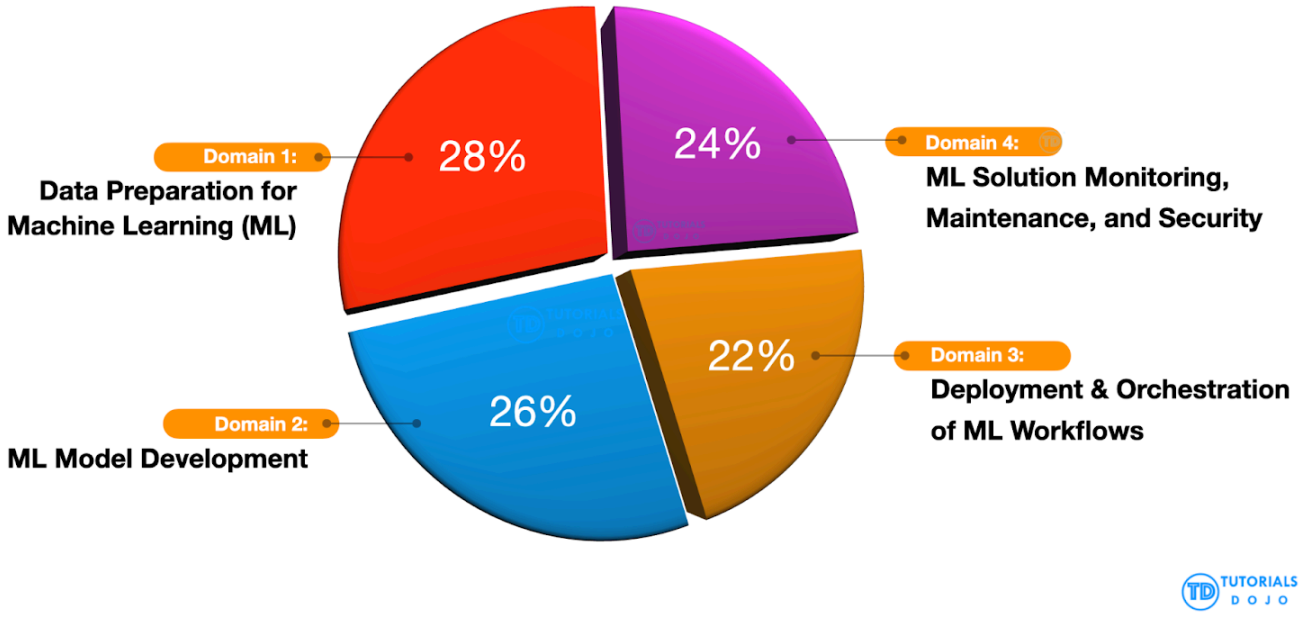
Exam Code:	MLA-C01
Prerequisites:	None
No. of Questions:	65
Score Range:	100-1000
Cost:	150 USD
Passing Score:	720
Time Limit:	130 minutes

Exam Domains

The AWS Certified Machine Learning Engineer Associate (MLA-C01) exam has four different domains, each with a corresponding weight and topic coverage. The domains are as follows:

- Domain 1: Data Preparation for Machine Learning (ML) (28%)
- Domain 2: ML Model Development (26%)
- Domain 3: Deployment and Orchestration of ML Workflows (22%)
- Domain 4: ML Solution Monitoring, Maintenance, and Security (24%)

One exam domain consists of several task statements. A task statement is a sub-category of the exam domain that contains the required cloud concepts, knowledge, and skills for you to accomplish a particular task or activity in AWS. In the AWS Certified Machine Learning Associate (MLA-C01) test, the **Domain 1: Data Preparation for Machine Learning** has the biggest weighting in the exam at 28%, so expect to see a lot of data preparation for ML-related scenarios in the exam. Conversely, the exam domain with the least exam weighting is **Domain 3: Deployment and Orchestration of ML Workflows**, so you have to limit the time you spend studying under this knowledge area.



Domain 1: Data Preparation for Machine Learning (ML)

1.1. Ingest and store data.
1.2. Transform data and perform feature engineering.
1.3. Ensure data integrity and prepare data for modeling.

Domain 2: ML Model Development

2.1. Choose a modeling approach.
2.2. Train and refine models.
2.3. Analyze model performance.

Domain 3: Deployment and Orchestration of ML Workflows

3.1. Select deployment infrastructure based on existing architecture and requirements.
3.2. Create and script infrastructure based on existing architecture and requirements.
3.3. Describe the training and fine-tuning process for foundation models.
3.4. Use automated orchestration tools to set up continuous integration and continuous delivery (CI/CD) pipelines.

Domain 4: ML Solution Monitoring, Maintenance, and Security

4.1. Monitor model inference.
4.2. Monitor and optimize infrastructure and costs.
4.3. Secure AWS resources.

Exam-Related AWS Topics and Services

The official exam guide contains a list of key tools, technologies, and concepts that may show up on the Machine Learning Associate MLA-C01 test. Keep in mind that this is just a non-exhaustive list of the tools and technologies that may or may not appear on the exam. This list can change at any time and is primarily given to test-takers to help them understand the general scope of services, features, or technologies for this certification. In addition, the general tools and technologies in this list appear in no particular order.

Here are the topics, AWS services, and concepts that you should focus on for your upcoming exam. You have to review your knowledge of the following:

• AI/ML • AI/ML Use Cases and Applications • Data formats and ingestion mechanisms • AWS data sources	• Amazon Translate • Amazon Comprehend • Amazon Lex • Amazon Polly

• Amazon SageMaker Data Wrangler • SageMaker Feature Store • Data cleaning and transformation techniques • Feature engineering techniques • Encoding techniques • Data annotation • Validating and labeling data • Pre-training bias metrics • Data classification • Anonymization • Data masking • Identifying and mitigating sources of bias • ML algorithms • SageMaker built-in algorithms • Foundation models • Model sizing • Regularization techniques	• Amazon Bedrock • Amazon SageMaker AI • Amazon SageMaker JumpStart • Amazon SageMaker Clarify • Amazon SageMaker Data Wrangler • Amazon SageMaker Model Monitor • Amazon SageMaker Feature Store • Amazon Augmented AI [Amazon A2I] • Amazon SageMaker Model Cards • Amazon Macie • Hyperparameter tuning techniques • Overfitting • Underfitting • Containerization concepts • Continuous Integration and Continuous Delivery (CI/CD) pipelines • Version control systems and basic usage

Remember that among the four exam domains, **Data Preparation for Machine Learning (ML)** holds the largest weight, accounting for 28% of the exam. This means that more than a quarter of the questions in the AWS Certified Machine Learning Associate (MLA-C01) exam will focus on the application of foundational models in machine learning. The Appendix section of the exam guide provides a list of relevant AWS services for each domain, and it's important to review these services thoroughly as they are key to your preparation.

In the Analytics domain, you should focus on services such as Amazon Athena, Amazon Kinesis, Amazon Data Firehose, Amazon EMR, AWS Glue, AWS Glue DataBrew, AWS Glue Data Quality, AWS Lake Formation, Amazon Managed Service for Apache Flink, Amazon OpenSearch Service, Amazon QuickSight, and Amazon Redshift. These services are fundamental to understanding how to handle and process data for machine learning applications.

For the Application Integration, Cloud Financial Management, Compute, and Containers domains, make sure you're familiar with Amazon EventBridge, Amazon Managed Workflows for Apache Airflow (Amazon MWAA), Amazon Simple Notification Service (Amazon SNS), Amazon Simple Queue Service (Amazon SQS), AWS Step Functions, AWS Billing and Cost Management, AWS Budgets, AWS Cost Explorer, AWS Batch, Amazon EC2, AWS Lambda, AWS Serverless Application Repository, Amazon Elastic Container Registry (Amazon ECR), Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS).

These tools and services are essential for integrating applications and managing compute resources in a cloud environment.

In the Databases category, focus on Amazon DocumentDB (with MongoDB compatibility), Amazon DynamoDB, Amazon ElastiCache, Amazon Neptune, and Amazon RDS. Understanding how these services are used to store, manage, and query data is crucial for implementing machine learning solutions in AWS.

For the Developer Tools section, study services like AWS Cloud Development Kit (AWS CDK), AWS CodeArtifact, AWS CodeBuild, AWS CodeDeploy, AWS CodePipeline, and AWS X-Ray. These tools help in automating and streamlining the software development process, which is key in building and maintaining machine learning workflows.

The Machine Learning section is at the core of the exam and includes essential services like Amazon SageMaker, Amazon Comprehend, Amazon Rekognition, Amazon Polly, Amazon Lex, Amazon Textract, Amazon Transcribe, Amazon Translate, Amazon Fraud Detector, Amazon Personalize, Amazon Kendra, Amazon Lookout for Equipment, Amazon Lookout for Vision, Amazon Bedrock, Amazon Mechanical Turk, Amazon Augmented AI (Amazon A2I), Amazon CodeGuru, and AWS HealthLake. These services are critical for building and operationalizing machine learning models within the AWS Cloud.

For the Management and Governance domain, make sure to familiarize yourself with AWS Auto Scaling, AWS CloudFormation, AWS CloudTrail, Amazon CloudWatch, Amazon CloudWatch Logs, AWS Compute Optimizer, AWS Config, AWS Organizations, AWS Service Catalog, AWS Systems Manager, and AWS Trusted Advisor. These tools help in managing and monitoring cloud resources, ensuring your machine learning models are running efficiently and securely.

In the Media, Migration, and Transfer, Networking, and Content Delivery categories, focus on Amazon Kinesis Video Streams, AWS DataSync, Amazon API Gateway, Amazon CloudFront, AWS Direct Connect, and Amazon VPC. These services are important for handling data transfer and networking, which are essential for the smooth functioning of machine learning workflows.

For the Security, Identity, and Compliance category, you need to study services like AWS Identity and Access Management (IAM), AWS Key Management Service (AWS KMS), Amazon Macie, and AWS Secrets Manager. These tools help secure your machine learning systems and protect sensitive data while ensuring compliance with various security standards.

Finally, for the Storage category, ensure you review Amazon Elastic Block Store (Amazon EBS), Amazon Elastic File System (Amazon EFS), Amazon FSx, Amazon S3, Amazon S3 Glacier, and AWS Storage Gateway. These

storage services play a crucial role in managing and storing large datasets needed for machine learning applications.

Exam Scoring System

You can get a score from 100 to 1,000 with a minimum passing score of **720** when you take the AWS Certified Machine Learning Associate exam. AWS uses a scaled scoring model to associate scores across multiple exam types that may have different levels of difficulty. Your complete score report will be sent to you by email 1 - 5 business days after your exam. However, as soon as you finish your exam, you'll immediately see a pass or fail notification on the testing screen.

For individuals who unfortunately do not pass their exams, you must wait 14 days before you are allowed to retake the exam. There is no hard limit on the number of attempts you can retake an exam. Once you pass, you'll receive various benefits, such as a discount coupon, which you can use for your next AWS exam.

Once you receive your score report via email, the result should also be saved in your AWS Certification account already. The score report contains a table of your performance of each domain and it will indicate whether you have met the level of competency required for these domains. Take note that you do not need to achieve competency in all domains to pass the exam. At the end of the report, there will be a score performance table that highlights your strengths and weaknesses, which will help you determine the areas you need to improve on.

		Score Perfomance	
Section	**% of Scored Items**	**Need Improvements**	**Meet Competencies**
Domain 1.0: Data Preparation for Machine Learning (ML)	28%		
Domain 2.0: ML Model Development	26%		
Domain 3.0: Deployment and Orchestration of ML Workflows	22%		
Domain 4.0: ML Solution Monitoring, Maintenance, and Security	24%		

Exam Benefits

If you successfully pass any AWS exam, you will be eligible for the following benefits:

- **Exam Discount** - You'll get a 50% discount voucher that you can apply for your recertification or any other exam you plan to pursue. To access your discount voucher code, go to the "Benefits" section of your AWS Certification Account, and apply the voucher when you register for your next exam.

- **Certification Digital Badges** - You can showcase your achievements to your colleagues and employers with digital badges on your email signatures, LinkedIn profile, or your social media accounts. You can also show your Digital Badge to gain exclusive access to Certification Lounges at AWS re:Invent, regional Appreciation Receptions, and select AWS Summit events. To view your badges, simply go to the "Digital Badges" section of your AWS Certification Account.

You can visit the official AWS Certification FAQ page to view the frequently asked questions about getting AWS Certified and other information about the AWS Certification: https://aws.amazon.com/certification/faqs/.

AWS CERTIFIED MACHINE LEARNING ASSOCIATE MLA-C01 EXAM STUDY GUIDE

The AWS Certified Machine Learning Engineer – Associate (MLA-C01) exam is designed to assess an individual's ability to build, deploy, and support machine learning applications using AWS services. It covers various topics: data collection, model development, hyperparameter optimization, and model performance evaluation. Candidates are expected to demonstrate their skills in deploying machine learning models, constructing CI/CD pipelines, and scaling AWS infrastructure for machine learning applications. The exam emphasizes practical skills to manage the end-to-end machine learning workflow, utilizing services such as Amazon SageMaker AI, AWS Lambda, Amazon EC2, and AWS Glue to address real-world business challenges.

This certification is ideal for professionals working as machine learning engineers, data scientists, or those building machine learning models on AWS. While deep programming expertise is not required, it is essential to have a strong understanding of machine learning principles, data handling, and AWS services. The MLA-C01 exam prepares candidates for technical roles in machine learning and artificial intelligence, offering opportunities to work on advanced projects and demonstrate expertise in AWS's robust machine-learning ecosystem. This certification can significantly enhance career prospects and provide a pathway to more specialized roles within the AI/ML field. You can view the complete details and guidelines for the certification exam here.

What to review

1. AWS AI and ML Services

Familiarizing yourself with AWS services for artificial intelligence (AI) and machine learning (ML) is essential for building and managing AI/ML solutions. Key services include tools for data preparation, model building, training, deployment, and inference. Understanding AWS infrastructure, such as Amazon EC2, Amazon S3, AWS Lambda, and Amazon SageMaker AI, is crucial for effective ML workflows. Additionally, be aware of the AWS Shared Responsibility Model for security and compliance and the importance of AWS Identity and Access Management (IAM) for secure resource access. Finally, familiarize yourself with AWS Regions, Availability Zones, Edge Locations, and pricing models to efficiently manage costs and scalability in ML solutions.

2. Fundamentals of Artificial Intelligence (AI) and Machine Learning (ML)

It is vital to grasp the foundational concepts of artificial intelligence and machine learning. You should understand the distinctions between supervised learning, unsupervised learning, and reinforcement learning and the algorithms commonly used in each type. Supervised learning is typically used for classification and

regression tasks, while unsupervised learning is often applied in clustering and dimensionality reduction. Reinforcement learning, on the other hand, is used for training models to make sequences of decisions. You should also familiarize yourself with commonly used algorithms such as decision trees, support vector machines, k-means clustering, and neural networks. Understanding how these algorithms are applied in various domains, from fraud detection to recommendation systems, will help you select the appropriate algorithm for different problems.

3. Data Wrangling and Preprocessing

Data preparation is a critical step in the machine learning pipeline, and mastering it is essential for successful model development. You need to understand how to clean, transform, and process raw data into a usable form. This includes handling missing values, scaling numerical data, encoding categorical features, and managing imbalanced datasets. Tools like Amazon SageMaker Data Wrangler, AWS Glue, and Amazon Redshift are designed to simplify and automate data wrangling tasks, helping you prepare your data efficiently. Proper data preprocessing can significantly impact the performance and accuracy of machine learning models, making it a crucial area of focus for exam preparation.

4. Model Training and Evaluation

You should have a clear understanding of how to train machine learning models and how to select the correct algorithm for the specific problem you are addressing. This involves understanding the principles behind different algorithms and how they work in other contexts. You will also need to know how to tune hyperparameters to optimize model performance using grid or random search techniques. Evaluating model performance is also a key skill. You must understand how to use metrics like accuracy, precision, recall, F1 score, and confusion matrix to assess a model's performance. These evaluation metrics will help you determine your model's performance and guide decisions on model selection and improvement.

5. Deployment and Operationalization

Deploying machine learning models into real-world environments is a critical aspect of ML. You should be familiar with deploying models using services like Amazon SageMaker, which allows you to deploy models for real-time predictions or batch processing. After deployment, monitoring the model's performance is equally important. You need to understand how to monitor metrics such as latency, throughput, and resource utilization to ensure that the deployed model meets performance expectations. Additionally, knowing how to scale models to handle varying amounts of data and how to update and maintain models over time is key for ensuring their ongoing effectiveness. Tools like SageMaker Pipelines and SageMaker Model Monitor will help automate these tasks.

6. Security and Compliance in ML

Security is fundamental when working with machine learning, especially when dealing with sensitive or personal data. You need to understand how to secure data in transit and at rest using encryption tools such as AWS KMS (Key Management Service). Understanding AWS's compliance programs, such as GDPR, HIPAA, and SOC 2, will help you ensure that machine learning models comply with the relevant regulatory requirements. Additionally, managing access to resources securely using IAM (Identity and Access Management) is crucial for protecting data and models in a multi-user environment.

7. Ethical AI and Responsible AI Practices

As AI and ML solutions become more pervasive, understanding the ethical considerations in their development and deployment is essential. Ethical AI practices include ensuring fairness in machine learning models, preventing bias in training data, and making models explainable. Fairness can be achieved by identifying and addressing potential biases in the data or the model's predictions. Additionally, ensuring transparency and interpretability in AI models is vital, especially in regulated industries.

8. Real-World Applications of AI/ML

Exploring real-world case studies will provide practical insights into how AI and ML technologies are applied in various industries. Healthcare, for example, leverages machine learning for predictive diagnostics, while retail uses it for personalized product recommendations and demand forecasting. ML is used in fraud detection, credit scoring, and algorithmic trading in finance. By understanding these applications, you will improve your ability to design and deploy machine learning models and better understand how these technologies solve complex problems across various sectors.

How to review

As with any exam, the very first step is always the same - **KNOWING WHAT TO STUDY**. Although we have already enumerated them in the previous section, I highly suggest you go over the AWS Certified Machine Learning Engineer - Associate (MLA-C01) Exam Guide again and see the exam contents.

AWS already has a vast number of free resources available for you to prepare for the exam. I suggest you first read the Overview of Amazon Web Services whitepaper, and gain a good understanding of the different AWS concepts and services. Check out the amazing Tutorials Dojo cheat sheets to supplement your review for this section. Also, check out this article: **Top 5 FREE AWS Review Materials**.

Beyond understanding the core AWS services, it's essential to dive into AWS's AI/ML offerings. Familiarize yourself with services like Amazon SageMaker AI, Amazon Bedrock, Amazon Rekognition, Amazon Lex, Amazon Polly, and many more, as these are crucial for the exam.

AWS also offers a comprehensive digital course on Machine Learning, consisting of a collection of free courses designed to enhance your understanding and skills. This collection includes:

- AWS Machine Learning Services Overview
- AWS DeepRacer
- Data Science
- Math for Machine Learning
- MLS-C01 Exam Readiness

These courses not only cover the MLS-C01 exam topics but also provide valuable insights for the AWS Certified Machine Learning Engineer – Associate (MLA-C01) and AWS Certified AI Practitioner (AIF-C01) certifications.

Common Exam Scenarios

Scenario	Solution
Domain 1: Data Preparation for Machine Learning (ML)	
A machine learning engineer should choose a combination of AWS services to ingest and process real-time transaction data before delivering it to SageMaker for inference.	• Amazon Kinesis Data Streams • Amazon Managed Service for Apache Flink
A manufacturing company is creating a customer segmentation model using Amazon SageMaker with a dataset stored in an S3 bucket. This dataset contains nominal categorical variables like customer region and product preferences, along with numerical variables such as purchase frequency and total spending. The ML engineer must preprocess the data for effective analysis and model training in SageMaker.	Amazon SageMaker Data Wrangler
Extract and combine data from Amazon RDS databases, Amazon DynamoDB, and other data sources to create a unified dataset for training the machine learning model.	AWS Glue
A company is developing a machine learning (ML) platform for sentiment analysis of news articles. The platform needs storage to process large volumes of text data. The ML platform must handle concurrent access from multiple EC2 instances for data preprocessing, model training, and inference.	Amazon EFS

A file format that is best suited for effective real-time parsing and analysis with diverse dataset.	JSON Lines
A manufacturing company has an unstructured dataset with customer purchase information containing duplicate records for attributes like name, email, and address. They need to apply machine learning techniques to remove duplicates with minimal manual intervention.	AWS Glue FindMatches transform
Domain 2: ML Model Development	
A healthcare company seeks to enhance patient outcome predictions using generative AI applications. The company requires a solution that allows selection from various predictive models, guarantees the confidentiality of private data during model fine-tuning, and eliminates the need for managing the underlying ML infrastructure.	Amazon Bedrock
A machine learning engineer is working on a model to predict the likelihood of specific crop diseases in a large agricultural field. To build this model, the engineer is using Amazon SageMaker's built-in linear learner algorithm for classification, specifically configuring the `predictor_type` hyperparameter to `multiclass_classifier`. The goal is to minimize the number of false positives in the predictions.	Adjust the `target_precision` hyperparameter to a higher value.
A company aims to utilize Amazon Bedrock for its AI-driven projects. They require a tool that can develop advanced conversational chatbots to interact with customers and provide relevant information from their data systems.	Develop a chatbots that provide relevant information to customers.

It is a metric that helps ML engineer to fine-tune the model's ability to detect fraud while minimizing false positives to ensure genuine transactions are not impeded.	• False Positive Rate • F1 Score
A machine learning (ML) engineer is tasked with building a model using Amazon SageMaker AI. This model will analyze sales transaction data to categorize customers into different behavior-based groups. The engineer needs to test various algorithms and ensure that the model runs effectively while tracking the results for evaluation.	Implement SageMaker's built-in algorithms for training the model and leverage SageMaker Experiments to automatically track and analyze the results.
A company is exploring generative AI to enhance its customer service chatbot. The company aims to fine-tune models from third-party services to enhance the chatbot's comprehension of business-specific customer inquiries.	• Offers a wide range of pre-trained foundation models (FMs) from multiple providers. • Amazon Bedrock enables private customization of foundation models (FMs).
Domain 3: Deployment and Orchestration of ML Workflows	
It is a service that helps automate and manage workflows, ensuring seamless integration with other AWS services, from data preparation to model deployment.	Amazon SageMaker Pipelines
An AWS service that helps companies orchestrate continuous integration and continuous deployment (CI/CD) pipelines.	AWS CodePipeline
It's a deployment method designed to handle occasional data spikes, minimize deployment costs, and ensure scalable, efficient processing of inference requests.	Use Amazon SageMaker Asynchronous Inference.

It is a deployment strategy that rolls out the new model to a small percentage of transactions while the majority continue to be processed by the old model.	Canary deployment
A tech company builds ML prediction models and stores a large, regularly updated dataset in Amazon S3. The goal is to automate model creation and updates, including applying NLP transformations whenever new data is uploaded.	Create an Amazon SageMaker Pipeline to handle the processing workflow. Then, set up Amazon EventBridge to trigger the pipeline execution whenever new data is added to the S3 bucket.
A logistics company uses a 2 GB custom ML model hosted on-premises to forecast delivery times. They need to migrate the model to AWS to reduce operational overhead and use a fully managed service. The model experiences unpredictable traffic, with peaks during business hours and low activity at night and weekends. The solution must scale automatically based on demand and be cost-effective during idle periods.	Deploy the model as a serverless SageMaker AI endpoint without Provisioned Concurrency.
It is the most cost-effective EC2 instance purchasing option for a machine learning training job that runs once a month and can tolerate interruptions.	Spot Instances
Domain 4: ML Solution Monitoring, Maintenance, and Security	
It assists the company in utilizing machine learning to analyze log data from various AWS resources.	Amazon CloudWatch Logs Insights
A retail company deployed an ML-based recommendation system on an Amazon SageMaker endpoint, aiming to monitor input data and model predictions for anomalies while reducing operational overhead.	Deploy SageMaker Model Monitor to continuously monitor the input data and model predictions for anomalies.

An IAM component that is appropriate for managing access control in the AWS environment where users are assigned permissions directly to AWS resources like Amazon SageMaker, S3, and AWS Glue.	IAM Roles with Identity-Based Policies
A company requires a solution for SQL-based data querying and interactive dashboards to visualize the results. Additionally, the solution should incorporate machine learning for trend forecasting.	Use Amazon Athena to query the data in Amazon S3, integrate with Amazon QuickSight for interactive dashboards, and leverage Amazon QuickSight ML Insights for forecasting trends.
It is an Amazon SageMaker AI feature can be used to monitor machine learning models for data and prediction anomalies.	Amazon SageMaker Model Monitor
A team of ML engineer needs to identify and address the root cause of inaccuracies to improve the model's performance.	Run a SageMaker Clarify processing job to detect potential biases in the training data.
It is an AWS service that can assist the company in logging API activities across its AWS environment for compliance purposes.	AWS CloudTrail

Validate Your Knowledge

Once you feel confident with your review, it's a great idea to test your knowledge through sample exams. **Tutorials Dojo** offers a highly recommended set of practice tests for the Machine Learning Engineer Associate exam that can help you prepare effectively. These tests feature a wide range of unique questions, giving you the opportunity to check whether you've overlooked any key topics that might appear on your exam. For a thorough preparation, you can pair these practice exams with this study guide eBook.

If you've performed well on the **Tutorials Dojo AWS Certified Machine Learning Engineer Associate Practice Tests** and feel ready, go ahead and pursue your certification with confidence. However, if you find certain topics more difficult, take some extra time to review them and focus on the hints within the questions to help guide you toward the correct answers. If you're still unsure about passing, it might be wise to reschedule your exam and give yourself more time to prepare. Ultimately, the time and effort you invest in your preparation will surely pay off.

Sample Practice Test Questions:

Question 1

An ML engineer is creating a hyperparameter tuning job in Amazon SageMaker using Bayesian optimization. The ML engineer wants to ensure the reproducibility of the results across multiple runs of the tuning job.

Which approach would meet the requirements?

1. Use warm start to run a hyperparameter tuning job based on the results of the initial job.
2. Set a random seed and use it during hyperparameter generation.
3. Use the MaxParallelTrainingJobs field to limit the number of training jobs.
4. Use Random Search instead of Bayesian Optimization for the tuning job.

Correct Answer: 2

Amazon SageMaker hyperparameter tuning uses either a Bayesian optimization or random search strategy to efficiently find the best values for hyperparameters. These processes often involve randomness in selecting initial parameter values and during iterative optimization steps.

Setting a random seed ensures that the tuning job generates the same sequence of random numbers across multiple runs, leading to consistent and reproducible results. Without a random seed, the job generates different sequences each time, causing variations in the search process and final results, even with identical configurations.

Hence, the correct answer is: **Set a random seed and use it during hyperparameter generation.**

The option that says: **Use warm start to run a hyperparameter tuning job based on the results of the initial job** is incorrect. Although this approach would enable a tuning job to reuse the results of a prior job, it does not inherently ensure reproducibility since it depends on the original job's results and can vary due to randomness in hyperparameter generation.

The option that says: **Use the MaxParallelTrainingJobs field to limit the number of training jobs** is incorrect because this field only controls resource usage and does not address the randomness of hyperparameter generation or the tuning algorithm's internal behavior.

The option that says: **Use Random Search instead of Bayesian Optimization for the tuning job** is incorrect. Switching to Random Search won't automatically guarantee reproducibility. Like Bayesian Optimization, Random Search relies on random number generation to select hyperparameters. Reproducibility with Random Search can only be achieved by explicitly setting a random seed.

References:

https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-considerations.html#automatic-model-tuning-random-seed

https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html#automatic-tuning-bayesian-optimization

Check out this Amazon SageMaker Cheat Sheet:

https://tutorialsdojo.com/amazon-sagemaker/

Question 2

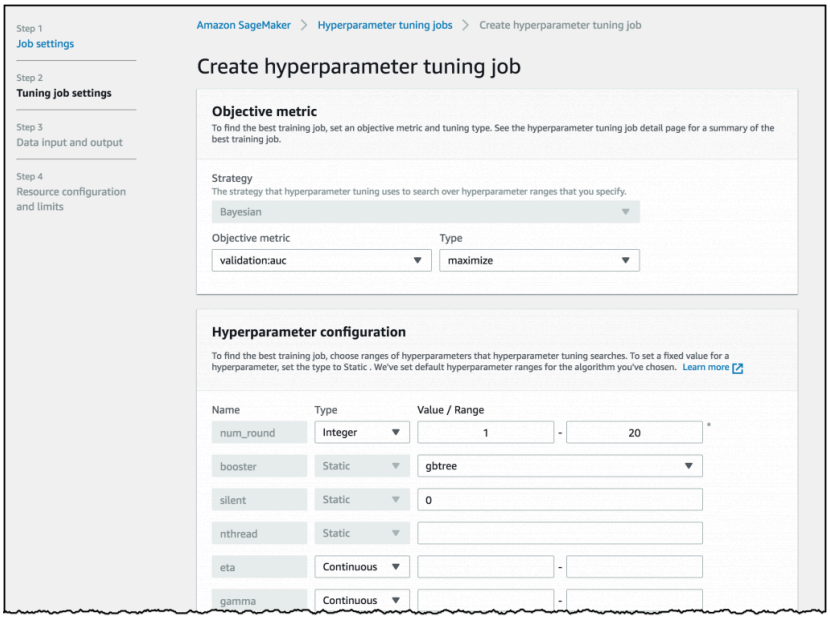
An ML engineer is training a K-Means clustering model on Amazon SageMaker. The model will be used to segment customers based on purchasing behavior. The engineer aims to experiment with multiple hyperparameter ranges to find the best version of the model.

How can the engineer achieve this with the least effort possible?

1. Use Amazon SageMaker Automatic Model Tuning (AMT) to perform hyperparameter optimization (HPO).
2. Train the model using Amazon SageMaker Canvas in hyperparameter optimization (HPO) mode.
3. Manually run parallel SageMaker training jobs with different hyperparameter combinations.
4. Use Amazon SageMaker Canvas in Ensembling mode to tune the model.

Correct Answers: 1

Amazon SageMaker's Automatic model tuning (AMT), also known as hyperparameter tuning, finds the best version of a model by running many jobs that test a range of hyperparameters on your dataset. You choose the tunable hyperparameters, a range of values for each, and an objective metric. You choose the objective metric from the metrics that the algorithm computes. Automatic model tuning searches the hyperparameters chosen to find the combination of values that result in the model that optimizes the objective metric.



The Amazon SageMaker k-means algorithm is an unsupervised algorithm that groups data into clusters whose members are as similar as possible. Because it is unsupervised, it doesn't use a validation dataset that hyperparameters can optimize against. But it does take a test dataset and emits metrics that depend on the squared distance between the data points and the final cluster centroids at the end of each training run. To find the model that reports the tightest clusters on the test dataset, you can use a hyperparameter tuning job. The clusters optimize the similarity of their members.

Hence, the correct answer is: **Use Amazon SageMaker Automatic Model Tuning (AMT) to perform hyperparameter optimization (HPO)**

The option that says: **Train the model using Amazon SageMaker AutoPilot in hyperparameter optimization (HPO) mode** is incorrect. AutoPilot is designed for supervised learning tasks (e.g., classification and regression). It does not natively support unsupervised algorithms like K-Means clustering. Thus, this option is not applicable.

The option that says: **Manually run parallel SageMaker training jobs with different hyperparameter combinations** is incorrect. While this approach is feasible, it requires significant manual effort to manage and execute the training jobs, making it inefficient compared to using AMT.

The option that says: **Use Amazon SageMaker AutoPilot in Ensembling mode to tune the model** is incorrect. Similar to the AutoPilot's HPO mode, Ensembling is not applicable to unsupervised learning tasks like K-Means clustering. Since K-Means does not involve labeled data or prediction tasks, AutoPilot in Ensembling mode cannot address the goal of hyperparameter tuning for the K-Means model.

References:
https://docs.aws.amazon.com/sagemaker/latest/dg/k-means-tuning.html
https://docs.aws.amazon.com/sagemaker/latest/dg/algo-kmeans-tech-notes.html

Check out this Amazon SageMaker Cheat Sheet:
https://tutorialsdojo.com/amazon-sagemaker/

Click **here** for more **AWS Certified Machine Learning Engineer Associate exam questions**.

What to expect from the exam

There are five types of questions on the examination:

- **Multiple-choice:** Includes one correct response and three incorrect options (distractors).
- **Multiple-response:** Requires selecting two or more correct answers from a list of five or more options. All correct responses must be chosen to receive credit for the question.
- **Ordering:** Involves arranging 3–5 responses in the proper order to accomplish a given task. Credit is awarded only if all responses are selected and ordered correctly.
- **Matching:** Requires pairing a list of responses with 3–7 prompts. All pairs must be matched accurately to earn credit for the question.
- **Case study:** Features a single scenario with two or more related questions. The same scenario applies to all questions in the case study. Each question is evaluated independently, and credit is given for each correct answer.

Distractors, or incorrect answers, are response options that an examinee with incomplete knowledge or skill would likely choose. However, they are generally plausible responses that fit in the content area defined by the test objective.

Unanswered questions are scored as incorrect; there is no penalty for guessing.

Majority of questions are usually scenario-based. Some will ask you to identify a specific service or concept. While others will ask you to select multiple responses that fit the given requirements. No matter the style of the question, as long as you understand what is being asked, then you will do fine.

Your examination may include 15 unscored items that are placed on the test by AWS to gather statistical information. These items are not identified on the form and do not affect your score.

The AWS Certified Machine Learning Engineer - Associate (MLA-C01) examination is a pass or fail exam. Your results for the examination are reported as a scaled score from 100 through 1000, with a minimum passing score of 720.

A few more tips:

1. Be sure to get proper sleep the night before, and don't be lazy in preparing for the exam. If you feel that you aren't ready enough, you can just reschedule your exam.
2. Come early to the exam venue so that you have time to handle mishaps if there are any.
3. Read the exam questions properly, but don't spend too much time on a question you don't know the answer to. You can always go back to it after you answer the rest.

4. Keep your reviewer if you plan on taking other AWS certifications in the future. It will be handy for sure.
5. And be sure to visit the Tutorials Dojo website to see our latest AWS reviewers, cheat sheets and other guides.

DATA PREPARATION FOR MACHINE LEARNING (ML)

Ingest and Store Data

Transform Data and Perform Feature Engineering

Ensure Data Integrity and Prepare Data for Modeling

Optimizing Data for Machine Learning Models

A. Ingest and Store Data

Chapter 1.1 Data Formats and Ingestion Mechanisms

Validated formats

Validated formats mean that the data is prepared, cleaned, and verified for consistency and correctness before entering a system or ML model. This usually involves structured data formats that follow a specific plan which makes sure that all data fields are correct and follow set rules.

Examples of validated formats

- JSON (JavaScript Object Notation)
- Parquet
- CSV (Comma-Separated Values)

Importance of validated formats

- Error prevention
- Data Integrity
- Model Reliability

Example scenario

In the Philippines, an online shopping business gathers information from web purchases. They use a checked CSV format with each row having the correct structure (e.g., product ID, amount, cost, and buyer ID). The business uses tools such as AWS Glue or Amazon SageMaker Data Wrangler to check and tidy the data before feeding it into the ML model.

Non-validated formats

On the other hand, non-validated formats mean that data has not gone through any checking or cleaning before entering the system. The data might have errors, missing values, incorrect data types, or unnecessary details. These types of data can cause problems if used in an ML pipeline directly.

Examples of non-validated formats

- Plain text
- Unstructured JSON

- Log files

Risks of using non-validated formats

- Inconsistent results
- Higher processing overhead
- Data loss

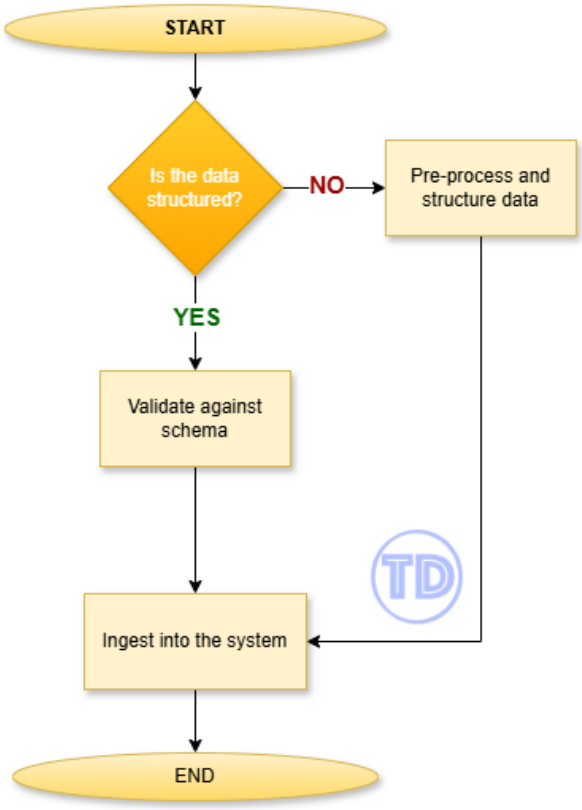
Example scenario

A transport company in Manila gathers GPS details from its delivery vehicles in an unstructured log format. The logs may include incomplete data or mistakes, such as missing timestamps or wrong coordinates. Without validation, this data will result in wrong analysis or model forecasts regarding delivery schedules or traffic patterns.

Common Data Formats

- **Apache Parquet** - A column based data file format. It gives powerful compression and encoding methods to manage complex data in large quantities and works with many programming languages and analytics tools.
- **JSON (JavaScript Object Notation)** - It is a light and simple text format which people find easy to read and write. Same with machines that find it easy to understand and create. People use it to share data between servers and web applications.
- **CSV (Comma-Separated Values)** - It is one of the simplest data formats for showing table-like information. Each line in the CSV file shows one record (columns split by commas or other signs like semicolons).
- **Apache ORC (Optimized Row Columnar)** - It is another column based way to store data and offers fast performance for tasks that involve a lot of reading. It helps make things run faster and saves space, which is especially useful when dealing with huge amounts of data.
- **Apache Avro** - It is a compact, fast, and binary data serialization format. People often use it to share information between different systems, particularly in setups where events trigger actions and in Kafka data pipelines.
- **RecordIO** - It is a data format made by Apache MXNet mainly for deep learning uses. It is built to store and move big datasets efficiently, which is perfect for training models.

Choosing the right format



You can always refer to the flowchart above to remind you when choosing the right format for your use case.

The right format is essential to gather useful insights for decision making. This is applicable in data engineering, machine learning, and big data processing. The following factors are what you should consider when choosing a data format.

Factors to consider

- **Data Type and Structure** - Select a format depending on the data type (structured, semi-structured or unstructured). Formats such as JSON manage mixed data types well.
    - **Example:** A BPO firm in the Philippines uses JSON for call logs which contain both structured details (like call duration) and unstructured notes. JSON works well because it deals with both organized and unorganized information.

- **Performance Considerations** - Choose file types that make searching fast and work well with how you often look at data, such as the column based formats for studying numbers.
    - **Example:** A transport business in Manila uses Apache Parquet to study GPS information, allowing quick searches to check how deliveries are doing.
- **Data Volume** - For big data collections, pick formats such as Parquet or ORC to manage compression and effective processing on a large scale.
    - **Example:** A retail chain in the Philippines uses Apache ORC to squeeze and handle billions of loyalty program transactions.
- **Storage Efficiency** - Choose formats that use less storage, especially for keeping data for a long time.
    - **Example:** A bank in the Philippines uses Apache Avro to store ATM transaction logs tightly while keeping the data structure adaptable.
- **Compatibility with Tools and Frameworks** - Pick a format that fits well with your analytics and ML tools.
    - **Example:** An online shopping startup in Makati uses JSON to easily connect AWS Lambda functions with DynamoDB.
- **Schema Flexibility** - Formats with schema evolution abilities to help adapt to changes in data structure over time.
    - **Example:** A tech company in the Philippines records chatbot interactions using Avro. This approach lets them include new details like sentiment scores without disrupting processes.
- **Data Lineage and Auditing** - Focus on formats that help check data structure and keep data steady for rules and tracking.
    - **Example:** A Philippine government office uses Apache Avro to maintain consistent data, ensuring compliance and tracking history.

Use cases to consider

- **Analytical Queries on Large Datasets**
    - **Format:** Parquet or ORC
    - **Why:** These formats work well with big datasets because they provide columnar storage, allowing you to read only specific columns, not the whole dataset. This feature makes them ideal for data storage or analysis tasks, like using Amazon Redshift or AWS Athena.

- **Example:** A telecom company in the Philippines looking at call records from lots of users picks Apache Parquet to keep the data. The columnar style lets them swiftly reach certain columns (like call length, place) without going through the whole dataset, making queries faster.

- **Real-Time Data Streaming**
  - **Format:** Apache Avro or RecordIO
  - **Why:** The Apache Avro is popular in real time data tasks like those with Apache Kafka as it helps with data structure and makes data saving more efficient. RecordIO on the other hand, is made for deep learning as it provides fast data retrieval during training.
  - **Example:** A bank in the Philippines is processing live transaction data from ATMs and mobile banking apps. They use **Apache Avro** to handle the real-time data stream because Avro supports efficient serialization, ensuring fast data transfer and minimal delay while maintaining a schema for data consistency. It is also compatible with **Apache Kafka**, which is used for stream processing.

- **Web API or Data Exchange Between Services**
  - **Format:** JSON
  - **Why:** JSON is light, easy to read and simple to use. It is perfect for sharing data between web apps or APIs.
  - **Example:** A Philippine online shopping company uses JSON to swap data between its website and backend tools. JSON's easy style, clear format and ability to handle layered structures make it perfect for sharing customer orders, product details and payment info through APIs.

- **Data Import/Export from Spreadsheets or Relational Databases**
  - **Format:** CSV
  - **Why:** CSV is a basic and widely accepted format that is ideal for small to medium sized data collections. It helps when people need to move data in or out of systems like relational databases such as MySQL and PostgreSQL or spreadsheets.
  - **Example:** A Filipino store business seeking to bring sales records from their POS systems to Amazon S3 uses CSV files for data sharing. The information includes basic table details (product IDs, amounts and prices) making CSV a great option because it works well with spreadsheet programs like Excel and database systems like MySQL.

- **Logging and Event Data**
  - **Format:** JSON or Apache Avro

- **Why:** JSON works well due to its flexibility and ease of parsing. Apache Avro fits streaming and logging needs when speed and small size matter for data serialization.
- **Example:** In the Philippines, a cloud hosting company tracks system events and user activities. They keep log files in JSON style which helps them gather detailed, structured information like event type, time and user ID.

Chapter 1.2 AWS Data Sources for Data Storage

Amazon S3: Object storage for ML data

Amazon S3

It is a scalable storage service for objects which is made to store and retrieve any amount of data at low latency and high availability. This service allows the following features:

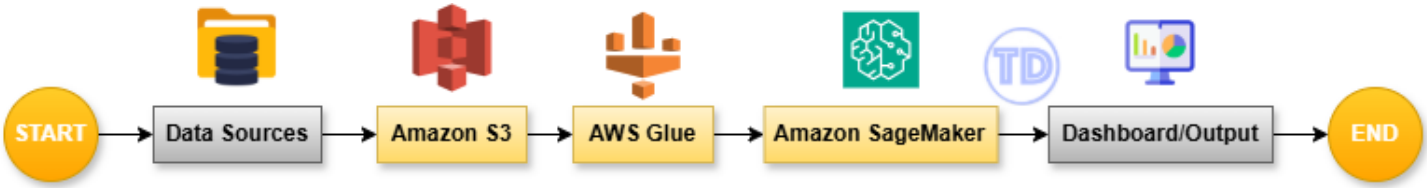
- **Can store diverse data types:** JSON, Parquet CSV, Avro, and etc.
- Designed for availability
- Works with ML services:

Key Features for ML Workflows

- **Query Capabilities: S3 Storage Classes:** Pick from S3 Standard Intelligent-Tiering or S3 Glacier to save money.
- **Versioning:** Keep old versions of datasets in case you want to recreate experiments accurately.
- **Query Capabilities:** Use Amazon Athena to search data in S3 directly which does not need a different database.

Example application

A logistics company in Metro Manila analyzes customer delivery data used for route optimization. All the delivery records, GPS data, and order details are stored in S3. Then, they will analyze the data using SageMaker to develop an optimized routing algorithm.



The flowchart above demonstrates the example application process from the data source to output. It is described as follows:

- **Data Sources:** Delivery records, GPS data, and order details.
- **Amazon S3:** The data is stored here.
- **AWS Glue:** The data is cleaned and transformed in this process.
- **Amazon Sagemaker:** The cleaned data will be used to train and deploy the optimized routing algorithm.
- **Dashboard/Output:** The optimized routes are shown and presented here.

Amazon Elastic File System (Amazon EFS): Scalable file storage

Amazon EFS

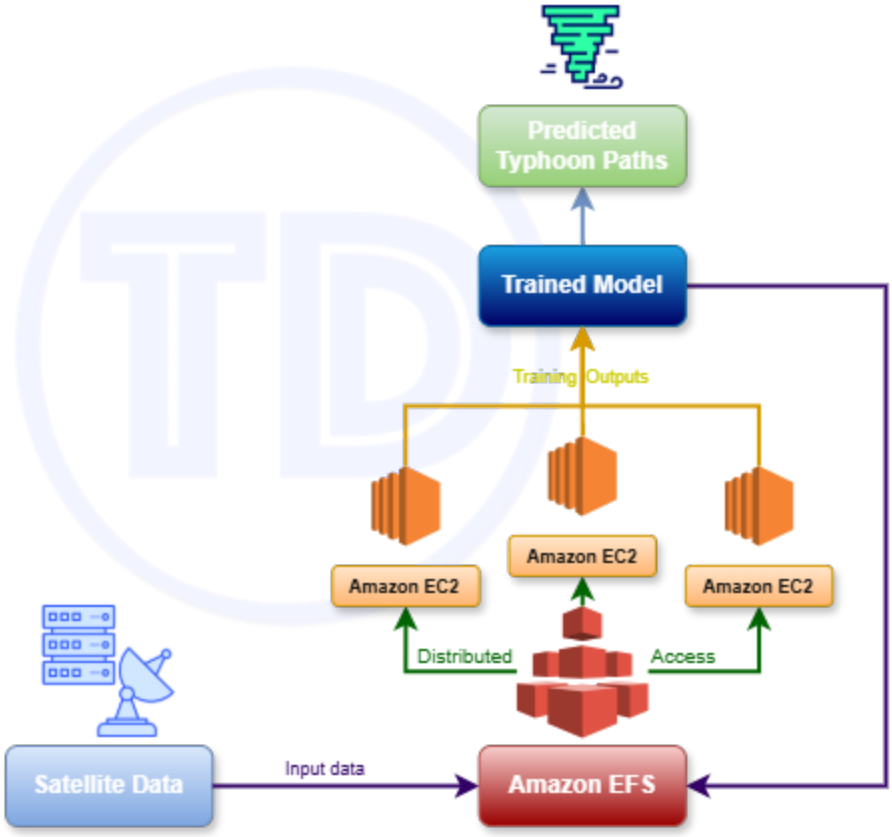
It provides a simple, scalable, and elastic file storage for AWS and on site resources. It's good for apps needing shared file access and low-latency operations.

Key Features for ML Workflows

- **POSIX-Compliant Storage:** Works with apps needing file level permissions.
- **Scalability:** Automatically expands and shrinks when data is added or taken away.
- **Parallel Access:** Gives simultaneous access to thousands of Amazon EC2 instances.

Example application

A research team under a Quezon City university uses EFS to store and share their collated satellite imagery data for an ML model predicting typhoon paths.



The graph above visualizes how the research team uses Amazon EFS as a central repository of their satellite data. Simultaneous training of multiple EC2 instances for the typhoon prediction model are also applied.

Amazon FSx for NetApp ONTAP: High-performance file storage

This service manages file storage completely built on ONTAP software. It mixes NetApp's fast file systems with AWS's large scale capacity.

Key Features for ML Workflows

- **Data Compression and Deduplication:** Cuts storage expenses for big datasets.
- **Integrated Data Management:** Snapshots replication features, which can be used for version control, backups, etc.
- **Hybrid Workloads:** Can integrate with on-premises NetApp systems.

Example application

In Cebu, a BPO company uses FSx for NetApp ONTAP. It analyzes and stores voice recordings for training an ML-based call sentiment analysis system.

The flowchart above shows how the BPO company analyzes their voice recordings. Amazon FSx for NetApp ONTAP delivers fully managed shared storage in the AWS Cloud along with well known data access plus management features of ONTAP.

Chapter 1.3 AWS Services Involved in Data Ingestion

Below are the AWS services you must know for data ingestion. This will help you be familiarized with the necessary services for your ML workflow.

Amazon Kendra: Data ingestion with semantic search

Amazon Kendra
This service is useful for data ingestion through semantic search, particularly for unstructured data sources.

AWS Glue: Complex ETL tasks

AWS Glue
It can be used for data cataloging and schema management when ingesting structured data.

AWS Glue DataBrew: Data preparation

AWS Glue DataBrew
This service is for data cleaning, normalization, and preparing data for analysis.

SageMaker Data Wrangler: Pre-modeling data activities

SageMaker Data Wrangler

This service is helpful for collecting, transforming, and preparing data for modeling.

Amazon Kinesis: Real-time data streaming service

Amazon Kinesis

It helps you gather, handle, and study live streaming data for ML apps. It suits situations needing very fast data ingestion and processing. Keyword here is, *"real-time"*.

Key Features for ML Workflows

- **Amazon Kinesis Data Streams (KDS):** For ingesting and storing real-time data right away. This handles gigabytes every second.
- **Amazon Data Firehose:** Puts streaming data into big storage places, warehouses, and tools for checking data like Amazon S3, Redshift or Elasticsearch automatically.
- **Kinesis Data Analytics:** Allows SQL-based questions to process streaming data instantly.

Example application

A ride-hailing company in Manila uses Amazon Kinesis to track the live location and trip details of their drivers. They utilize this information (by feeding it to an ML model) to improve dispatching and reduce customer wait times.

Apache Flink: Stream processing at scale

Apache Flink

It is a free open-source system for handling data streams supported by Amazon Managed Service. It can quickly manage big data flows with very fast processing and minimal delay.

Key Features for ML Workflows

- **Stateful Stream Processing:** Tracks data states for advanced ML tasks like sessionization.
- **Event Time Processing:** Handles data using the time an event happened, not when it was received.

- **Fully Managed Service:** AWS controls Flink infrastructure, cutting down the amount of operational work needed.

Example application

In Makati City, a stock trading app uses Apache Flink to handle stock price data streams quickly. It changes and combines the data before giving it to a prediction model in SageMaker.

Apache Kafka: Distributed event streaming platform

Apache Kafka

A free, open-source tool used for sending events. Many businesses utilize this for handling high-throughput, low-latency event streaming. You can utilize AWS with Kafka using Amazon Managed Streaming for Apache Kafka (Amazon MSK).

Key Features for ML Workflows

- **Event Stream Storage:** Keeps data for replay, which supports both batch and streaming tasks.
- **Partitioning and Replication**: Provides very high availability and fault tolerance.
- **Seamless Integration:** Connects easily with AWS services like Lambda Glue etc.

Example application

There is a delivery service company based in Cebu that uses Apache Kafka when keeping track of different parcel status updates and delivery events. These are all processed in real-time, training ML models to predict delays or even optimize delivery routes.

A comparison table can be seen below for quick reference:

Comparison Table			
Feature	**Amazon Kinesis**	**Apache Flink**	**Apache Kafka**
Primary use case	Real-time data gathering, handling, plus examination	Stateful stream processing plus event driven data pipelines	Messaging systems are distributed with event streams
Core functionality	Handling data in real time with scalable	Framework for handling data with quick stream	Pub/sub model for log data and event data

	methods	changes	
Integration with AWS services	Connected closely with AWS tools like Lambda SageMaker along with DynamoDB	Connected to AWS through Amazon Managed Service for Apache Flink	Accessible on AWS through Amazon Managed Streaming for Apache Kafka (MSK)

Chapter 1.4 AWS Storage Options: Use Cases and Tradeoffs

Below is a quick reference to compare the different AWS storage options: S3, EFS, and FSx.

Comparison Table			
Storage Option	**S3**	**EFS**	**FSx**
Primary use case	Object storage for static files plus backups along with big data analysis and archives.	Large file storage allows shared access plus applications needing NFS.	Fast file storage for specific tasks like Windows apps or HPC systems.
Core functionality	Long-lasting and scalable object storage supports version control along with lifecycle management.	A fully managed elastic file system allows shared access across many EC2 instances.	Managed file systems designed specifically for Windows (FSx for Windows File Server) or HPC tasks (FSx for Lustre)
Integration with AWS services	Many AWS services work together e.g., SageMaker along with Athena Redshift plus Lambda.	Connected with EC2 plus ECS along with SageMaker via NFS	Close connection with EC2 for HPC plus Windows-based tasks
Cost	Pay-as-you-go prices depend on how much storage you use and how you access it (Standard Glacier or Intelligent	Prices depend on storage type (Standard or Infrequent Access) and speed requirements.	Prices differ by FSx type with costs for storage, throughput plus data transfer.

	Tier).		
Performance tradeoffs	High strength plus growth potential; eventual agreement for overwrite PUTS/DELETES	Provides quick response times along with steady data flow, yet has more delay compared to nearby storage.	Fast data processing with quick response times for HPC plus tasks needing lots of input/output; tuned for very good performance
Data structure	Object-based storage places data in buckets and uses keys for organization.	A hierarchical file system has folders plus files, accessible through NFS.	Hierarchical file systems work very well for certain needs like HPC (e.g. Lustre) plus SMB (e.g. Windows).
Scaling needs	The system automatically scales to manage any data size without maximum limits.	Automatically adjusts to meet capacity needs; limited to 10 GB/s throughput in Standard class.	Scales rely on workload needs; FSx for Lustre handles hundreds of GB/s throughput.

Chapter 1.5 Data Ingestion Using AWS Tools

Here you will see the different roles of AWS Tools in the process of data ingestion.

Extracting data

Amazon S3

- **Use case application**
  - A chain of malls in the Philippines relies on Amazon S3 to store sales records in CSV format. The transactions get uploaded each day to an S3 bucket. These files help analyze market patterns like for example, a precise tracking of weekend sales trends in Metro Manila malls.

- **Best practices**
  - Amazon Athena enables SQL-like queries on CSV files directly in the S3 bucket, providing a scalable solution without the need to extract or download the entire dataset.

Amazon EBS

- **Use case application**
  - Delivery companies in Manila keep their live delivery records in Amazon EBS storage units. A later analysis of these logs helps them make better delivery paths in metro areas like Manila as well as Cebu.
- **Best practices**
  - Use Amazon DataSync to move logs from EBS to Amazon S3 for later batch tasks plus long-term storage.

Amazon RDS

- **Use case application**
  - A local microfinance institution relies on Amazon RDS to keep loan application info. Loan officers in Quezon City plus Davao branches check these records to assess applicants next to making loan decisions.
- **Best practices**
  - Use AWS Database Migration Service to relocate financial records into RDS and your business operations stay online with no interruptions.

Amazon DynamoDB

- **Use case application**
  - An e commerce site like Shopee stores order details plus payment records in DynamoDB. The data logs help track real time metrics as well as provide insights for millions of customers across the nation.
- **Best practices**
  - Turn on DynamoDB Streams to record any immediate changes in tables plus let AWS Lambda handle these updates for data analysis as well as alerts.

AWS services for acceleration and IOPS optimization

Amazon S3 Transfer Acceleration

It makes data transfers quicker by sending files through Amazon CloudFront's edge spots across the globe. A

key feature proves very useful for quick uploads plus works best when users need to send data over far distances.

IOPS

Also known as “Input/Output Operations per Second”. A database is fine-tuned to reduce IOPS, which can also help you reduce costs.

- **Example application**
  A logistics company with sites in Cebu and Davao needs to send customer data to an S3 bucket in Singapore. The S3 Transfer Acceleration speeds up data delivery via edge locations, which helps the upload process work better despite the distance.

EBS Provisioned IOPS

The feature executes numerous fast I/O operations every second plus serves databases as well as real time apps that need quick response times.

- **Example application**
  A smart city initiative in Manila includes sensors that track traffic flow and air quality in real time. The collected data flows into EC2 instances along with EBS storage that's fast due to Provisioned IOPS tech.

Ingesting data

Amazon SageMaker Data Wrangler

The tool makes data preparation easier for machine learning which also offers a user friendly interface to clean and analyze data with minimal coding requirements.

- **Example application**
  A sari sari store chain records daily sales data from its transactions plus deals with input errors in sales figures (like when PHP 50.00 appears as PHP 5000 due to missed decimal points). The team uses SageMaker Data Wrangler to fix and prepare the information before they start model training.

Amazon SageMaker Feature Store

A feature store acts as a central data hub that stores, shares, and manages machine learning features. The system works with real time predictions through online stores as well as batch operations via offline storage.

- **Example application**
  A credit card company depends on SageMaker Feature Store to track how customers act, such as how often they visit Jollibee with their monthly purchases at sari sari stores. The data assists multiple fraud detection models to work more accurately.

Chapter 1.6 Merging and Processing Data from Multiple Sources

Merging and processing data from multiple sources forms an important part of ML dataset preparation. A combination of input data from different sources makes datasets richer along with better feature representation as well as improved model results.

Techniques for Data Merging

When merging data, it often involves aligning datasets based on their common keys or columns, resulting in a unified dataset. The common tools you can use include:

- **Python (Pandas)**
  - **Best For:** Small to medium-sized datasets.
  - **Usage:** Perfect for local and in memory data merge tasks, just like when you combine sales data next to customer info from CSV files.
- **PySpark**
  - **Best For:** For large-scale datasets, which need distributed processing.
  - **Usage:** For handling datasets stored in cloud environments such as Amazon S3 or RDS.

AWS Glue: ETL (Extract, Transform, Load) Operations

AWS Glue

This service streamlines the ETL (Extract Transform Load) operations while also automating the data processing tasks. It works really well with multiple AWS data sources, such as Amazon S3, RDS, and DynamoDB.

Key Components

- **Crawlers:** These will automatically discover and then classify the data from your sources.
- **Data Catalog:** A central storage unit that holds the metadata which the crawler discovers.
- **Glue Jobs:** Where the ETL is executed.
- **Workflows and Triggers:** The workflows define the series of multiple glue jobs, while the triggers are the ones starting the glue jobs based on events.
- **Monitoring:** Where the performance of glue jobs are tracked enabling you to troubleshoot in case a problem arises.

Example application

An e-commerce company has different types and formats of their datasets. This means they will need to aggregate data from multiple sources to use these datasets for analyzation.

The company stores their data as follows:

- **S3 Buckets** - order data
- **RDS** - customer profiles
- **DynamoDB** - real-time inventory updates

To further visualize the process, a graph below can be used as a reference:

Using Apache Spark for Large-Scale Data Processing

Apache Spark

It is an open-source distributed processing system that works across multiple machines and excels at fast data processing through its in memory features.

Key Features

- **Scalability:** It can scale from small local systems to large cloud clusters.
- **Integration:** Also used as the core for AWS Glue ETL for more efficient and easier data processing and transformation.
- **Fault Tolerance:** It can handle failures which will ensure data integrity.

Example application
The Department of Transportation (executive department in the Philippines) processes different types of data to be used in concluding a more dependable transportation system in Manila.

- **S3 Buckets** - traffic data from Metro Manila
- **RDS** - weather data gathered from regional weather stations

Using Apache Spark, the datasets are merged to predict congestions patterns to help the department gather more insights.

Chapter 1.7 Troubleshooting Data Ingestion and Storage Issues

Data ingestion and storage matter for a good machine learning processes. However, things like full servers and size limits cause slowdowns next to mistakes in data flows. The chapter explains how to spot and fix these issues with AWS tools plus proven methods.

Diagnosing Capacity and Scalability Problems

Common Issues

- **Insufficient Throughput:** Data processing tasks fail when services such as Amazon Kinesis or DynamoDB lack enough capacity to handle the incoming data flow.
- **Storage Constraints:** The Amazon S3 buckets might reach their limits or need better storage rules, which leads to data upload errors.
- **Scaling Delays:** Database setups that depend on Amazon RDS or EBS often hit limits when traffic increases plus fail to meet speed requirements under heavy usage.

Example

In the Philippines, a logistics company faces the problem of delayed processing of real-time delivery updates during peak hours (e.g., holidays or sale events such as 11.11). They are using Amazon Kinesis Data Streams but run into speed limits, which results in data ingestion lags.

Techniques for Diagnosis

The following consists of the list of techniques you can try based on the given example.

- **Monitor Resource Utilization**
  - **Amazon CloudWatch** - monitor metrics like CPU utilization, throughput, and memory usage.
  - **CloudWatch Alarms** - enable this to be notified of threshold breaches
- **Analyze Traffic Patterns**
  - Investigate the data ingestion patterns to identify traffic spikes or any other possible problem.
- **Check AWS Service Limits**
  - **AWS Trusted Advisor** - utilize this to verify your service quotas such as the RDS instance connections.

Tools and Techniques for Debugging Storage and Ingestion Workflows

AWS Tools

- **AWS CloudWatch Logs Insights**
  - You must inspect specific logs from data pipelines and streams in AWS Glue Amazon Kinesis as well as Lambda.
- **Amazon CloudTrail**
  - Track your API performance to spot problems such as restricted access or unsuccessful data operations with S3, DynamoDB, as well as RDS.
- **Amazon S3 Event Notifications**
  - Set up the necessary notifications to identify and respond efficiently to failed uploads or updates in S3.
- **AWS Glue Job Metrics**
  - To debug possible ETL performance issues, utilize this to monitor Glue job execution metrics.

Best Practices

In debugging storage and ingestion workflows, it's important to remember some points that will make your job easier. The following are not rigid rules to be followed, but guidelines to help your work.

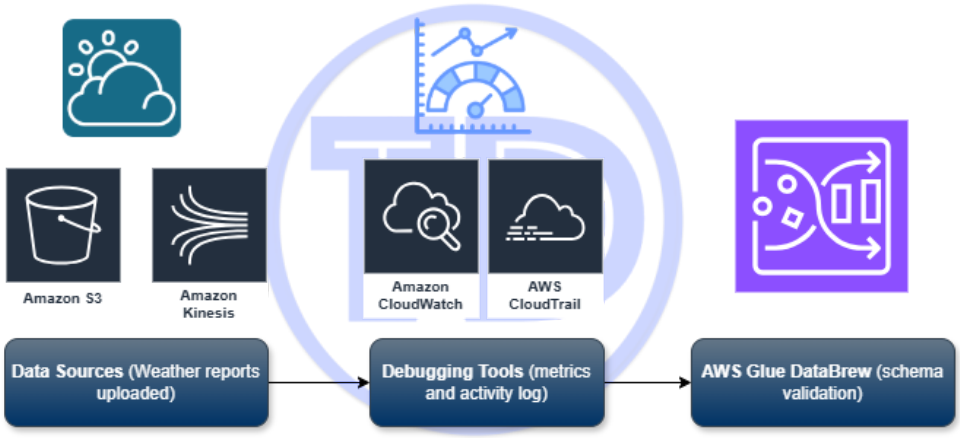
- Enable retry logic

- Data partitioning
- Schema validation

Example application

The Philippine Atmospheric, Geophysical, and Astronomical Services Administration (PAGASA) decided to ingest weather data from regional stations into S3 for analysis. However, due to mismatched database schema, this pipeline tends to fail. As a result, they can't get accurate real-time insights during typhoons.

- **Solution Using Tools:**
  - **AWS Glue DataBrew** - spot and fix database schema differences.
  - **CloudWatch Logs** - identify ingestion errors.



B. Transform Data and Perform Feature Engineering

This is an important part as it will make sure that the dataset can result into meaningful insights.

Chapter 1.8 Data Cleaning and Transformation Techniques

Outlier Detection and Removal

Outliers
These are data points that appear abnormally far from normal values in a dataset. They often mess up a model, creating less accurate predictions.

Techniques

- Z-Score or Standard Deviation
- IQR (Interquartile Range)
- Visual Methods: Boxplots or scatter plots
- AWS Tools: SageMaker Data Wrangler or AWS Glue DataBrew can be used for automated outlier detection.

Example application
A logistics company in Metro Manila examines delivery schedules and notices that abnormal high times occur because of traffic jams during holidays. These extremes might represent outliers that need removal or to be flagged.

Imputation of Missing Data

Missing data
When a dataset has missing values, those are considered missing data. One way to replace the missing data with appropriate values is through imputation.

Imputation techniques:

- **Mean/Median/Mode**: Replace empty values with a column's mean, median or mode value.
- **Forward Fill/Backward Fill**: Use the previous or next data to fill the value
- **Predictive Imputation**: Use other features in the dataset to train a model and predict the values.
- **AWS Tools**: **SageMaker Data Wrangler** for built-in imputation workflows or **AWS Glue** transformations.

Example Application
A bank based in Cebu needs to analyze loan data. However, there are missing income values for some customers. They have decided to replace those missing values with the average income of their age group and location.

Combining Datasets and Deduplication Methods

Combining datasets
Merging data from multiple sources.

Deduplication
Removal of redundant records to maintain data integrity.

Steps you can take:

1. **Join Operations**: Merge datasets using keys (e.g., customer ID).
2. **Deduplication**: Identify duplicates using their unique identifiers, and then apply the appropriate logic to remove the duplicates.
3. **AWS Tools**: For easier and more efficient workflow, use AWS Glue to join datasets and handle duplicates.

Example application
Combining customer profiles from mobile app data and website registrations can cause duplicates of data. Applying the methods above or even utilizing AWS Glue will make the work easier with lesser mistakes.

Chapter 1.9 Feature Engineering Techniques

Feature engineering helps machine learning models perform better through changes to raw data. The process turns basic information into useful parts which a model understands. Main methods include data scaling along with standardization, feature splitting binning or log transformation as well as data normalization.

Data scaling and standardization methods

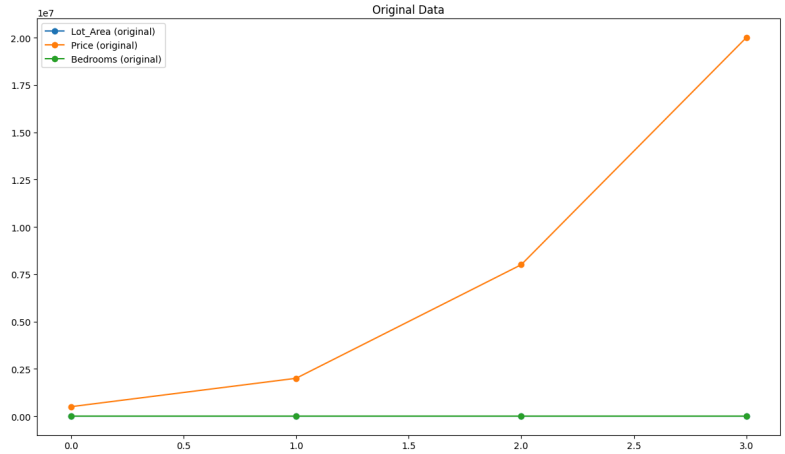
The scaling or standardization process puts numerical features in a similar range. This step prevents features that have large values from taking control over features with smaller values.

Example
A startup in the Philippines is building a house pricing prediction model. The dataset includes:

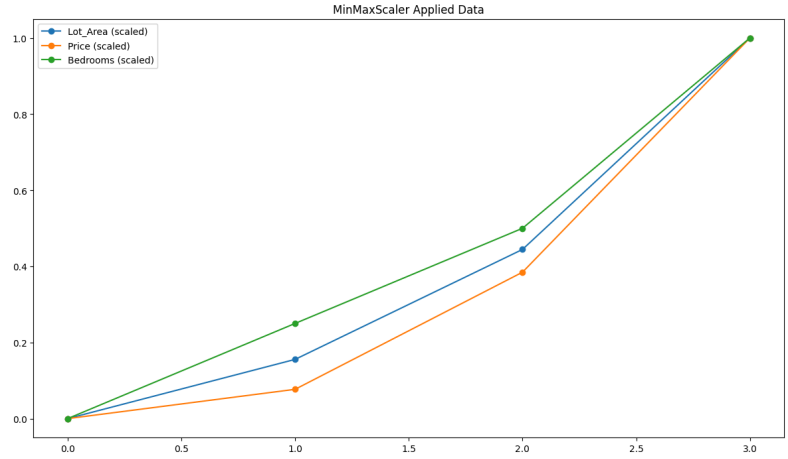
- **Lot Area (sqm)**: from 50 to 500
- **Price (PHP)**: from ₱500,000 to ₱20,000,000
- **Number of Bedrooms**: from 1 to 5

Since house prices are significantly larger than lot sizes and bedroom numbers, the models tend to focus too much on price data. A scaling method like MinMaxScaler or StandardScaler helps each feature add value in fair amounts.

Before data scaling and standardization methods:



The first chart shows the original data plotted. The features have different scales, with the 'Price' feature dominating the chart due to its larger values than the other features. The blue line that is supposed to be representing the 'Lot Area' is not visible or might appear flat due to the same reason.



The second chart above shows that after scaling (MinMaxScaler), all features fall within the (0, 1) range, making it more comparable in magnitude.

Feature splitting, binning, and log transformation

These transformations adjust feature relationships to create more normal data distributions.

Example
A local sari-sari store keeps tabs on each customer's purchases. The amount of items follows an uneven pattern, as most people select 1-3 products, yet a small number choose to buy in large quantities.

- **Feature Binning**
  - Groups values into intervals to simplify modeling
- **Log Transformation**
  - Applies logarithmic transformation to help reduce the skewness of highly varying numeric features

Chapter 1.10 Encoding Techniques for ML Data

Encoding methods help prepare category and text data for ML models. This section explains one-hot encoding, binary encoding, label encoding or tokenization for text data.

One-hot encoding, binary encoding, and label encoding

One-hot encoding
One-hot encoding (OHE) turns groups of data into binary vectors. A category appears as a specific binary vector that contains a single "1" with zeros in other spots.

The example shows a dataset with regions in the Philippines where a business works:

Region
NCR
Visayas
Mindanao

After applying one-hot encoding, the table can be transformed into:

NCR	Visayas	Mindanao
1	0	0
0	1	0
0	0	1

Binary encoding
The method transforms categories into compact binary formats reducing dimensionality.

Example
There are three product categories in a sari-sari store:

Product Category
School supplies
Laundry supplies
Groceries

Each category will get a numerical label: School supples - 1, Laundry supplies - 2, Groceries - 3.

These numbers will then get turned into binary:

Product Category	Label	Binary	Split Bits
School supples	1	001	0 0 1
Laundry supplies	2	010	0 1 0
Groceries	3	011	0 1 1

Label encoding

Label encoding converts categories into numbers and works well with algorithms that need ordered values.

Example

A company has a loyalty program with different membership levels:

Membership Level
Silver
Gold
Platinum

Using label encoding, they will be:

Membership Level	Encoded Value
Silver	0
Gold	1
Platinum	2

Tokenization for text data

Tokenization splits text into smaller units, which we call **tokens**. These units consist of words or subwords that NLP models need to function.

Example

In an article there is a Filipino news headline written: "Bumagsak ang presyo ng bigas sa palengke" (*The price of rice in the market has dropped.*).

This headline can be split into different ways.

Word Tokenization

["Bumagsak", "ang", "presyo", "ng", "bigas", "sa", "palengke"]

Subword Tokenization - frequent prefixes or suffixes can be used

["Bumag", "sak", "ang", "presyo", "ng", "bi", "gas", "sa", "pa", "lengke"]

Character Tokenization - each character of the words are tokenized

["B", "u", "m", "a", "g", "s", "a", "k", " ", "a", "n", "g", " ", "p", "r", "e", "s", "y", "o", “n”, “g”, “b”, “i”, “g”, “a”, “s”, “s”, “a”, “p”, “a”, “l”, “e”, “n”, “g”, “k”, “e”]

How to know which tokenization to use?

Tokenization Type	Pros	Cons
Word Tokenization	Easier to interpret	Large vocabulary
Subword Tokenization	Can handle new words	More complex
Character Tokenization	Can handle any word	Longer sequences

Chapter 1.11 Tools for Data Exploration, Transformation, and Feature Engineering

Using SageMaker Data Wrangler for Data Exploration and Transformation

Amazon SageMaker Data Wrangler makes data preparation easier for machine learning (ML) by offering ready-made transformations along with tools for data visualization or statistical summaries. You can clean, filter, and visualize your data without complex coding work.

This service is also useful for providing data transformations, feature engineering, and balancing datasets, particularly for binary classification.

Example
Imagine you work as a data analyst at a logistics company where you need to examine delivery times or delays per route. SageMaker Data Wrangler helps you import this data quickly, remove unusual entries like heavy traffic stops ("colorum" checkpoints) as well as make features like the standard delivery time for each barangay (neighborhood).

AWS Glue and AWS Glue DataBrew for ETL (Extract, Transform, Load)

AWS Glue
It serves as a serverless ETL service to transfer large amounts of data between data lakes, databases or warehouses without server management.

AWS Glue DataBrew
It provides an interface that lets users prepare data through visual steps, which include ways to deal with missing values along with encoding methods. This way, it can assist in feature engineering with visual tools for non-technical users.

Example
A bank in Makati uses AWS Glue to transfer financial records from local computers to Amazon Redshift for data analysis. AWS Glue DataBrew helps remove sensitive details like secret PIN numbers automatically.

Chapter 1.12 Transforming Streaming Data

This section explains how to work with data that flows continuously in real time. The methods matter for analytics programs, applications that react to events as well as machine learning models that adapt.

Real-time Data Processing Using AWS Lambda

AWS Lambda
It enables real-time data processing by executing code in response to events. It adjusts its capacity on its own to handle data streams from Amazon Kinesis Data Streams or DynamoDB Streams without server management by users.

Example
A fintech startup in Quezon City monitors real-time transactions. AWS Lambda can process every transaction to detect fraud. In case an unusual pattern is detected (e.g., repeated transfers to unknown accounts within seconds), Lambda can trigger a security alert.

Spark-based Transformations in Amazon EMR

Amazon EMR (Elastic MapReduce)
A scalable service makes big data frameworks like Apache Spark easier to operate. For tasks that need speed, Spark excels at handling huge amounts of streaming data with detailed changes in real time.

Example
A media company can analyze viewer interactions during live streaming of a major event. With the use of Spark on Amazon EMR, it can aggregate data on views, comments, and reactions to update dashboards in near real-time.

Chapter 1.13 Data Annotation and Labeling for High-Quality Datasets

High-quality datasets make better machine learning models. This chapter explains data annotation along with labeling with tools such as Amazon SageMaker Ground Truth as well as Amazon Mechanical Turk.

Using SageMaker Ground Truth for Labeled Data Generation

Amazon SageMaker Ground Truth

A fully managed solution which lets users create labeled datasets. Ground Truth combines manual or automated labeling methods. The system helps cut labeling costs through active learning, which marks the most reliable predictions without human input.

Example
An e-commerce company wants to classify product images. SageMaker Ground Truth can automate this labeling process by assigning common categories (e.g., "electronics," "clothing") to frequently occurring images. For more niche categories (e.g., *Filipino crafts* like "banig" or "kalinga fabric"), human annotators can verify or refine the labels.

Amazon Mechanical Turk for Crowd-Sourced Data Annotation

Amazon Mechanical Turk (MTurk) serves as a platform where people assign tasks to remote workers. The service works well for projects that need human decisions on tasks like image sorting, opinion reviews or written copies of speech.

Example

A news company like Inquirer needs to analyze thousands of reader comments for positive or negative sentiment. The company can outsource this task to Mechanical Turk workers who help create useful datasets through quick annotations.

Best Practices for MTurk

- Clear annotation instructions (if possible, provide sample images and labels)
- Quality control measures (as simple as multiple workers verifying the same data point)

C. Ensure Data Integrity and Prepare Data for Modeling

Chapter 1.14 Understanding Pre-Training Bias Metrics

Class Imbalance (CI) and Difference in Proportions of Labels (DPL)

Class Imbalance
The term refers to cases where a class (like fraud versus non-fraud) has many more examples than the other class. This mismatch leads to a model with poor performance.

> **Example**
> For example, in a Filipino e-commerce dataset where the majority of transactions are legitimate, but only a small percentage are fraudulent, the model might end up biased towards predicting legitimate transactions.

Difference in Proportions of Labels
This helps to identify if labels are spread unevenly across different groups.

> **Example**
> For instance, a Filipino bank's loan dataset that includes many applications from specific region (could be majority are from the NCR [National Capital Region]) or age categories could favor those applicants in unexpected ways.

Bias Detection in Numeric, Text, and Image Data

- **Numeric data -** The ML model could lead to unfair loan decisions where some income levels receive more approval than others.
- **Text data** - Text data from customer feedback shows sentiment patterns that skew towards unfavorable views about certain groups.
- **Images** -.The model may show preferences for specific facial features because of imbalances in its training data.

Using SageMaker Clarify for bias metrics analysis

Amazon SageMaker Clarify
This service helps in bias detection, ensuring fairness in the data by identifying issues before model training.

Chapter 1.15 Data Encryption and Security

Techniques for Securing Data during Ingestion, Storage, and Processing

Data encryption at rest and in-transit
The encryption makes data unreadable when stored or sent across networks.

AWS KMS (Key Management Service)
This service lets users create or manage encryption keys.

AWS Encryption Services and Best Practices

Amazon S3
Data stays safe through server-side encryption at rest. The SSL/TLS protocol guards all information during transit.

IAM (Identity and Access Management) roles
These roles serve as a safeguard to restrict who can obtain encryption keys and protect sensitive data from unauthorized decryption.

Example
In a fintech application from the Philippines, an AWS KMS system encrypts transaction records before their storage in Amazon S3. The transfer process remains secure through IAM roles in AWS, which blocks access from unwanted users.

Chapter 1.16 Data Classification, Anonymization, and Masking

Types of information

- **Personal Identifiable Information (PII)**
  - includes data: names, addresses along with contact details.
- **Protected Health Information (PHI)**
  - personal information about a person's medical condition along with physical or mental well-being.

Approaches for Handling Sensitive Data (e.g., PII, PHI)

- **Data masking**
  - A process to replace actual information with fictitious values lets organizations keep data patterns intact while protecting sensitive details.
- **Anonymization**
  - Data anonymization removes personal details from records so no one can link the information to specific people.

Example

A DOH (Department of Health) healthcare app masks data to hide patient names and contact details before it shares records with analytics companies. The process lets the app track its performance while it maintains PHI compliance.

Best Practices for Data Masking in AWS

- **AWS Glue**
  - For ETL tasks to mask or anonymize data which needs protection.

- **AWS Macie**
  - A specialized AI service that discovers classifies and protects sensitive data like PII in datasets. This service finds personal information in digital files along with sensitive items to keep records safe.

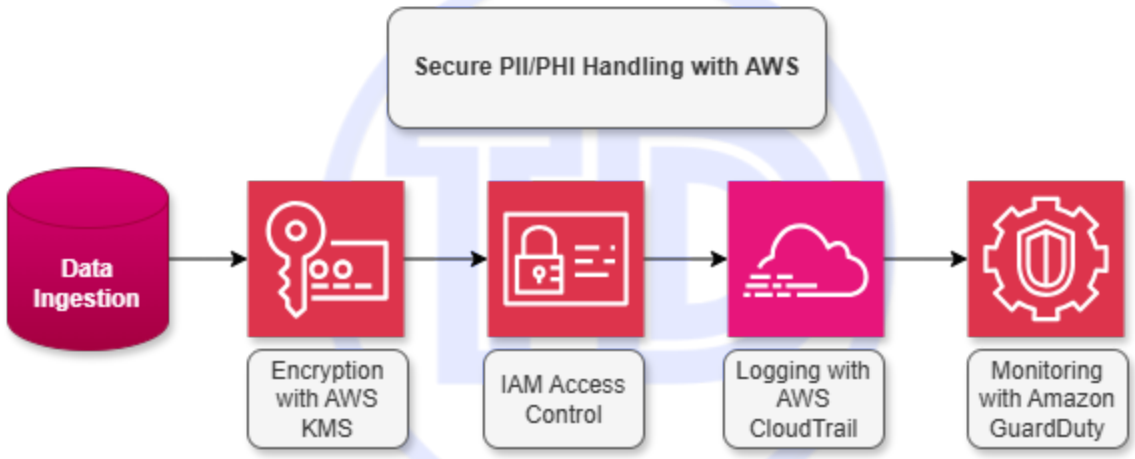
Chapter 1.17 Understanding Compliance Requirements

Handling Personally Identifiable Information (PII) and Protected Health Information (PHI)

PII (Personally Identifiable Information)
It consists of data that identifies a particular person. Common identifiers such as names or addresses are included. A resident's contact details, Social Security number along with telephone info serve as additional data points for identification.

PHI (Protected Health Information)
It consists of personal data related to medical care such as patient records, laboratory examinations as well as insurance documentation.

Example
A healthcare startup in Manila stored their data in their Amazon S3 bucket which consists of names, age, and diagnoses of the patients. According to the Data Privacy Act of 2012 (RA 10173) in the Philippines, these PII data must be encrypted, which can be done with AWS Key Management Service (KMS), while controlling the access using AWS IAM.



Above is an example of secure handling of PII/PHI data with AWS for reference.

Data residency requirements and their impact on ML workflows

Data residency
These are regulations dictating where data can be stored and processed. There are countries which mandate that confidential details stay inside their borders. With this, organizations must choose AWS Regions that comply with these rules.

Example

A private bank in the Philippines uses Amazon SageMaker to build a fraud detection model. However, they have sensitive data and as a bank has to follow BSP (Bangko Sentral ng Pilipinas) regulations. They can use the AWS Local Zone in Manila (ap-southeast-1-mnl-1a) which has Asia Pacific (Singapore) as its parent region to ensure that the data will remain within the country.

**Note that this is an illustrative example and the AWS Local Zone in Manila operates independently but users must verify all data stays within its boundaries. A proper inspection confirms no information flows to Asia Pacific (Singapore) regions.*

Chapter 1.18 Validating Data Quality

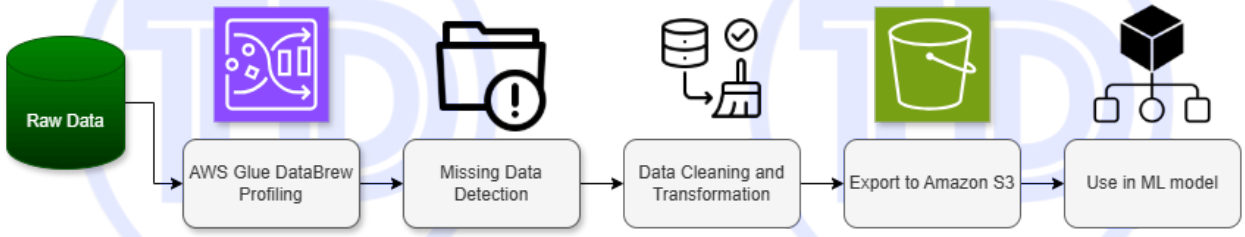
Using AWS Glue DataBrew for Data Quality Checks

AWS Glue DataBrew

The platform enables users to examine or validate data visually before implementing it in machine learning tasks. It also performs automatic detection of anomalies along with data profiling as well as transformations.

Example

An e-commerce company based in Cebu uses AWS Glue DataBrew to clean their sales data. They've discovered double transactions and missing values so they've decided to use DataBrew transformations before sending the data to Amazon SageMaker.



Leveraging AWS Glue data quality features for validation

AWS Glue Data Quality
A data validation step which confirms the correctness of datasets before ML model training starts. It can detect issues such as missing values, data drift, and schema mismatches.

Example
A telecom company in the Philippines uses AWS Glue Data Quality to ensure that there are no duplicate customer IDs or invalid phone numbers before using the dataset for their churn prediction.

Chapter 1.19 Identifying and Mitigating Bias in Data

Using SageMaker Clarify to identify and mitigate bias

Amazon SageMaker Clarify
This is helpful for ML engineers to detect biases before, during, and after model training. Bias detection reports and model explainability insights are provided.

Example
An insurance company in Valenzuela City built an AI model for loan approvals. However, they've noticed that the rejection rates for applicants outside the said city are higher. As a response, they've used SageMaker Clarify to validate and fix the bias of the model before deploying it.

Addressing Selection Bias, Measurement Bias, and Other Bias Types

1. **Selection Bias** - Happens when a dataset doesn't represent the totality of the population.
2. **Measurement Bias** - If there are errors in data collection, it results to biased predictions.
3. **Label Bias** - In the case of human-labeled data, subjective errors can occur.

Example
A BPO (Business Process Outsourcing) company in the Philippines developed an ML model for resume screening. However, since the majority of their data came from Manila-based companies, a selection bias occurred against applicants from Visayas and Mindanao. In order to balance their dataset, they have used oversampling techniques.

Model Explainability with SageMaker Clarify
This service has tools for model explainability, which means you can understand which features affect your model's prediction the most. Typically used for ensuring transparency and accountability. This is done during the data preparation phase.

Chapter 1.20 Preparing Data for Model Training

Techniques to prepare data for ML models

Dataset Splitting
With dataset splitting, your dataset is divided into three (3) parts: training (70-80%), validation (10-20%), and testing (10%).

> **Example**
> A transportation company may divide their dataset of ride requests to train their model of its demand pattern.

Shuffling
This is a technique wherein the dataset is randomly arranged to avoid having pattern bias for the model.

> **Example**
> If all of the transactions of OFWs are in the first part of the dataset, the model has the possibility of being overfit.

Data Augmentation
In a general sense, this is a technique that creates synthetic variations of data. In computer vision, it is possible to use image flips and rotations.

> **Example**
> An example is a local basket weaving e-commerce startup that uses this to augment their product images.

Feature Scaling and Normalization
For datasets such as financial records, which consist of both small and large amounts, MinMaxScaler or RobusScaler in Amazon SageMaker Data Wrangler are used to ensure that the data points are uniform.

Storing Training Data Efficiently

Amazon Elastic File System (EFS)
In cases that real-time model training is done across multiple EC2 instances, EFS is used since it is elastic.

> **Example**
> A logistics company might use a shared file system for their delivery tracking model training.

Amazon FSx for Lustre
This is ideal for high-performance machine learning (HPC) workloads that need sub-millisecond latency.

> **Example**
> For a real-time animal monitoring of Philippine reserves, Amazon FSx for Lustre can be used for drone imagery models.

Other Example Applications

- **Traffic Congestion Prediction**
  - MMDA uses traffic sensor data to train their ML model before predicting the heavy traffic hours in EDSA. During the data preparation stage, they have split the historical data into different date and time of the year.

- **Customer Segmentation for Local Retail Chains**
  - In a grocery chain in Manila, ML models are used to segment their customer base based on their purchase patterns. They will first divide the data before training to test the accuracy of unseen data.

D. Optimizing Data for Machine Learning Models

Chapter 1.21 Data Preprocessing for Enhanced Model Performance

Techniques to Reduce Prediction Bias and Overfitting

Overfitting

A machine learning (ML) model exhibits overfitting when it excels at training data but fails to predict results correctly from new, unseen data.

Prediction bias

This happens when the model has bias due to imbalanced or incorrect data handling.

Key techniques to mitigate these issues:

- **Cross-validation**
  - This is when the dataset is split into multiple subsets, and then after the model is trained, it will be validated multiple times to ensure good performance.
  - **Example:** A telco provider analyzes their customer churn using k-fold cross-validation to make sure that their model performance is consistent across different customer segments.
- **Feature Engineering**
  - The technique of creating new features based on raw data to improve model learning.
  - **Example:** In banks, derived features such as 'average monthly balance' are used for their loan approval prediction models.
- **Regularization**
  - L1 (Lasso) or L2 (Ridge) regularization methods add specific penalties whenever model coefficients become too large. These mathematical restrictions prevent a model from turning overly complex and stop it from overfitting the data.
  - This will be discussed in the next section "ML Model Development".
- **Handling Imbalanced Datasets**
  - With the use of techniques such as oversampling, undersampling, or class-weight adjustments, imbalanced data can be addressed.
  - **Example:** To reduce prediction bias in fraud detection models, add training samples for rare fraudulent transactions.
- **Temperature scaling**
  - This is a post-training technique, used in model tuning, particularly when recalibrating prediction probabilities for classification.
  - **Example:** If an ML model is overconfident, with a high probability but doesn't match realistic observations, this technique is useful for adjusting these probabilities, ensuring they are more realistic.

Optimizing Training Data through Transformation and Augmentation

Data transformations convert and adjust training data to make models learn better. Numbers get scaled categories, turn into code, or data shapes change. Augmentation adds variety to existing datasets at no extra collection expense.

- Normalization and Standardization Scale
- One-Hot Encoding and Label Encoding
- Data Augmentation

Other Example Applications

- **Weather Forecasting in PAGASA**
  - The ML models for typhoon predictions use data augmentation so that they will be trained on different weather patterns for each region in the Philippines.
- **Healthcare Analytics**
  - In the case of remote clinics, ML models for disease diagnosis apply data preprocessing techniques like outlier removal in order to avoid skewed predictions caused by erroneous patient data.

Chapter 1.22 Data Loading and Configuration for Model Training

Efficient Data Loading Strategies for ML Models

Efficient data loading makes model training quicker as well as prevents resource limitations. Poor approaches to data management can cause slow training or memory problems.

Below are best practices and techniques:

- **Batch Data Loading**
  - The system divides datasets into smaller batches instead of loading everything at once. Training iterations process one batch at a time, which helps save memory space.
  - **Example:** For an agricultural ML model for crop yield prediction, the batches of satellite imagery data are loaded per region of the Philippines to avoid extensive memory usage.
- **Pre-fetching and Data Pipelining**
  - The model pre loads groups of data simultaneously to minimize gaps between training cycles.

    - **Example:** For ride hailing company demand prediction models, data for the next batch is being prepared while the current batch is still being processed.
- **Data Caching**
    - The system stores datasets in memory if users access them often. A direct memory caching process reduces input and output overhead. This approach works really well with extensive data collections located on remote storage platforms such as Amazon S3.
- **Sharding**
    - Training nodes process separated portions of extensive datasets in distributed systems. Each portion, also known as a shard, and functions independently.
    - **Example:** In a fraud detection model, each shard can match with the transaction data from different branches of a Philippine bank.

Using Amazon FSx and Amazon EFS for Model Training Data Storage

AWS delivers ready-to-use storage solutions that excel at ML model training. The services grant quick access to vast data quantities along with seamless expansion options for distributed machine learning tasks.

Amazon FSx for Lustre

The system excels at fast computations for HPC along with machine learning tasks. A direct connection to Amazon S3 allows quick access to vast data collections without any manual transfers.

Example

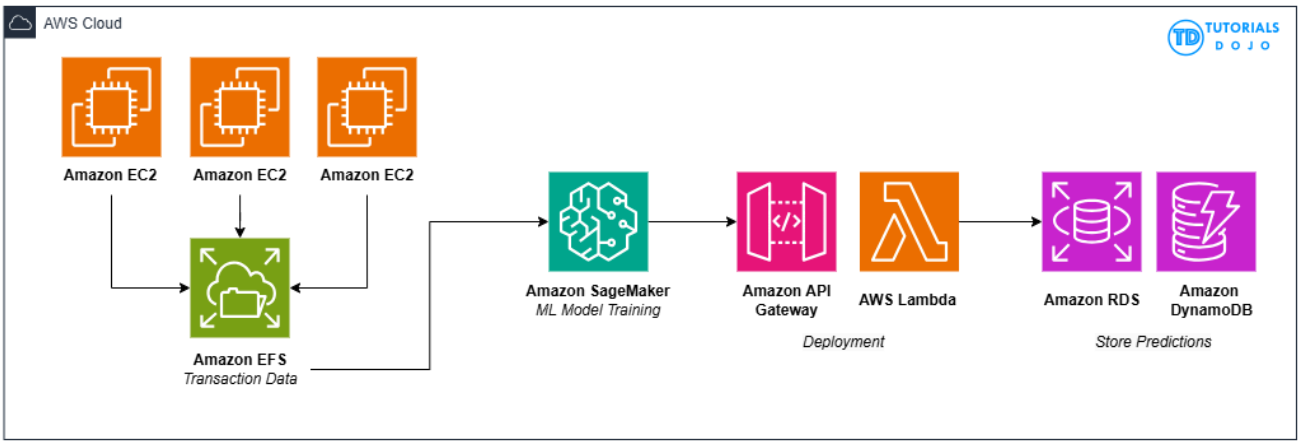
For universities such as UP (University of the Philippines) and DLSU (De La Salle University) which conduct genome sequencing research, FSx for Lustre can help them train deep learning models on massive biological data.

Amazon Elastic File System (Amazon EFS)

It provides a managed file storage solution that grants parallel access to thousands of EC2 instances. The service adapts storage capacity to match current requirements as well as copies information to several AZs.

Example

A logistics company in Visayas uses Amazon EFS for storing customer transaction data for training a package delivery time prediction model.



Other Example Applications

- **Weather Research**
  - In PAGASA, FSx is used for Lustre in order to load historical and real-time weather data for typhoon forecasting models.
- **Public Transport Optimization**
  - The ML models used for traffic analysis and route optimization in Metro Manila by MMDA rely on distributed data loading strategies using AWS storage services, which handle large traffic datasets efficiently.

Domain 1: Data Preparation for Machine Learning (ML) Sample Questions

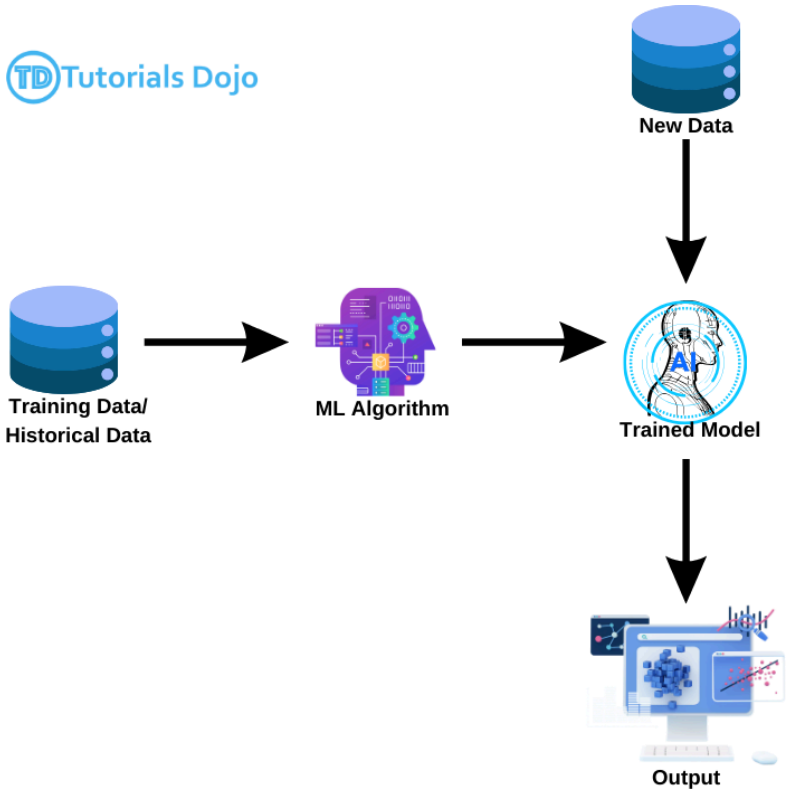
1. A company is working on a machine learning project that involves analyzing customer data to predict churn rates. Which of the following statements best describes the machine learning approach the company should take?
    a. Develop a set of hard-coded rules and conditional statements to generate predictions.
    b. Feed the customer data into machine learning algorithms, which will learn patterns and build models to make predictions.
    c. Manually analyze the customer data and replicate the analysis process using custom code to automate predictions.
    d. Use raw, unprocessed customer data directly as input to train the machine learning model.

Explanation

Machine Learning (ML) is the process of training computers, using math and statistical processes, to find and recognize patterns in data. After patterns are found, ML generates and updates training models to make increasingly accurate predictions and inferences about future outcomes based on historical and new data. For example, ML could help determine the likelihood of a customer purchasing a particular product based on previous purchases by the user or the product's past sales history.

The machine learning process typically involves:

- Data collection and preparation: Gathering relevant data and preprocessing it (cleaning, formatting, feature engineering) to make it suitable for model training.
- Model training: Feeding the prepared data into machine learning algorithms, which learn patterns and relationships within the data to build a model.
- Model evaluation: Assessing the trained model's performance using evaluation metrics and techniques like cross-validation.
- Model deployment: Integrating the trained and evaluated model into applications or systems to make predictions or decisions on new data.



Machine learning involves feeding data into algorithms, which then build models that can make predictions or decisions without being explicitly programmed with rules.

Therefore, the correct answer is: **Feed the customer data into machine learning algorithms, which will learn patterns and build models to make predictions.**

The option that says: **Develop a set of hard-coded rules and conditional statements to generate predictions** is incorrect because machine learning don't just rely on simple hard-coded rules and if-else conditions but rather learns from data and adapts over time. Additionally, AWS emphasizes that machine learning algorithms should not be explicitly programmed with rigid rules.

The option that says: **Manually analyze the customer data and replicate the analysis process using custom code to automate predictions** is incorrect because it only contradicts the automated nature of machine learning. Machine learning algorithms are designed to learn patterns and make predictions automatically from data without the need for manual analysis or replication of human processes.

The option that says: **Use raw, unprocessed customer data directly as input to train the machine learning model** is incorrect because AWS recommends data preprocessing and feature engineering to prepare the data

for effective machine learning model training. Using raw, unprocessed data directly can lead to poor model performance and inaccurate predictions.

References
https://community.aws/content/2drbbXokwrIXivItJ8ZeCk3gT5F/introduction-to-artificial-intelligence-and-machine-learning?lang=en
https://aws.amazon.com/what-is/machine-learning/

Check out this AWS Machine Learning And AI Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheets-aws-machine-learning-and-ai/

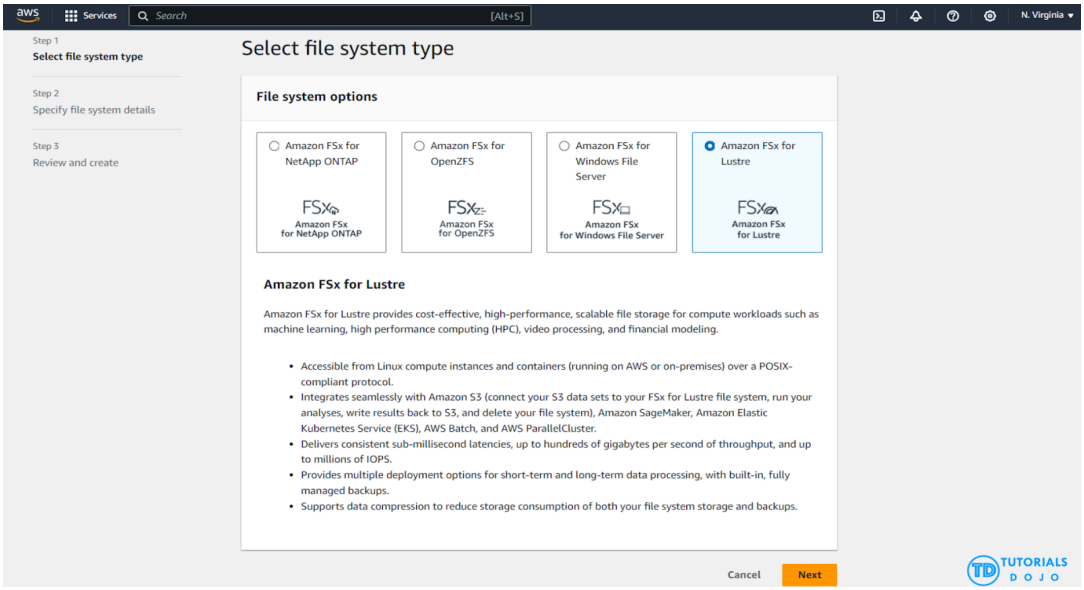
2. A research institute is developing a deep-learning model to analyze satellite images for weather forecasting. The dataset consists of millions of high-resolution images stored in a format that requires file-level access. The model training process involves multiple GPU-powered Amazon EC2 instances that require high throughput and low latency access to the dataset for efficient distributed training.

   Which AWS storage solution is the MOST suitable to support the deep-learning model training process?

   a. Amazon Elastic File System (Amazon EFS)
   b. Amazon Simple Storage Service (Amazon S3)
   c. Amazon FSx for Lustre
   d. Amazon FSx for Windows File Server

Explanation

Amazon FSx for Lustre is a fully managed high-performance file system designed for workloads that require fast, shared storage with sub-millisecond latencies and high throughput. This makes it particularly suitable for machine learning and high-performance computing (HPC) applications, such as deep learning model training. With Amazon FSx for Lustre, researchers can achieve efficient distributed training of their models by leveraging its capability to provide low-latency and high-bandwidth access to large datasets. This ensures that the multiple GPU-powered Amazon EC2 instances can access the data simultaneously without significant delays, leading to faster and more efficient training processes.



Furthermore, Amazon FSx for Lustre integrates seamlessly with Amazon S3, allowing easy data transfer between the high-performance file system and scalable object storage. This integration is beneficial for workflows that involve pre-processing large datasets or saving model training results. The file system also supports file-level access, which is essential for applications requiring fine-grained data storage and retrieval control. Given these features, Amazon FSx for Lustre is the most suitable choice for supporting NLP model development, as it meets the high throughput and low latency requirements essential for efficient deep learning training on satellite image datasets.

Hence, the correct answer is: **Amazon FSx for Lustre.**

The option that says: **Amazon Simple Storage Service (Amazon S3)** is incorrect. Although S3 is excellent for storing large datasets and integrates well with many AWS services, it does not provide the necessary low-latency and high-throughput file-level access required for efficient distributed training. Amazon S3 is an object storage service designed primarily for high durability and scalability.

The option that says: **Amazon FSx for Windows File Server** is incorrect. This file system is specifically for Windows-based applications that require SMB (Server Message Block) protocol support. This option is less suitable for high-performance computing tasks typically associated with Linux-based systems and deep-learning model training as it does not offer the same level of performance optimization as Amazon FSx for Lustre for high-throughput.

The option that says: **Amazon Elastic File System (Amazon EFS)** is incorrect because EFS only provides scalable file storage for use with Amazon EC2 instances. It is not optimized for the high-performance

requirements of deep learning model training on GPU instances. EFS is designed for general-purpose file storage and does not offer the same level of throughput and low latency.

References:
https://docs.aws.amazon.com/fsx/latest/LustreGuide/what-is.html
https://aws.amazon.com/fsx/lustre/
https://aws.amazon.com/fsx/

Check out this Amazon FSx Cheat Sheet:
https://tutorialsdojo.com/amazon-fsx/

References for Domain 1

What is AWS Glue? - AWS Glue

Prepare ML Data with Amazon SageMaker Data Wrangler - Amazon SageMaker AI

Apache Avro

Apache ORC • High-Performance Columnar Storage for Hadoop

Parquet

Amazon S3 - Cloud Object Storage - AWS

Object Storage Classes – Amazon S3

Apache Kafka

Fully Managed Cloud File System - Amazon FSx for NetApp ONTAP - AWS

What is Amazon Kinesis Data Streams? - Amazon Kinesis Data Streams

AWS Glue DataBrew - Tutorials Dojo

Prepare ML Data with Amazon SageMaker Data Wrangler - Amazon SageMaker AI

Change data capture for DynamoDB Streams - Amazon DynamoDB

Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration - Amazon Simple Storage Service

Serverless Data Integration – AWS Glue Features – AWS

Amazon Mechanical Turk

https://docs.aws.amazon.com/machine-learning/latest/dg/

Tutorials Dojo on SageMaker Data Preparation

Amazon EFS Overview

Amazon FSx for Lustre

Amazon SageMaker Cheat Sheets

Amazon Machine Learning Documentation.

AWS SageMaker Cross-validation Guide

SageMaker Data Transformation with Data Wrangler

Amazon SageMaker Cheat Sheet

Amazon ML Documentation on Data Preparation.

Amazon FSx Cheat Sheet

Amazon EFS Cheat Sheet

AWS Data Loading Best Practices.

Amazon SageMaker Built-in Algorithm Tuning

Amazon EFS Overview

Amazon FSx for Lustre Documentation

Amazon SageMaker Augmentation Guide

AWS SageMaker Data Loading

ML MODEL DEVELOPMENT

Choosing a Modeling Approach

Transform Data and Perform Feature Engineering

Ensure Data Integrity and Prepare Data for Modeling

Optimizing Data for Machine Learning Models

A. Choosing a Modeling Approach

Chapter 2.1 Using AWS AI Services to Solve Business Problems

Amazon Translate: Language Translation Use Cases
It works as a neural machine translation service with complete management. This service converts text between many languages in real time, which helps companies communicate across linguistic boundaries.

> **Example**
> A tourism company in Cebu will translate their promotional materials from English to Japanese and Chinese to attract international tourists. With the help of Amazon Translate, they can efficiently reach their target audience without the need of manual translations.

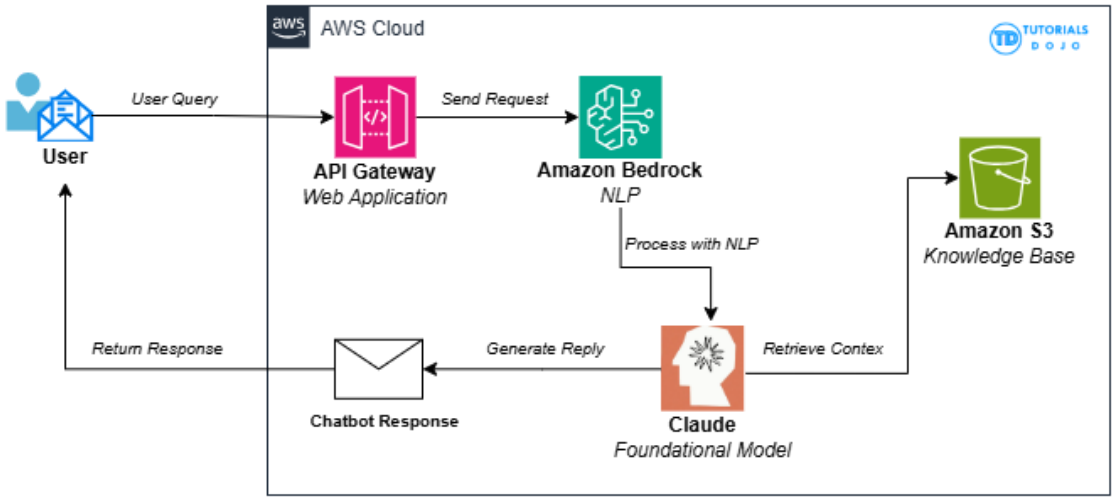
Amazon Transcribe: Speech-to-Text Applications
It converts speech to written text through AI-based systems. The service has features such as speaker identification, custom vocabularies, and real-time transcription.

> **Example**
> In Metro Manila, a BPO company uses Amazon Transcribe for automating their call transcriptions for quality assurance purposes. This saves them time and improves customer satisfaction by providing accurate conversation records.

Amazon Rekognition: Image and Video Analysis
It analyzes images and videos through machine learning algorithms. It can also detect objects, recognizes faces as well as moderate content.

> **Example**
> A shopping center in Quezon City relies on Amazon Rekognition to monitor their surveillance camera feeds. The service examines video recordings to identify potential threats like bags left behind or people who enter restricted zones.

Amazon Bedrock: Foundation Models for NLP and Image Generation
This service offers direct access to foundation models (FMs) for AI generation tasks. Users can create text content, chatbots as well as visual art. The platform removes all complexities of infrastructure setup, model training or deployment processes.



Example

The financial firm in Makati integrated Amazon Bedrock to improve its customer support chatbot. The chatbot now resolves advanced questions about loans as well as investment options.

SageMaker AutoPilot: No-code solution

This is a no-code solution that automates the model selection process

Example

An example is when a medium-sized retail chain in Cebu City is aiming to predict which products will sell the most during the Sinulog Festival. With SageMaker AutoPilot, they can upload their sales data, initiate AutoPilot, then AutoPilot will automatically execute the model selection and deployment process.

SageMaker JumpStart: Start modeling process right away

This offers pretrained, open-source models to jump-start the modeling process.

Example

A group of students in the Philippines joined a hackathon in Machine Learning. With limited timeframe to build ML solutions, they can utilize the SageMaker JumpStart to quickly start building their ML projects without having to develop from scratch.

Chapter 2.2 Considerations for Interpretability in Model Selection

Why Interpretability Matters in Model Selection

Interpretability

This shows how people can understand the decisions made by the model. Financial institutions or medical centers require transparent decisions from automated processes. The models must produce clear results that help users trust the technology as well as meet compliance requirements from regulators.

Importance of Interpretability

- **Business Transparency**
  - To exercise transparency towards clients and stakeholders, they should be able to understand how the ML models arrive at their predictions.
  - **Example:** An applicant must know why their loan is denied in a bank loan application.

- **Regulatory Compliance**
  - Legal authorities require companies in regulated sectors, such as insurance or medical services, to provide clear explanations of automated decisions for compliance purposes.
  - **Example:** The banks in the Philippines must always comply with the BSP (Bangko Sentral ng Pilipinas) regulations regarding fair lending practices.

Tools and Techniques to Ensure Model Explainability

- **SageMaker Clarify**
  - This service helps not only in bias detection, but also in post-model performance analysis by providing model explainability.
- **SHAP (SHapley Additive exPlanations)**
  - It explains why a model makes predictions.
  - **Example:** With a hospital model that predicts return patients, SHAP shows how elements such as a person's age, prior medical visits or prescribed drugs influence the final result.
- **LIME (Local Interpretable Model-agnostic Explanations)**
  - LIME explains complicated models in a simple way.
  - **Example:** It creates clear reasons for predictions of customer churn models that Philippine telco companies use.

Chapter 2.3 Using SageMaker Built-In Algorithms

Overview of Built-In Algorithms

Amazon SageMaker
It provides algorithms for different ML use. The process is more efficient due to its ready-to-use feature and the algorithms are tuned for performance.

Common Algorithms

1. **XGBoost**
    - The gradient boosted decision tree algorithm works really well for regression tasks along with classification problems.
    - **Example:** A Cebu online shop detects fraud with XGBoost by analyzing how customers buy products.
2. **Linear Learner**
    - A straightforward method that helps both with predictions of numbers and with sorting objects into categories.
    - **Example:** A property firm in Metro Manila predicts house values through factors like the neighborhood, Location, square footage as well as included facilities.
3. **K-Means Clustering**
    - This algorithm groups data points with matching attributes through unsupervised learning.
    - **Example:** A retail chain segments shoppers into specific groups according to what they buy. This allows them to design precise marketing campaigns for each customer type.

Chapter 2.4 Comparing and Selecting the Right Model or Algorithm

Evaluating available models and choosing the most appropriate for specific problems

Different models serve different purposes. In order to select the right model or algorithm, it's better to understand the use case first. For example, a Logistic Regression model is more interpretable for an application. However, it might be less accurate than using Neural Networks (although this is more complex).

Example
In a Philippine financial institution, they want to know which of their customers are more likely to default on a loan. They can use decision trees to see right away the reasons for such defaults. These could be age and income, which affect loan approvals.

Algorithm tradeoffs

When choosing an algorithm, you can't have it all. There are factors, however, that you can consider and weigh accordingly.

- **Performance**
  - The following are used for evaluating model performance.
    - **Accuracy** - correctly classified instances
    - **Precision** - true positives among instances predicted as positive
    - **Recall** - true positives among actual positive instances
    - **F1 Score** - harmonic mean of precision and recall
    - **AUC** - "Area Under the Curve"; commonly used in binary classification
    - **Mean Absolute Error (MAE)** - used in regression problems
  - Some deep learning models such as CNNs and Transformers require more computational power too.
- **Scalability**
  - Training and Inference speed
  - Models such as Random Forest, Gradient Boosting Machines, and the like scale well with large datasets but require specialized hardware such as GPUs
- **Interpretability**
  - Explainability
  - Models like Linear Regression, Logistic Regression, and Decision Trees can offer better interpretability.

Example
A financial institution in Visayas is building a loan approval model. To maximize accuracy and explainability, they have decided to go with Decision Trees or Logistic Regression (especially that regulatory transparency is a priority).

Chapter 2.5 Cost Considerations in Model Selection

Evaluating the cost of model training and inference

To evaluate the cost of model training and inference, you can utilize AWS SageMaker pricing. You pay per computation at storage usage. For instance, in model training, you can use Spot Instances to save up to 90% of the cost.

Example
A logistics company in the Philippines uses predictive maintenance for their delivery vehicles, which can scale their training during off-peak hours with the use of Spot Instances.

Selecting cost-effective solutions

When developing ML solutions, the common problem is the increasing cost of training, infrastructure, and deployment. Fortunately, you can use SageMaker JumpStart.

SageMaker JumpStart
Provides pre-trained models that you can deploy right away without needing to use a lot of time and money in training your ML model.

How can you save with SageMaker JumpStart?

1. **Less compute resources costs**
    a. There are ready-made pre-trained models that they can use right away without the need of many computational resources.
2. **Doesn't need to use large dataset or manual labeling**
    a. With pre-trained models, you don't need to invest in data annotation services.
    b. You can save time and money by being able to fine-tune the model on your own dataset.
3. **With On-demand compute, deployment is less costly**
    a. You can use AWS-managed inference endpoints (without the need to set up EC2 instances or Kubernetes clusters), and you will only pay when using the model.
    b. Billing is per second so resources are not wasted.

Example
A small e-commerce business in Siargao wants to have an AI chatbot for their customer support due to the increasing number of foreign tourists. However, they are only starting and have a small budget. With JumpStart, they can do the following:

- Choose an Amazon Lex-based chatbot without the need to train from scratch
- Use their small dataset of English and Filipino inquiries to fine-tune their model

- Deploy this model to SageMaker-managed inference in which they will only pay whenever they're using it

As a result, they can have a fully functioning chatbot in just a few days, not months, and it is ready for deployment!

B. Training and Refining Models

Chapter 2.6 Key Elements in the Training Process

Model training is similar to teaching a kid how to read their books. You need to repeatedly teach them about the information until they understand it fully. In AWS machine learning, it is important to understand the following concepts:

- Epochs
- Steps
- Batch size

These concepts will determine how fast the model can learn from its training.

Understanding key training components

- **Epochs**
  - One epoch is equal to one full cycle of the dataset going through the model.
  - **Example:** A student studying for their UPCAT (University of the Philippines College Admission Test) won't learn all the topics with a one-time review. She needs to repeatedly (epochs) review the required topics until she understands them very well.
- **Steps**
  - This is the number of batch updates in one epoch.
  - **Example:** In a call center training, the training sessions are divided per shift. In a shift (epoch), there are different lessons or training sessions (steps) that the trainees go through.
- **Batch Size**
  - This is the number of data samples that are simultaneously learned in one step.
  - **Example:** In a fast food chain based in Manila, crew training will be more efficient if they will train in one batch compared to individual training.

How to set appropriate training parameters?

To set appropriate training parameters for your model, you should take note of your dataset and use case. The following is some of the basic information you might want to get acquainted with.

- **Small dataset**
  - You can use a smaller batch size with more epochs to make the model more accurate.
- **Large dataset**
  - You can use a larger batch size with fewer epochs for faster model training.
- **Complex models**
  - You should use more epochs and be careful with batch size tuning if your model is complex (such as NLP).

Chapter 2.7 Methods to Reduce Model Training Time

The speed of model training is similar to event preparation. You need the right techniques to avoid wasting time and resources.

Early Stopping: Stopping training to avoid overfitting

This is a technique in which the training is automatically stopped if no improvements are seen from the model to avoid overfitting. This can also help save computation time.

Example
In a bar review, if you have studied for a few hours already and you still can't remember some of the facts, it's better to stop and take a break for a while. Similar to the idea of early stopping.

AWS Implementation:
Use Amazon SageMaker Automatic Model Tuning to automatically stop the training once the best performance is reached.

Distributed Training: Leveraging multiple instances for faster training

Another technique is distributed training. The training workload is divided into different compute instances to quicken the training process.

Example:
In a barangay feeding program, it will be faster if more volunteers cook and distribute the food compared to only having one person.

AWS Implementation:
Amazon SageMaker Distributed Training will make it easier to use data parallelism and model parallelism for faster ML training.

Chapter 2.8 Factors Influencing Model Size

In machine learning, the size of the model has an impact on performance, storage, and computation costs. It is important to have the right balance between accuracy and efficiency. There are factors that impact the model size as well.

How model architecture and data impact model size

Below is an example of how model architecture and data can impact the model size.

Example
In public transportation, if there are more people riding the jeep, the driver is more likely to drive slower (not always true, but you get the point!). In contrast, if it's at night and there are only two to three passengers, the jeep is driven faster. If the model size is too large, the inference time is slower.

AWS Implementation:
You can use Amazon SageMaker Neo to optimize ML models for faster and more scalable models.

Trade-offs between model complexity and resource consumption

It's the typical case that if the model is complex, it is more accurate. However, these models require more storage and computing power.

Example
A local movie platform in the Philippines uses a simple recommendation model. This is faster in implementation but not as accurate as a deep learning-based personalized recommendation.

AWS Best Practices

- Use pruning and quantization to lessen the model size.

- Use SageMaker Model Optimizer to automatically lessen the model's complexity while maintaining accuracy.

Model training, optimization, and sizing are the things that you need to take note of for the efficiency and scalability of your ML models. Using AWS AI services, you can easily optimize your ML workflows for faster, cost-efficient, and scalable deployment.

What you can try now

- You can try using Amazon SageMaker for faster and more cost-efficient model training.
- Use AWS AI Services for efficient ML workflows applicable to real-world business problems.

Chapter 2.9 Improving Model Performance

Techniques for boosting model performance

Your ML model's performance is measured by how well it can generalize to unseen data. For optimizing your model, you can try techniques such as feature engineering (transforming your dataset into meaningful features) and hyperparameter tuning (adjusting your model's parameters).

Example

In an online retail business located in Manila, there is a challenge in predicting which products customers will buy more frequently. Their newly minted data science team has used feature engineering to understand customer behavior further. Columns such as "last purchase date" and "frequency of purchase" were added as new features. This move improved their recommendation system.

Regularization Techniques:

Regularization

This is used to prevent overfitting (when the model performs well on training data but not on unseen data).

Common techniques used:

- **L1 Regularization (Lasso Regression) -** Coefficients of less important features are lessened to force some weights to be zero for a simpler model.
- **L2 Regularization (Ridge Regression) -** To avoid extreme values, the model is forced to have smaller but nonzero weights.

- **Dropout -** In deep learning, some neurons are randomly enabled or disabled in each training step to avoid reliance on specific features.

Example
The dev team of an e-wallet fraud detection system noticed that their ML model was overfitting. It has a good performance in detecting past fraudulent transactions but not new ones. They've used the L2 regularization technique to balance the influence of features and avoid relying on "high transaction amount" as their sole fraud indicator.

Chapter 2.10 Hyperparameter Tuning

Hyperparameter optimization techniques

Hyperparameter tuning
This is the process in which the best combination of hyperparameters is found to maximize model performance.

Hyperparameter Tuning Methods:

- **Random Search** - This is a fast but trial-and-error method of trying out random sets of hyperparameters of the model.
- **Grid Search** - Compared to the random search, this is systematic and tries all the possible combinations. More accurate but slower.
- **Bayesian Optimization** - This method uses probability models to look for the best parameters. The downside is the computation overhead.
- **Hyperband** - This is introduced by the SageMaker AMT, which is a technique for faster and more efficient hyperparameter tuning for finding an optimal model.

Example
In a sentiment analysis model used for customer feedback in a BPO center in Makati, Bayesian optimization is utilized. This is used to search for the best learning rate and batch size. Before, they were doing this manually, which took a lot of time and effort.

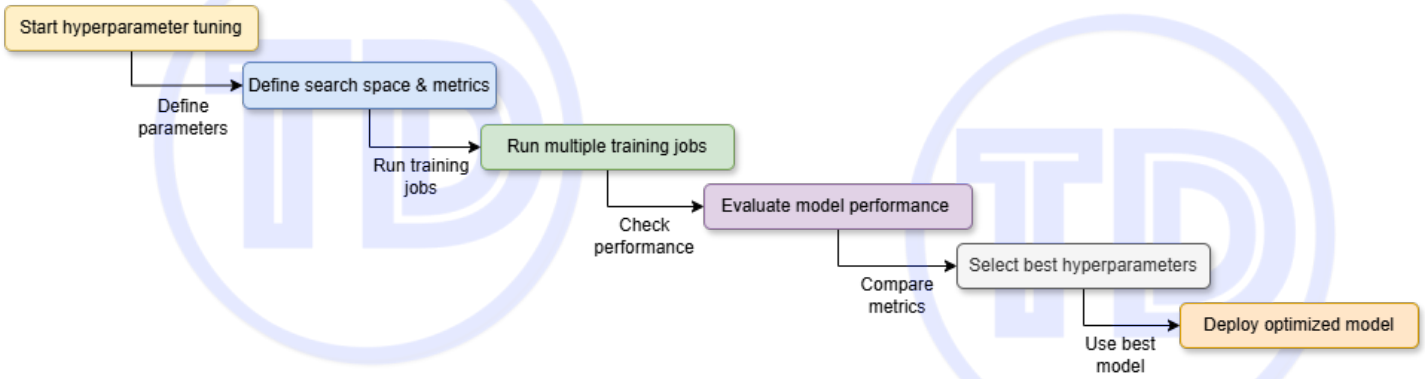
Using SageMaker Automatic Model Tuning (AMT)

This service automates the process of hyperparameter tuning with the use of Bayesian Optimization to look for the best-performing combination.

Key Features

- Parallel training: can test multiple configurations simultaneously
- Cost-effective: compute power is not wasted
- Automated: There is no need for manual training

Below is how SageMaker AMT selects the best hyperparameters over multiple iterations.



Example

In a recommendation system for online sellers, SageMaker AMT is used to find the best hyperparameters for their XGBoost model. Before using this service, it took a week just for tuning, now, it can happen within a few hours only.

Chapter 2.11 Model Hyperparameters and Performance

Understanding the impact of hyperparameters

Hyperparameters

These are the configurations or settings that control how your ML model learns. It depends on what your model is, but here are the most critical ones to consider:

- **Tree Depth (Decision Trees & Random Forests)** - the deeper it is, the more complex it will be, but it is prone to overfitting

- **Learning Rate** - how fast your model updates their weights; too high is unstable, and too low will take more time
- **Number of Layers (Deep Learning)** - more layers is equal to a more powerful model but computationally expensive

Example
The predictive maintenance system used for MRT/LRT trains used tuning of tree depth to avoid overfitting on historical train failures. With the right hyperparameter tuning, they got the best balance of accuracy and efficiency.

Tuning Models to balance bias and variance

- **Bias** (can lead to underfitting) - the model is too simple and can't capture patterns on the dataset; the model is also more likely to have more assumptions about the target or end result.
- **Variance** (can lead to overfitting) - the model is too complex and performs well on training data but not on unseen data.

The goal is to look for the best trade-off to avoid underfit or overfit.

Example
In a traffic congestion prediction system in Metro Manila, their previous model is high bias (can't predict rush hour peaks). With high variance, it has too many features, resulting to slower inference. This is resolved with hyperparameter tuning and feature selection.

Chapter 2.12 Integrating Models Built Outside SageMaker

Bring pre-built models into the SageMaker

This part will teach you how to bring pre-built models into the SageMaker environment for training and inference.

Amazon SageMaker
This service will support importing and running of pre-trained machine learning models built outside of SageMaker. You can use these pre-built models from TensorFlow, PyTorch, or Scikit-learn and then deploy them to SageMaker for inference and training.

Example
A Makati fintech company uses a pre-built fraud detection model built on the on-premises servers. For a faster

inference and scaling, they import these models to SageMaker and use its auto-scaling features to handle the sudden surge of transactions during the "sweldo" period (salary period).

Custom datasets for fine-tuning pre-trained models

This part is about using custom datasets for fine-tuning pre-trained models. You can use SageMaker JumpStart and Amazon Bedrock.

Fine-tuning

This method words by training your pre-trained model using your custom dataset. AWS SageMaker JumpStart and Amazon Bedrock have pre-built foundation models that you can fine-tune using your own data.

Example

A Philippine e-commerce platform fine-tunes a pre-trained chatbot model from SageMaker JumpStart using their customer interactions in Tagalog and Bisaya. With this, the chatbot is more natural with its responses to their local customers' inquiries.

Chapter 2.13 Regularization Techniques and Overfitting Prevention

Preventing overfitting and underfitting using regularization

Regularization techniques

As discussed in the previous section, these are used to avoid overfitting and underfitting.

- **L1 Regularization (Lasso)** - applies absolute value penalty on coefficients to drop some features.
- **L2 Regularization (Ridge)** - applies squared penalty to retain small weights and avoid over learning of noise from the data.
- **Dropout** - randomly disables some neurons during the training phase so that the model will be more generalizable.

Example

A real estate platform in the Philippines uses an ML model to predict property prices. At first, the training accuracy is high but not on real-world data. Using L2 regularization and dropout, the generalization of the model had improved, making it more accurate on new listings.

Managing catastrophic forgetting in Neural Networks

Catastrophic forgetting
This is a phenomenon wherein a neural network forgets its past learnings after being trained on a new dataset. This is a problem for lifelong learning and incremental model updates.

Possible solutions

- **Replay Memory** - old and new data are combined in the training set
- **Elastic Weight Consolidation (EWC)** - important weights are retained from the previous training
- **Progressive Networks** - use new network layer while the old layers are retained for past knowledge

Example
An AI chatbot in a bank in Manila is regularly updated to teach it new financial products. Before using replay memory, the chatbot forgets how to respond to previous questions about savings accounts. With Elastic Weight Consolidation, the chatbot can now retain its previous knowledge while learning new knowledge.

Chapter 2.14 Model Ensembling and Performance Boosting

Combining models using ensembling Techniques: Bagging, Boosting, Stacking

Model ensembling
These are techniques wherein predictions from multiple models are combined to increase the accuracy and robustness of the system.

- **Bagging (Bootstrap Aggregating)** - A lot of weak models are trained individually, then average their predictions (e.g, Random Forest).
- **Boosting** - Sequential training in which each new model enhances the errors of previous models (e.g., XGBoost, AdaBoost).
- **Stacking** - The outputs of multiple models are combined and inputted to a final meta-model for the final prediction.

Example
A logistics company in the Philippines uses machine learning to predict delivery delays. Using ensembling (bagging using Random Forest and boosting using XGBoost), their prediction accuracy increased, helping them optimize their delivery routes.

Chapter 2.15 Managing Model Versions and Repeatability

In machine learning, versioning and repeatability are necessary to ensure consistent model performance in production. Amazon SageMaker Model Registry is used to track the model versions and ensure they are compliant with auditing and reproducibility.

Using SageMaker Model Registry for Versioning and Auditing

Amazon SageMaker Model Registry

This is a fully managed tool used to track different versions of machine learning models. It's like a version control system similar to Git but only for ML models. You can track all your ML models across multiple private AWS and non-AWS model repositories in one central service, simplifying ML operations and ML governance.

Key Features

- **Model Versioning** - Each of the trained models gets a version for easy rollback.
- **Approval Workflow** - There is a feature for approving and rejecting model versions before they are used in production.
- **Tracking Metadata** - training job, hyperparameters, and model artifacts are automatically listed.
- **Auditing** - You can easily track who deployed which model and when.

Example

In an e-commerce platform, ML models are used for product recommendations. Each month, they train a new model with the latest transaction data, and using the SageMaker Model Registry, they are able to track the working ML models. If there is a drop in sales conversion, they can rollback to the previous version with better performance.

Managing model metadata for reproducibility and compliance

Metadata is the information about the model, such as:

- **Training data** - What dataset is used?
- **Hyperparameters** - What settings are used for training?
- **Evaluation metrics** - how accurate is the model?
- **Deployment logs** - When is the model deployed, and who issued approval?

With reproducibility, decisions made on ML models are easily traced, preventing unexpected errors or biases.

Example

A bank in Manila uses a fraud detection model for their credit card transactions. They have to track the model's metadata to avoid bias on other demographics (such as in different cities or regions outside Manila).

In case there's a customer complaint, the company can trace how the model processed its decision in a fraudulent transaction. Following the Bangko Sentral ng Pilipinas (BSP) regulations, they can oblige with the auditing feature of the Amazon SageMaker Model Registry.

C. Analyzing Model Performance

Chapter 2.16 Model Evaluation Techniques and Metrics

When there's a trained ML model, it should be evaluated to know how accurate it is. There are different evaluation metrics used depending on the type of model used.

Common Evaluation Metrics

Classification Metrics

- **Accuracy -** How many predictions are correct out of all predictions?
- **Precision** - How many true positive predictions are there?
- **Recall** - How many actual positive instances are detected?
- **F1 Score** - A combination of Precision and Recall, necessary for imbalanced datasets.
- **ROC AUC (Receiver Operating Characteristic - Area Under Curve)** - Shows how effective a model is in detecting positives

Regression Metrics

- **RMSE (Root Mean Squared Error)** - measures how big the error is in a prediction; the lower the error, the more accurate a model is
- **MAE (Mean Absolute Error)** - measures the average absolute difference between the actual and predicted values

How to interpret evaluation tools

Confusion Matrix

- **True Positives (TP) -** correct prediction of positive instance
- **True Negatives (TN) -** correct prediction of negative instance
- **False Positives (FP) -** incorrect prediction of positive instance
- **False Negatives (FN) -** incorrect prediction of negative instance

  Example

  If you are building a COVID-19 detection model, it's better to have high recall to ensure that no positive cases are overlooked.

Heat Maps

This is used to show the distribution of errors in different categories.

> **Example**
> In Sentiment Analysis, you can use this to see how correct the model's classification of sentiments (happy, sad, neutral).

Overall Application

An online lending platform uses a classification model to know which customer is eligible for loans. Using the confusion matrix, they can easily see how accurate the loan approval/rejection model is.

If the false positives are higher, some customers are not being approved even though they are eligible. On the other hand, having higher false negatives means customers are being approved despite not being eligible.

Chapter 2.17 Establishing Performance Baselines

Setting baseline performance using simple models or heuristics

Before training complex ML models, you need to set the baseline performance for a point of comparison. This could be simple statistical heuristics like mean prediction or rule-based models.

Example

A weather forecasting project in PAGASA can use the historical average temperature per month as its baseline model. Suppose the average temperature of April in Manila is 34°C. In that case, this can be used as a baseline for prediction and can be compared to an ML model that uses historical data and atmospheric patterns.

Comparing model performance against the baseline

After setting the baseline, the performance of the ML model will be measured using performance metrics such as RMSE (Root Mean Square Error) for regression and Accuracy/F1 Score for classification.

Example

In a loan approval model of a Makati bank, they use random guessing as their baseline model, which has 50% accuracy. If the improved ML model they have has 85% accuracy, it means it's better than the baseline and worth being deployed.

Chapter 2.18 Identifying Overfitting and Underfitting

Recognizing Signs of Overfitting and Underfitting in Training and Validation Sets

Overfitting

If the model performed well on the training data but not on validation and test data, it is overfitting.

Example: An e-commerce company built an ML model to predict customer churn. If they have only used the historical purchases of one person, it will overfit and won't be able to generalize on new data.

Underfitting

If the model's performance is weak during training and validation, it means it doesn't understand the patterns in the dataset.

Example: If they have only used a single feature, like age, to predict when the customer will leave, it is not enough and won't give them usable insights.

Chapter 2.19 Using SageMaker Clarify for Model Insights

Analyzing training data and model performance with SageMaker Clarify

Amazon SageMaker Clarify

This tool gives insights into how an ML model works and if there's bias in the data.

Example

In a job application ML model, the model may give the most importance to the school background of an applicant rather than their skills and work experience. With the help of SageMaker Clarify, they can confirm whether the model is based on graduates from the "big four" universities in the Philippines or not.

Identifying and mitigating model bias using SageMaker Clarify

Bias

An ML model is biased if there is an unfair advantage in a group, such as gender, age, location, etc.

Mitigation techniques
Examples are rebalancing the dataset, fairness-aware training, and post-hoc bias correction. They can be used for a fairer model prediction.

Example
SageMaker Clarify can be used for a loan approval ML model to ensure that their model are not biased against self-employed applicants (e.g., freelancers, street vendors, sari-sari store owners). Then, they can take action accordingly.

Chapter 2.20 Convergence Issues and Debugging

Diagnosing and resolving convergence issues during model training

Convergence issues
This occurs if an ML model fails to reach optimal performance during training. Reasons can include poor data quality, incorrect hyperparameter settings, or even inefficient optimization algorithms.

Debugging process
You can use this as a guide to understand the debugging process done during model training.

- **Issue identification** - check model accuracy, loss curves, and training logs
- **Data analyzation** - make sure your dataset is clean and balanced
- **Hyperparameters optimization** - adjust batch size, learning rate, or model architecture
- **AWS Tools** - You can debug efficiently with the Amazon SageMaker Model Debugger.

Example
A fintech startup based in Cebu trains its ML model to detect possible fraudulent credit card transactions. If the model's loss is not decreasing, they can check on convergence issues by analyzing outliers such as rare fraudulent transactions and implementing necessary actions.

Using SageMaker Model Debugger to Identify and Fix Training Problems

Amazon SageMaker Model Debugger
This tool can help you detect training anomalies and inefficiencies in real time. You can log and analyze model parameters, such as loss values and gradients, to prevent issues before deployment.

Example
A medium-sized sari-sari store wants to predict their holiday sales to prepare the stocks they need. However, the model overfits, which has predicted extreme demand fluctuations. With SageMaker Model Debugger, they discover weight initialization issues and fix them to stabilize predictions.

Chapter 2.21 Comparing Shadow and Production Model Performance

Comparing the Performance of a Shadow Model vs. a Production Model

Shadow model
This is a term for a newly trained model that is tested with a production model without affecting real users. Businesses can validate improvements before deployment with this.

Example
There is a telco company in the Philippines that wants to upgrade its customer churn prediction model. Before making their final decision, they will deploy a shadow model that runs parallel to the existing one and compare results for one month.

Techniques for Performing A/B Testing and Model Validation in Real-World Environments

A/B testing
This means that you will be deploying multiple models (or versions) in a live environment and then comparing their performance on real-world data. You can execute this via SageMaker Model Monitor and traffic-splitting strategies.

Example
In Manila, there is a ride-hailing app that wants to test a new surge pricing algorithm. Before making a decision, they split traffic between the old and new models, analyzing their users' behavior.

Domain 2: ML Model Development Sample Questions

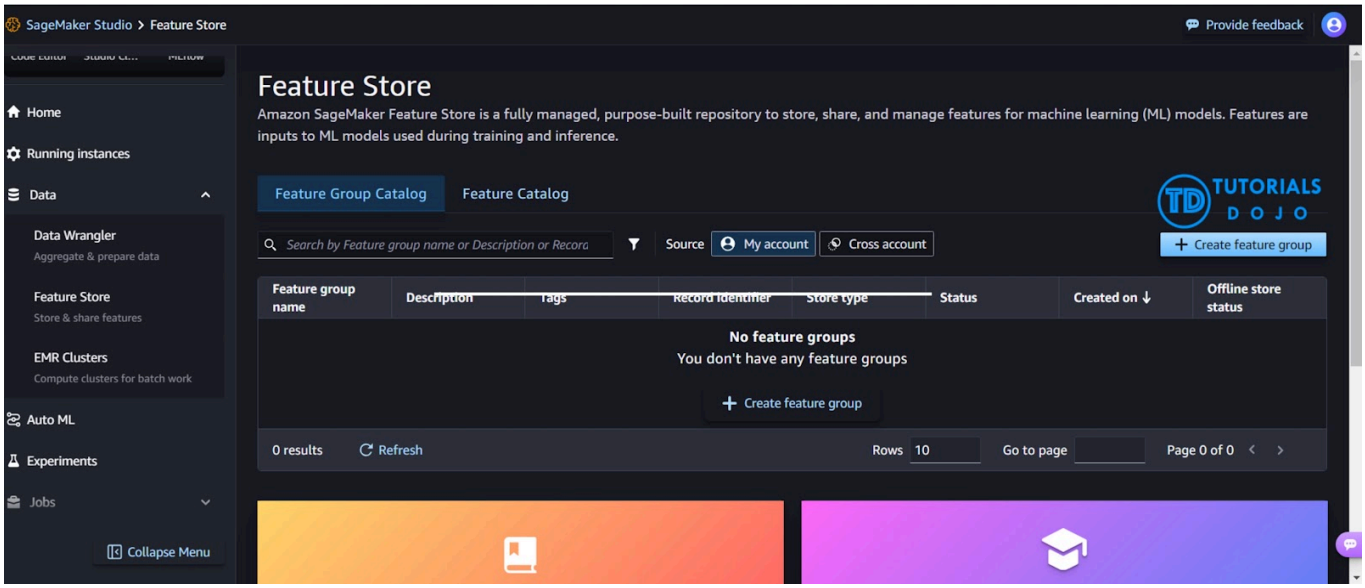
1. A data scientist working with the Amazon SageMaker Feature Store is required to incorporate a new feature into an existing feature group that contains historical data. The feature group is designed to support several machine learning models. The data scientist must ensure that the new feature is added and that all records, including historical ones, reflect this new feature.

   Which two actions should the data scientist perform? (Select TWO.)

   a. Apply the *PutRecord* command to update the records that are missing data for the new attribute.
   b. Set the new feature as the primary key to update all records automatically.
   c. Reindex the feature group to apply the new feature across all records.
   d. Execute the ***DeleteFeatureGroup*** command to remove the existing feature group.
   e. Utilize the *UpdateFeatureGroup* command to incorporate the new feature into the feature group. Specify the attribute name and type.

Explanation:

You can update and describe your feature group and add features and records through the Amazon SageMaker Feature Store API console. A feature group is an object that holds your data, while a feature describes a column in the database. When you add a feature to the feature group, you're essentially adding a column to the table. When you add a new record to the feature group, you are filling up values for features that are associated with a specific record ID.



To modify an existing feature group in the Amazon SageMaker Feature Store, you can use the `UpdateFeatureGroup` command to introduce new features. This operation allows you to add additional columns (features) to the feature group, ensuring that the feature group can evolve as your data requirements change. After adding the new feature, you can use the `PutRecord API` to update the records within the feature group. The `PutRecord` API enables you to ingest or overwrite records, ensuring that the newly added feature has data populated across both historical and new records.

Hence, the correct answers are:

- Utilize the `UpdateFeatureGroup` command to incorporate the new feature into the feature group. Specify the attribute name and type.

- Apply the `PutRecord` command to update the records that are missing data for the new attribute.

The option that says: **Execute the `DeleteFeatureGroup` command to remove the existing feature group** is incorrect because deleting the feature group would only result in the loss of all existing records, including historical data. The requirement is to maintain these records while adding a new feature. Removing the feature group contradicts this objective, as it would erase the entire feature group and its contents. Instead, the feature group should be updated without deletion.

The option that says: **Set the new feature as the primary key to update all records automatically** is incorrect because setting a new attribute as the primary key would not automatically update all records with the new

attribute. Primary keys are unique identifiers that cannot be easily changed or used to update records retrospectively.

The option that says: **Reindex the feature group to apply the new feature across all records** is incorrect because reindexing is simply not a valid operation in Amazon SageMaker Feature Store. You must manually update records using the appropriate commands, such as PutRecord, rather than relying on a reindexing operation.

References:
https://docs.aws.amazon.com/sagemaker/latest/dg/feature-store-update-feature-group.html
https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_feature_store_PutRecord.html

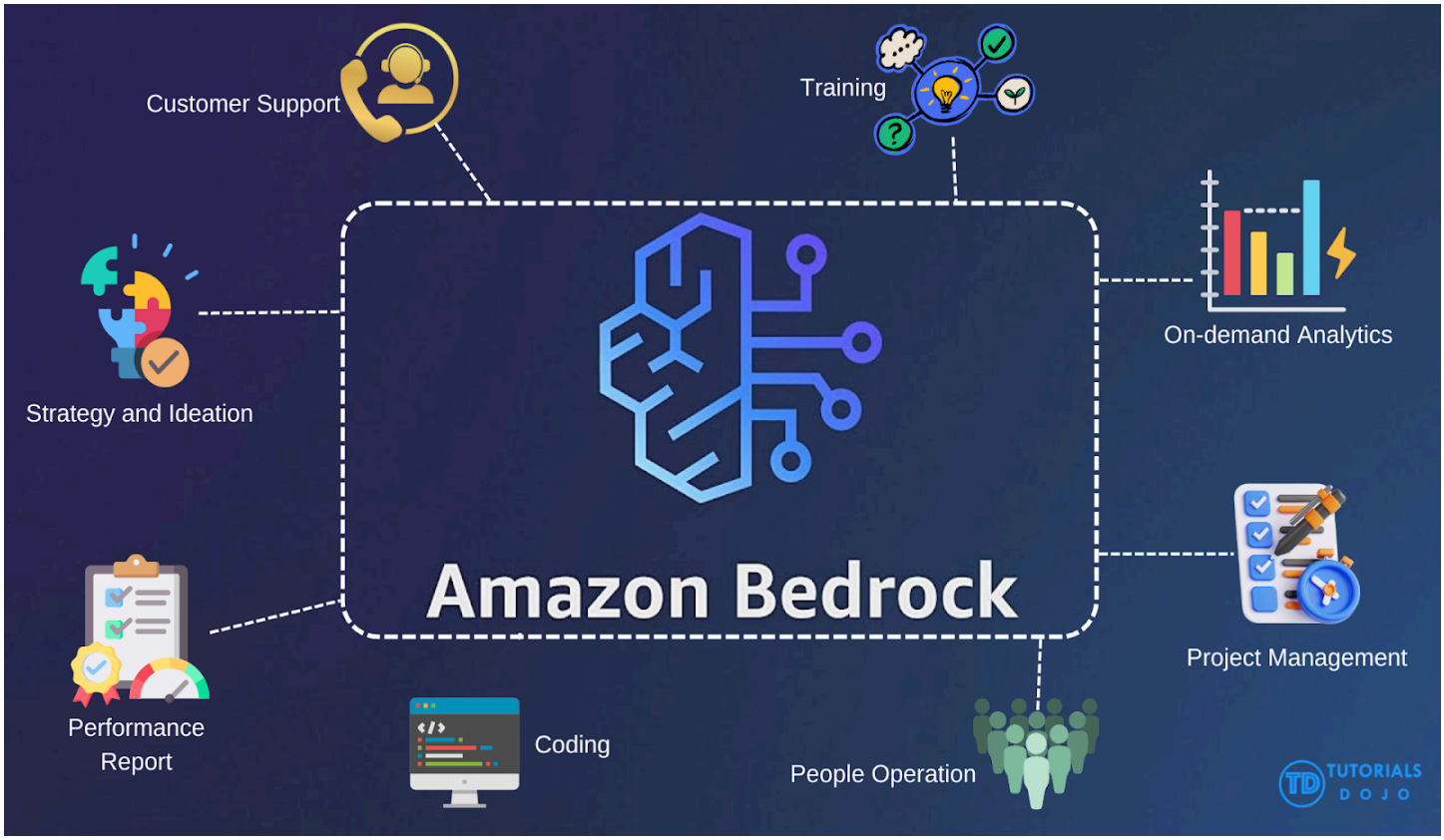
Check out this AWS Machine Learning and AI Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheets-aws-machine-learning-and-ai/

2. A large retail company is exploring generative AI to enhance its customer service chatbot. The company aims to fine-tune models from third-party services to enhance the chatbot's comprehension of business-specific customer inquiries.

   Which benefits of Amazon Bedrock will help the retail company meet its requirements? (Select TWO.)

   a. **Amazon Bedrock enables private customization of foundation models (FMs).**
   b. Amazon Bedrock requires customers to handle infrastructure provisioning.
   c. **Offers a wide range of pre-trained foundation models (FMs) from multiple providers.**
   d. Offers distinct APIs for accessing different foundation models (FMs).
   e. Ensures that prompts and responses remain completely private and secure.

Explanation
Amazon Bedrock is a service that provides access to foundation models (FMs) from leading AI providers. It allows businesses to integrate and scale generative AI applications without the complexity of managing underlying infrastructure. With Amazon Bedrock, companies can easily access various pre-trained models for tasks such as text generation, chatbots, image recognition, and more. This service offers flexibility in choosing the best models for specific use cases, enabling organizations to leverage cutting-edge AI technology to enhance their applications and customer experiences.



One of the primary benefits of Amazon Bedrock is its support for the private customization of foundation models. This means businesses can fine-tune pre-trained models to suit their unique needs and data better, ensuring more accurate and relevant outputs. Additionally, Amazon Bedrock ensures data privacy and security, as prompts and responses are handled securely within the AWS environment. This allows companies to deploy AI solutions while confidently complying with data protection regulations. Amazon Bedrock empowers businesses to innovate rapidly and effectively in the AI landscape by providing a broad selection of models and robust security features.

Hence, the correct answers are:

- Offers a wide range of pre-trained foundation models (FMs) from multiple providers.

- Amazon Bedrock enables private customization of foundation models (FMs).

The option that says: **Ensures that prompts and responses remain completely private and secure** is incorrect. While important, the scenario does not specifically highlight privacy as a concern for enhancing the chatbot's functionality. Therefore, it's not among the most directly relevant benefits for the given requirement.

The option that says: **Offers distinct APIs for accessing different foundation models (FMs)** is incorrect. This option simply describes a way of getting details about different foundation models available on Amazon Bedrock.

The option that says: **Amazon Bedrock requires customers to handle infrastructure provisioning** is incorrect. Amazon Bedrock is primarily designed to take care of the infrastructure provisioning and management. Customers do not need to handle the underlying infrastructure, allowing them to focus on developing and deploying their AI applications.

References:
https://docs.aws.amazon.com/bedrock/latest/userguide/what-is-bedrock.html
https://aws.amazon.com/bedrock/

Check out this Amazon Bedrock Cheat Sheet:
https://tutorialsdojo.com/amazon-bedrock/

References for Domain 2

Amazon Translate Documentation

Tutorials Dojo - Amazon Translate

Amazon Transcribe

Tutorials Dojo - Amazon Transcribe

Amazon Rekognition

Tutorials Dojo - Amazon Rekognition

Amazon Bedrock

Tutorials Dojo - Amazon Bedrock

AWS SageMaker Clarify

Machine Learning and AI Cheat Sheet

SageMaker Algorithms

Amazon SageMaker Cheat Sheet

SageMaker

Comprehend

TutorialsDojo - SageMaker

TutorialsDojo - Machine Learning

SageMaker Neo for Model Optimization

AWS Model Optimization

Model Training Basics

Tutorials Dojo: Machine Learning Concepts

Optimizing Model Size

SageMaker Distributed Training

Tutorials Dojo: Amazon SageMaker Training

Early Stopping

Tutorials Dojo: AWS Machine Learning Optimization

AWS Feature Engineering Guide

Tutorials Dojo: Feature Engineering

AWS Regularization Guide

Tutorials Dojo: ML Best Practices

AWS Hyperparameters Guide

Tutorials Dojo: Tuning Best Practices

AWS Hyperparameter Tuning

Tutorials Dojo: Hyperparameter Tuning

AWS SageMaker AMT Guide

Tutorials Dojo: SageMaker Tuning

AWS Bias-Variance Tradeoff

Bring Your Own Model to SageMaker

Amazon SageMaker Cheat Sheet

SageMaker JumpStart

Tutorials Dojo: Amazon Bedrock Guide

Regularization in Machine Learning

AWS Machine Learning Cheat Sheets

Training Models Incrementally in SageMaker

AWS AI and ML Study Guide

Model Evaluation Metrics

AWS SageMaker Debugger Guide

Tutorials Dojo: ML Model Debugging

Diagnosing Training Issues with SageMaker Debugger

Tutorials Dojo: SageMaker Debugger

Model Ensembling in SageMaker

Amazon SageMaker Model Registry

Amazon SageMaker Model Metadata

Amazon SageMaker Baselines

Avoiding Overfitting in ML

Hyperparameter Tuning on SageMaker

Amazon SageMaker Clarify

Mitigating Model Bias Using SageMaker Clarify

AWS ML Model Evaluation Guide

Tutorials Dojo: Choosing ML Metrics

Optimizing AWS ML Costs

Tutorials Dojo: Cost Optimization for ML

Shadow Deployment in AWS SageMaker

Tutorials Dojo: Model Versioning in AWS

AWS Guide to A/B Testing for ML

Tutorials Dojo: Model Validation Strategies

Hyperband with SageMaker

DEPLOYMENT AND ORCHESTRATION OF ML WORKFLOWS

Selecting Deployment Infrastructure for ML Models

Creating and Scripting Infrastructure for ML Models

AWS Resources we'll be using in this Domain:

For this part, it is expected that we'll incorporate principles from DevOps, data engineering, and machine learning. However, before we proceed with the actual discussion concepts behind deployment, let us first know the potential AWS services that might get involved in this domain. Feel free to go back to this list if you forget the definition and use case of a particular service.

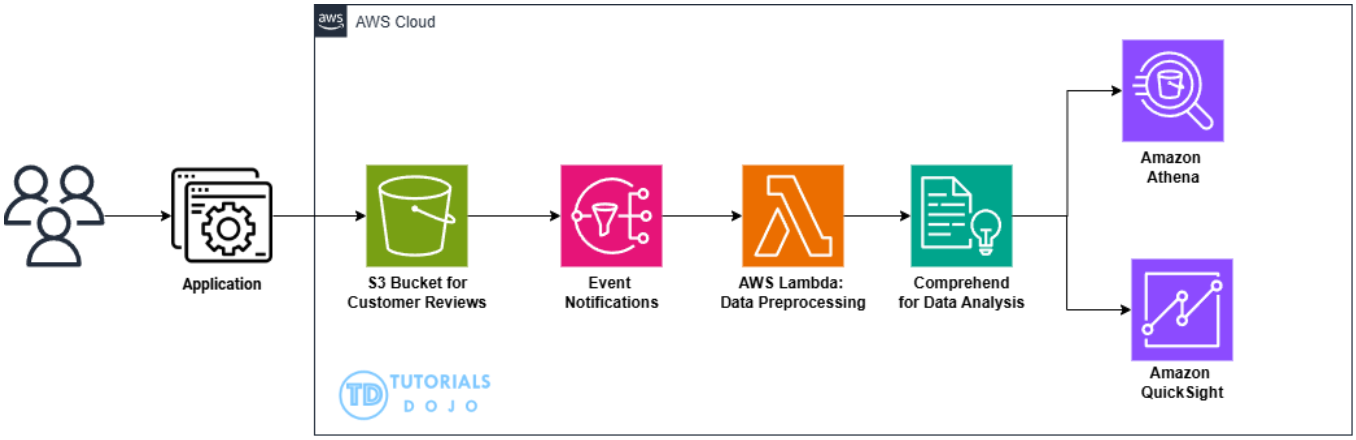
Amazon Comprehend

Uses **Natural Language Processing** to identify the semantics, insights, and features within a document. It can recognize entities, subjects, language (ex. Filipino), sentiments, or common topics within a written text. While Amazon Comprehend isn't directly related to infrastructure deployment, it is crucial for NLP workflows prior to deployment. While Amazon Comprehend isn't directly related to infrastructure deployment, it is crucial for NLP workflows prior to deployment.

Remember that Amazon Comprehend analyzes the following types of insights:

- **Entities** - Serves as the identifier of a particular topic in a document.
- **Key Phrases** - Phrases that are frequently occurring, or at least exist within a documentation.
- **Personally Identifiable Information** - Comprehend can detect potential data that can be identified as sensitive or personal information.
- **Language** - It can detect language, for example, Filipino.
- **Sentiment** - The set of dominant emotions that can be felt in a document
- **Targeted Sentiment -** The feelings associated with the specific context within a document. It can be good, bad, neutral, or combined.
- **Syntax** - parts of declamation for every word document.

Sample Scenario: Creating an NLP workflow for deployment, to be used in Athena for search queries and QuickSight for the dashboard, which will update automatically due to event notification and lambda functions.

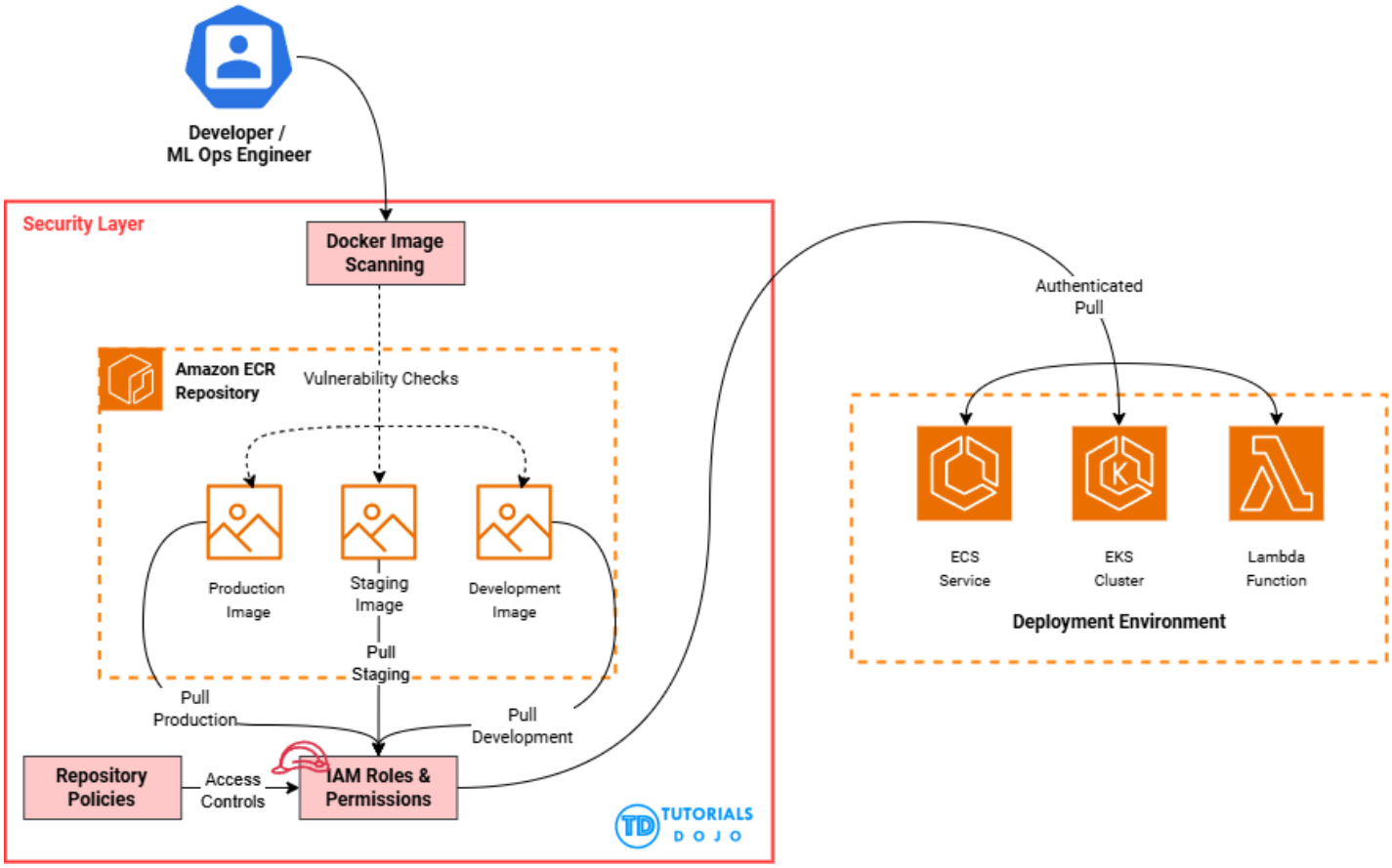


Amazon Elastic Container Registry (Amazon ECR)

Amazon ECR (Elastic Container Registry) serves as a way to store and manage **Docker Images** on AWS. For those who don't know, a **Docker image** is a read-only template for re-creating Docker containers, which includes the sufficing code, libraries, and dependencies to run software within a container. Hence, in ECR, you can store and version your Docker images for custom ML models and automation jobs. This is usually used with SageMaker training or inference jobs. Furthermore, it is integrated with ECS (Elastic Container Service), backed by Amazon S3.

Sample Scenario

A sample scenario would be a Developer using **Amazon ECR repositories** (Development, Staging, and Production), to separate the flow of container images within the deployment pipelines, and it is encompassed with security layers to preserve the security and integrity.



Amazon FSx for Lustre

Is a fully managed file storage solution that excels at high-speed tasks plus processes that manage huge datasets as efficiently as possible. A top parallel file distributed system called **Lustre** *(Linux + Cluster)* forms its base, which really fits the needs of high-performance computing (HPC), Machine Learning (ML), and Big data tasks. It can scale up to 100 Gbps throughput to individual client instances, higher IOPS, and sub-ms disk latencies.

Storage Options:

- **SSD** - Amazon FSx for Lustre offers **SSD storage** for rapid response for busy workload tasks plus fast processing of small random files
- **HDD** - Amazon FSx for Lustre offers **HDD storage,** which serves high-volume data tasks and handles large sequential files

Easy Data Repository Integration with Amazon S3
Amazon FSx for Lustre can link with Amazon S3 and lets you process data in the cloud with Lustre's high-performance file system. Once you connect it to an S3 bucket, FSx for Lustre shows S3 objects as files for easy access. The system lets you write data back to S3 along with repository tasks that help move data as well as metadata between FSx for Lustre and S3.

Can be used for Lustre and on-premise data repositories
Data processing workloads from on-premises to AWS cloud infrastructure can be done using Amazon FSx for Lustre and AWS DirectConnect.

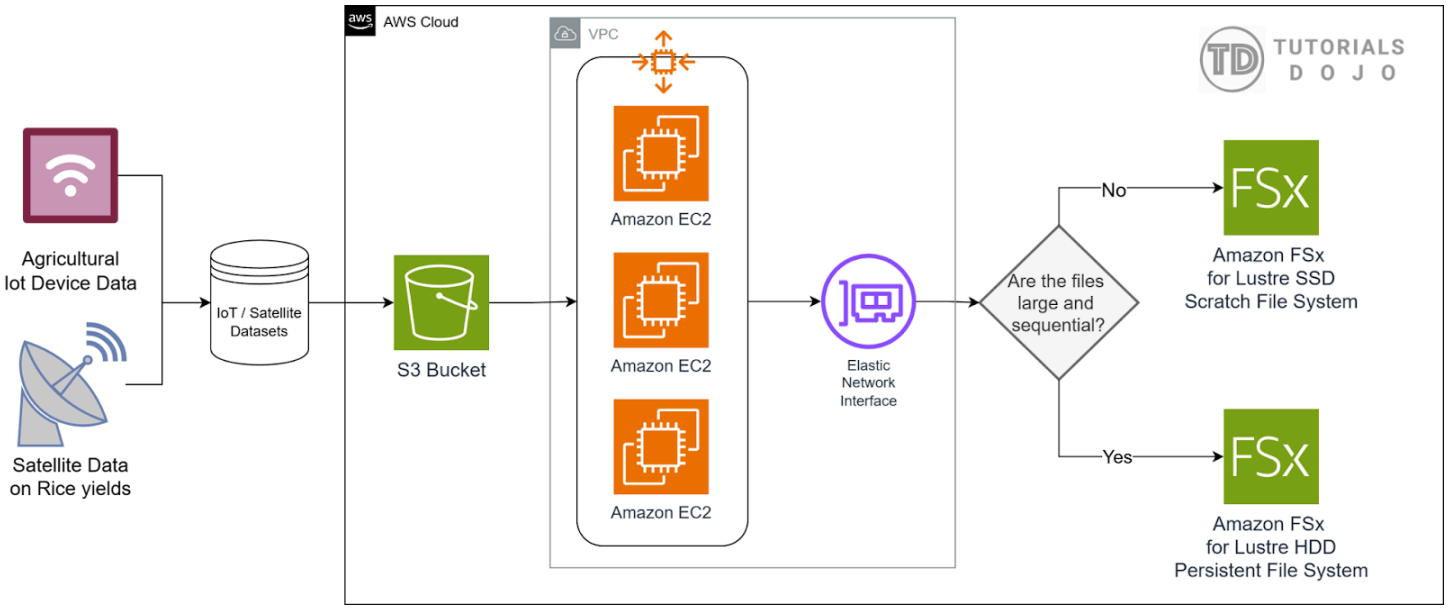
Multiple Deployment Options for FSX for Lustre

- **Scratch File System**
  - A temporary storage and shorter-term option for data processing.
  - Offers high burst throughput up to six times the baseline throughput of 200 MBps per TiB.
  - Use for **cost-optimized** storage for short-term processing.

- **Persistent File System**
  - Designed for long-term storage workloads
  - Availability Zones can replicate this file system.
  - Monitors frequent file system errors or hardware failures.
  - Use this for long-term and sensitivity to potential operational disruptions.

Sample Scenario

RicePhil, a governmental sector of the Department of Agriculture (DA), uses Amazon FSx for Lustre to process Large Datasets from IoT sensors and satellites for better crop yields. It must have high-performance computing and machine learning workloads.



For this workflow, viewing it from left to right, the sector first collects data from IoT devices in farms and satellite images stored in an Amazon S3 bucket, now entering the AWS environment. Second, the S3 bucket is connected to Amazon EC2 instances with auto-scaling and VPC to ensure the security and scalability of data. It will now undergo Elastic Network Interface (ENI) as a part of the network connectivity from the EC2 instance. Depending on the type of dataset, it will proceed to the Amazon FSx SSD Scratch File system if it is not sequential or HDD if it is sequential.

Amazon Lex (From Amazon Lex V1)

Amazon Lex is a service typically used for developing conversational user interfaces. It may include chatbots, virtual assistants, and using voice and text. It enables developers to create applications that can interpret both voice and text and are capable of interacting naturally.

Core Features of Amazon Lex:

- **Automatic Speech Recognition (ASR)** - converts speech into text, which leads to speech-based interactions.

- **Natural Language Understanding** - enables the machine to interpret the content being addressed by a human and respond to it in a manner similar to casual human conversations.
- **Multi-Turn Conversations** - Supports complex interactions consisting of multiple human language inputs before a conclusion.
- **Integration with AWS Services** - Can be integrated with AWS services such as Lambda functions, S3, or CloudWatch.
- **Context Management** - Can handle the context within the given conversations, giving more meaningful responses.
- **Omni-Channel Support** - You can integrate Lex with Slack bots, etc.

Common Concepts in Amazon Lex:

- **Bot** - serves as a point person for automating tasks, which may include triggering an ML workflow at a specific time. It can converse in natural language.
- **Intent** - Specifically indicates what it wants to do. Usually involves the intent name, Sample utterances, fulfillment of Intent.
- **Slot** - An intent accepts zero or more slots as parameters. You define these slots when you configure the intent. Amazon Lex requests the user to input specific values next to the slots at runtime. A user must provide all required slot values before the intent becomes fulfilled.
- **Slot Types** - Serves like a data-type for each slot, where you can use a pre-defined slot type or the reserved slot types AWS provides.

Example: GetCoffee Intent

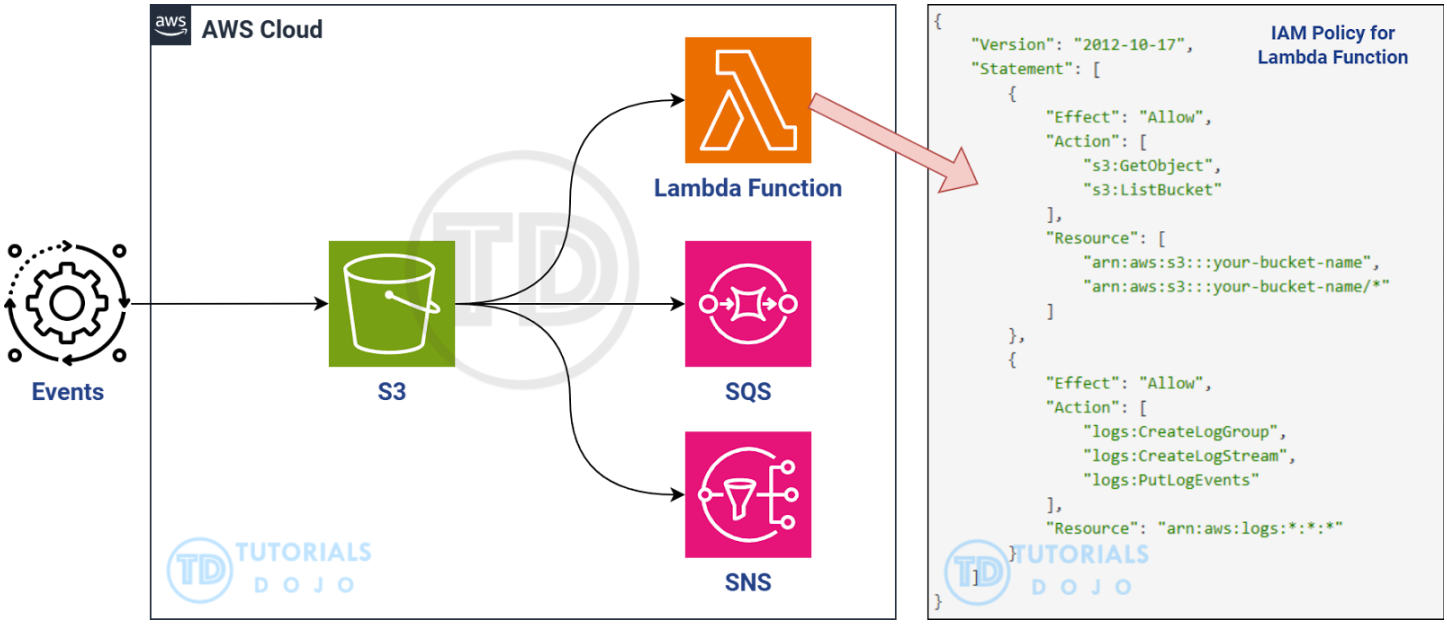
- **CoffeeType** - It may be espresso, Americano, French vanilla, etc.
- **SugarLevel** - What type of Sugar level, Low, Mid, or High?

Sample Scenario

A Philippine e-commerce company aims to automate customer support using Amazon Lex.

Amazon S3 Event Notifications

A feature that an S3 bucket can handle is S3 Event Notifications, usually in a span of seconds or minutes, wherein you can receive a notification if a particular event happens there. You can use it to set destinations such as sending notifications to a lambda function, Amazon SQS, or Amazon SNS to trigger an action. Note that you can create many event notifications as required.



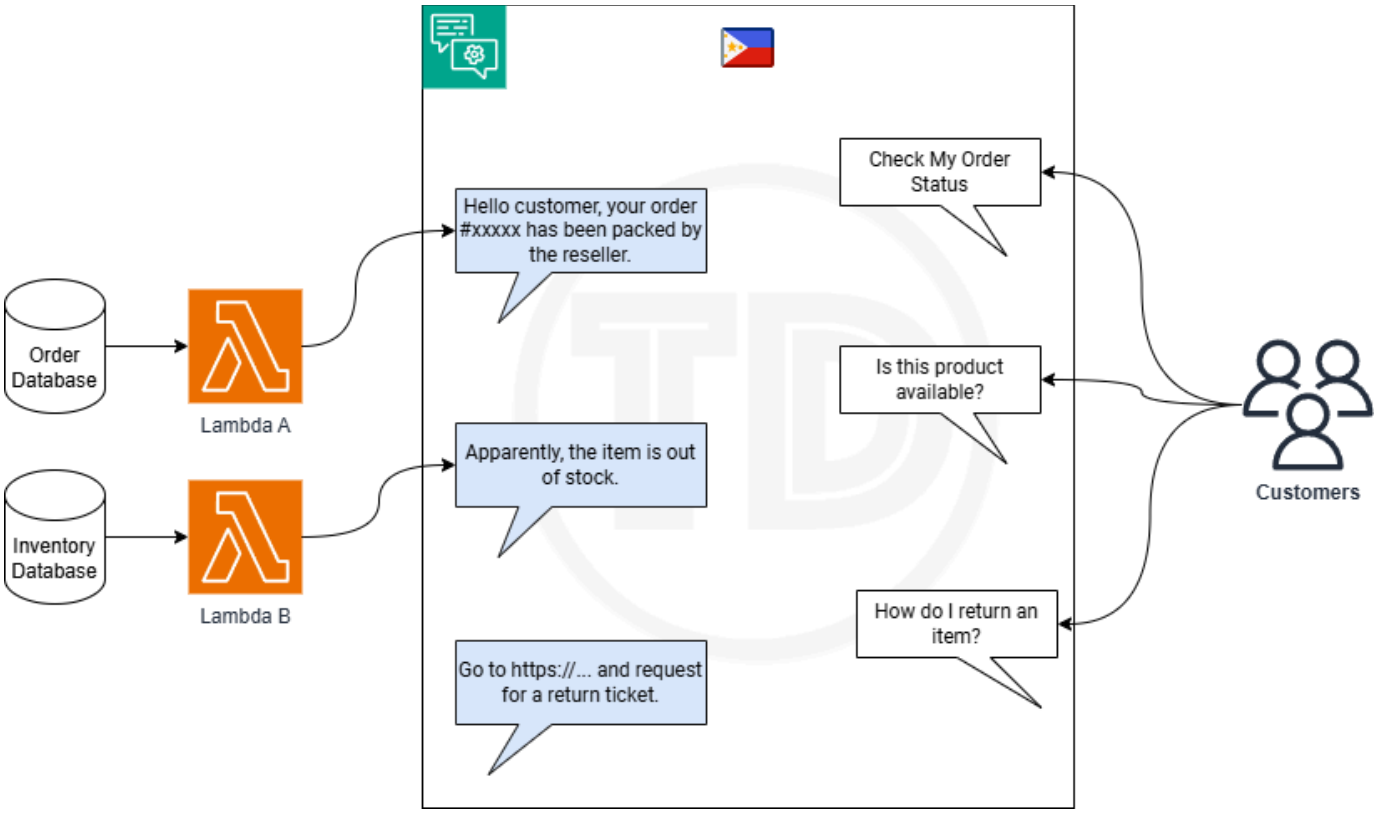
For instance, the sample code snippet on the right represents the IAM policy grants S3 to read objects and list the bucket, assigned to Lambda Function for creating a log group, log stream, and log events from CloudWatch logs. Here, the principle of least privilege is applied by limiting the S3 access based on its event notifications.

Sample Event Types (API Operations for S3 permissions):

- *s3:ObjectCreated:Put, s3:ObjectCreated:Post, s3:ObjectCreated:Copy* - Distinct API operations.
- *s3:ObjectCreated:** - an event type that requests a notification without taking note of the API that was used in object creation.

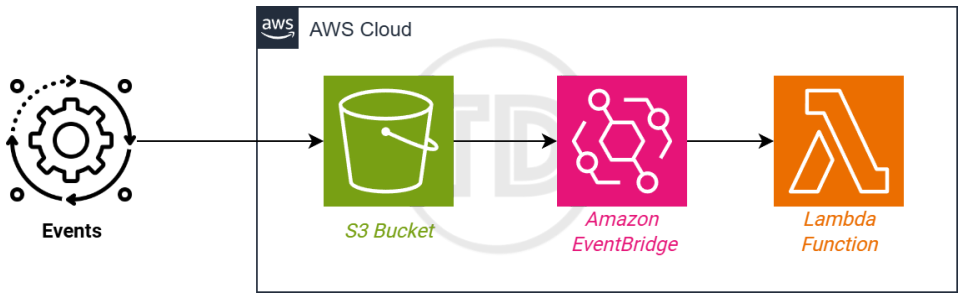
Sample Scenario

A Philippine-based e-commerce company called SariSari wants to automate customer support interactions using Amazon Lex Chatbot. The customers can inquire about the status of their orders, and Lex will send signals to Lambda functions for information retrieval. Then, the retrieved information from the respective database will be used as a response in return.



Amazon S3 Event Notifications with Event Bridge

This is primarily useful for creating event-driven architecture for AWS-based applications. The following shows how they are used.



S3 Event Generation

For every event, such as object creation, deletion, or restoration in S3, it provides notifications containing the event details and object modified.

Integration with AmazonEventBridge

S3 can be configured to send notifications directly to Amazon EventBridge, which is a serverless event bus, meaning it can receive events from AWS services to build event-driven architectures. This setup can withhold a flexible communication event management system, surpassing the traditional S3 event notification system.

Event Routing

EventBridge enables the creation of rules to route S3 events to various AWS services and custom applications, facilitating the development of intricate workflows and event-driven solutions.

Expanded EventTypes

Besides the common events that we can access from S3 objects, when connected to EventBridge, it can also provide events like object ACL changes, storage class transitions, and tagging updates.

Centralized Management

By acting as a central event bus for S3 and other AWS service events, EventBridge can make it easy for you to handle events across your AWS environments.

Enhanced Filtering and Routing

EventBridge allows advanced filtering using trails in events, enabling precise routing of events to specific targets.

Cross-Account and Multi-Region Support - EventBridge can handle S3 events, which can be sent across regions, accounts, and multi-accounts.

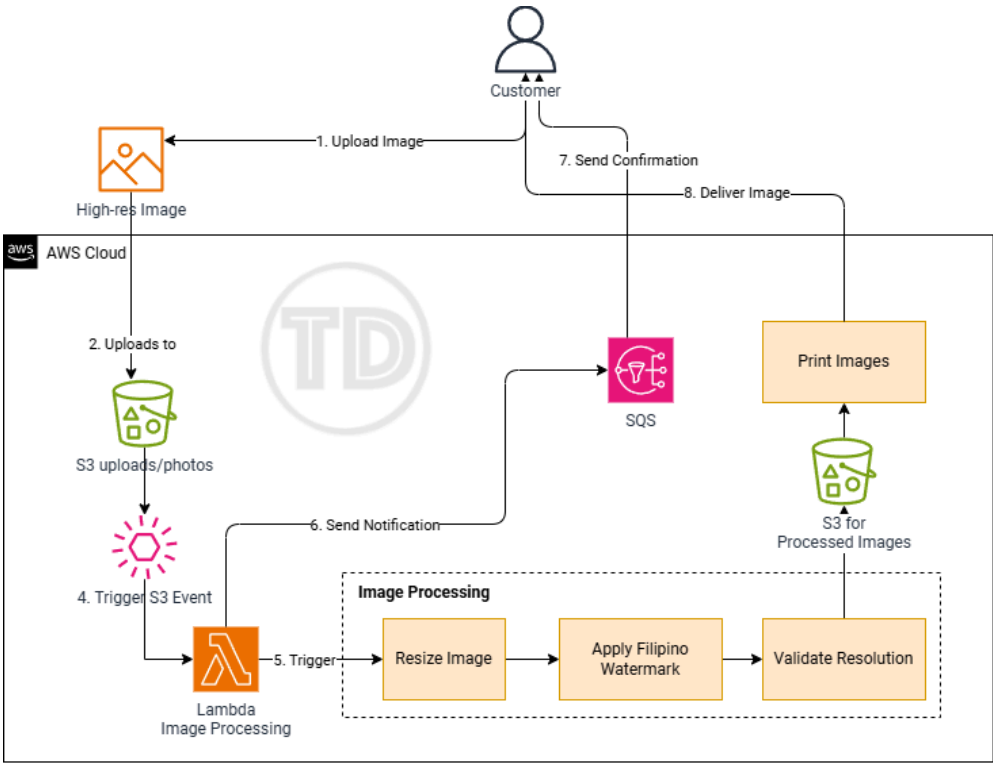
AWS Service Integration

This can be done simply by using S3 events, connecting it to the event bridge, then triggering actions in Lambda, Step Function, SNS, or SQS for serverless operations. Try this for an instance:

- Configure your S3 bucket to send events to EventBridge
- Set up EventBridge rules to select which events should proceed
- Pick the right targets, like Lambda or SNS for each rule

- Apply AWS guidelines to create secure permissions that protect all operations.

Sample: A Philippine-based online photo printing service wants to automate the image processing workflow whenever a customer uploads photos for printing. They use Amazon S3 Event Generation to trigger actions in real time as customers upload their files.



Amazon SageMaker AI

The next generation of Amazon SageMaker, which combines the concepts of data analytics and artificial intelligence, supports real-time, batch, and asynchronous inference, enabling flexible deployment options. The following services are a part of SageMaker AI, which we'll be discussing in relation to the Deployment and Orchestration of ML Workflows.

Amazon SageMaker Auto-Scaling Feature

Amazon SageMaker AI supports autoscaling for models hosted in the AWS environment. Auto Scaling means that it can accommodate dynamic adjustments of instances provisioned for a model in response to the density changes of workload. The higher the workload, the more instances are automatically provisioned, and the opposite happens if it lessens.

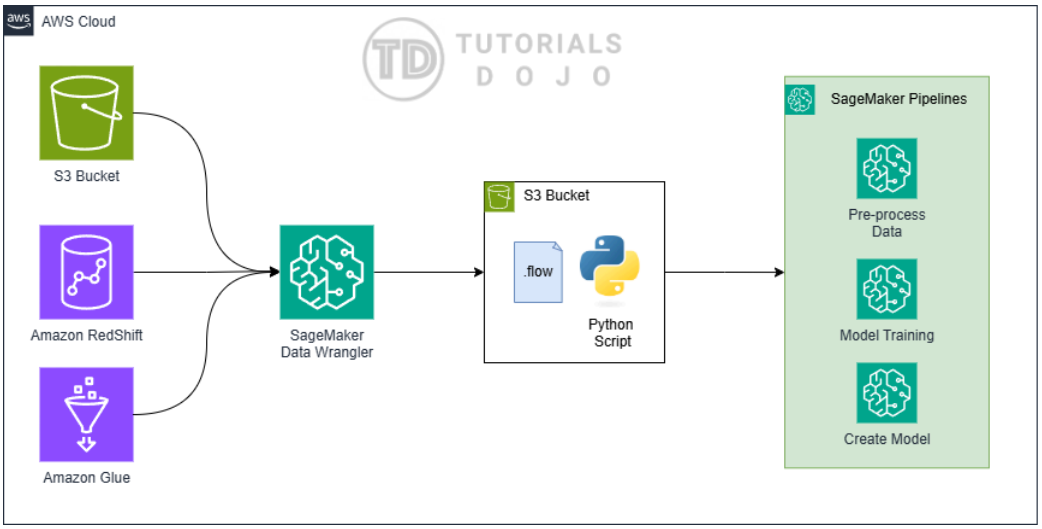
Amazon SageMaker - Data Wrangler

This service extensively supports ML workflow orchestration by automating the data preparation phase, a fundamental step before ML model deployment. ML engineers add it to Amazon SageMaker Pipelines to create automated end-to-end ML processes with ease.

Key Contributions of SageMaker Data Wrangler in Deployment & Orchestration:

- **Automated Data Preparation for ML Pipelines**
    - Data Wrangler allows you to import, clean, and transform data with little to no code. For better data to be fed.
    - It can be integrated with Amazon S3, AWS Glue, Redshift, Athena, and Snowflask for seamless data ingestion.
    - The transformed data can be directly used in SageMaker training jobs within SageMaker Pipelines.
- **Feature Engineering and Selection**
    - Data Wrangler provides 300+ built-in data transformation features so that it can offer the cleaning, normalization, and encoding of datasets seamlessly.
    - It supports **feature engineering** concepts, such as scaling, binning, and one-hot encoding for better model accuracy and bias reduction.
    - Processed features can be sent to **SageMaker Feature Store** for reusability across multiple ML workflows.
- **Integration with SageMaker Pipelines**
    - Data Wrangler workflows can be exported as pipeline steps in Amazon SageMaker Pipelines, enabling automation. The steps from the data wrangler, such as (data transformation feature extraction) become reusable components in the ML pipeline.

- **CI/CD for ML Workflows**
  - Data Wrangler integrates with AWS CodePipeline, AWS Step Functions, and Apache Airflow, ensuring a streamlined CI/CD process.
- **Model Monitoring & Data Drift Detection**
  - This allows teams to automate data preprocessing in production-grade ML models.



The diagram above shows the capabilities of SageMaker DataWrangler, wherein you can preprocess data coming from the data warehouse or Amazon S3, then create a model out of it.

Amazon SageMaker Feature Store

A fully managed repository for saving, fetching, and sharing ML features for various use cases. When we say "feature," it means the data points that represent a particular sector to be used as training data. It plays an important role in the deployment and orchestration of ML workflows by making sure feature engineering is streamlined and highly accessible for downstream tasks, such as feeding it to various ML models.

Key Contributions of SageMaker Feature Store in Deployment & Orchestration

- **Centralized Feature Management**
  - Feature Score can acquire and organize ML features for reusability across multiple ML models.
  - Enables standardization in feature settings for training and inference stability.
  - It can enable time-based queries, enabling querying in a specific timestamp.
- **Supports Online and Offline Feature Stores**
  - **Online Store:** Designed for low-latency access.

- **Offline Store:** It can store large-scale datasets, such as batch processing and training.
- It can manage syncs automatically, ensuring consistency between training and inference environments.

- **Seamless Integration with SageMaker Pipelines**
  - Feature Store serves as a step in SageMaker Pipelines.
  - Here, Step Functions or SageMaker Processing Jobs can play a role for automated feature transformations.

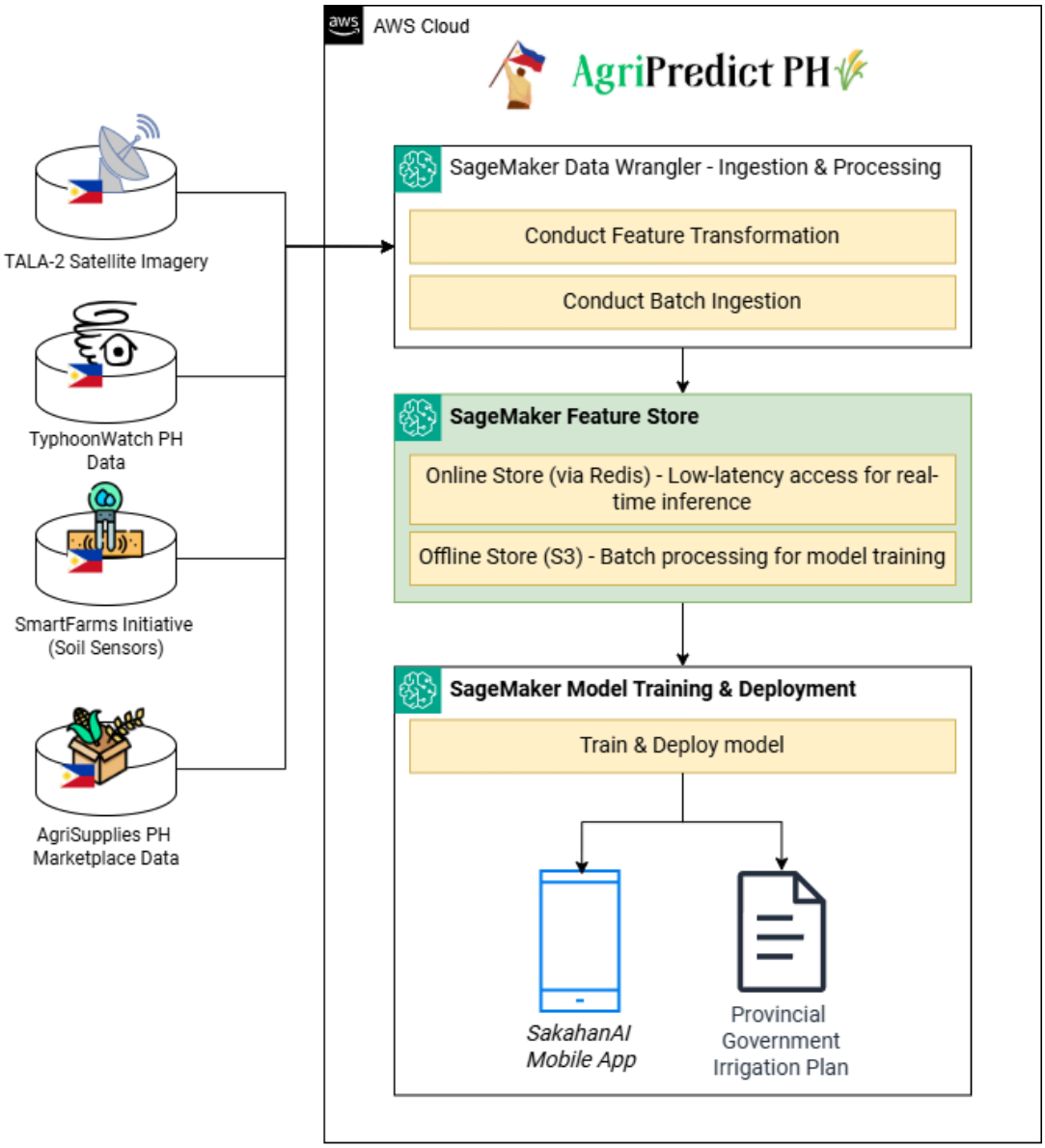
- **Enables CI/CD for ML Workflows**
  - It can be integrated with CodePipeline, AWS Step Functions, and Apache Airflow for a stable CI/CD workflow, which are tools for CI/CD, ensuring readability accessed features for training and inference.

- **Enhances Real-Time Model Predictions**
  - It reduces inference latency since it can fetch features from online stores that are precomputed rather than re-computation processes.

- **Feature Monitoring and Drift Detection**
  - Alongside the SageMaker model monitor, the SageMaker Feature Store accepts monitoring for potential data drift, meaning it can trigger when there is a significant change in model accuracy.

Sample Scenario

AgriPredict PH, a fictional startup from Cebu, uses the Amazon SageMaker Feature Store to help farmers predict crop harvest and cope with disasters. The business faces problems: isolated data, repeated features, plus mismatches between training records and live numbers that harm model accuracy and operation. The team takes steps to protect records and follow regulations. They use the tool to build one place for their machine learning features.



Key Features and Benefits from Search Results:

- **Centralized Feature Management** - through the help of SageMaker Feature Store, it provides AgriPredict PH with a central location for managing all their ML features, segregating the online store with Redis and the offline store with S3.
- **Feature Consistency** - it provides an assurance of consistency between training and inference by automatically replicating features from the online store to the offline store for model building.

- **Feature Processing and Ingestion** - AgriPredict PH can ingest data from various agricultural and environmental sources and integrate feature processing to transform data into ML-based features. They have an option to use SageMaker Data Wrangler to publish it directly to the feature store.
- **Feature Storage and Reuse** - Feature groups in SageMaker Store tags and indexes feature groups for better retrieval in the Amazon SageMaker Studio.
- **Real-time and Batch Ingestion** - Feature Store enables the ML system to perform streaming and batch data ingestion.
- **Security and Governance** - Feature Store integrates AWS Lake Formation for fine-grained access controls to protect feature store data and grant access based on role.
- **MLOps** - Stores a key component in the MLOps lifecycle.
- **Cross-Account Sharing** - Features extracted can be available to multiple accounts through cross-account sharing, discovery, and access.

Sample Feature Groups Online and Offline

- **Crop_Cultivation_Feature (Online Store)** - sample syntax

```
{
"cultivar\_id": "RC-1023-VisMin",
"typhoon\_resilience\_score": 0.87,
"water\_requirement\_liters": 1200,
"historical\_avg\_yield": "4.2 MT/hectare",
"best\_regions": \["Eastern Visayas", "Caraga"]
}
```

- **Regional_Fertilizer_Costs (Offline Store)** - sample output

Province	Unit Price (₱/kg)	Demand Surge Predictor
Nueva Ecija	52.50	0.78
Isabela	49.75	0.92
Bukidnon	55.20	0.63

AgriPredict PH uses SageMaker Feature Store to simplify its ML workflows. The system cuts the gap between training and serving, upgrades data records, and brings data experts and technical developers closer together. The option to reach and reapply features gives AgriPredict PH insight close to the truth and arriving at the right moment for Filipino farmers. That action raises crop production and readies farmers for hard times.

Amazon SageMaker "ModelExplainabilityMonitor"

Explainability is one of the most crucial parts of any AI model deployment. This is to ensure trust and clarify every decision made by a model. Hence, this is an ML service that can automate the monitoring of model explainability for transparency with regulations like GDPR and other financial regulations.

Key Features and Functions:

- **Explainability Reports with SHAP Values**
  - The tool applies SHAP calculations in feature attribution to determine how each feature affects predictions, along with reasons behind model decisions.
  - The feature attribution shows which factors (like a person's credit rating or salary) made the biggest impact on results.

- **Automated Explainability Monitoring**
  - After deploying models to SageMaker Endpoints, the ModelExplainabilityMonitor starts automatic supervision that creates detailed analysis documents for explainability, detecting changes or deviations in feature importance.
  - It also ensures that the model is clear, consistent, and reliable after deployment.

- **Integration with SageMaker Pipelines**
  - A system produces explainability reports as an automatic step in SageMaker Pipelines to monitor results after deployment.
  - The pipeline triggers alerts or starts retraining when feature values or model results show differences or a drift.

- **Compliance and Auditing**
  - Financial, medical, or government sectors need models that follow rules like GDPR or Fair Lending. The ModelExplainabilityMonitor creates clear feature reports so that regulators understand how the system made each decision.

- **Seamless Integration with Model Monitor**

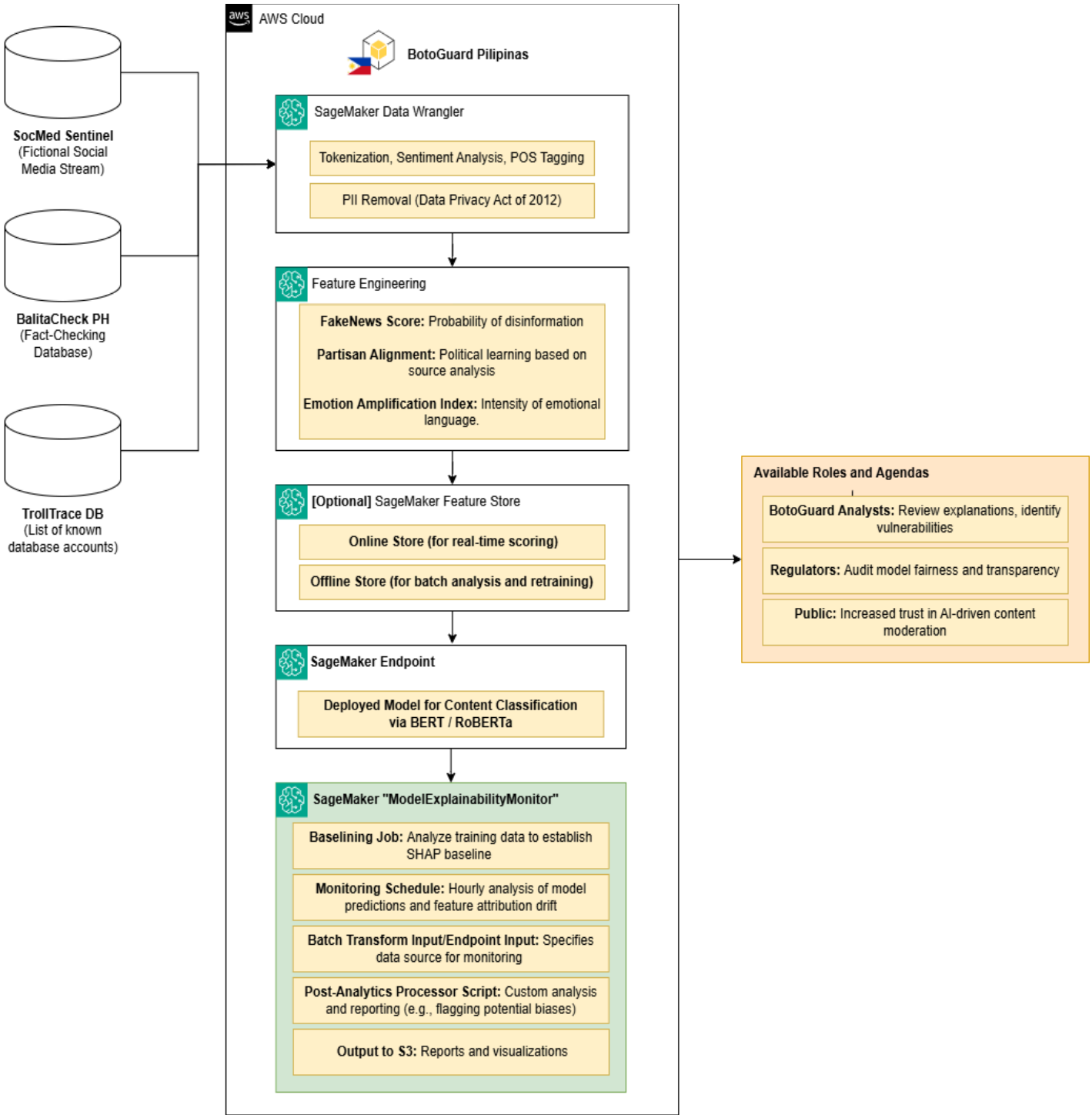
- The ModelExplainabilityMonitor works with other SageMaker tools through a data quality monitor; it looks for incomplete or wrong data. Next, a model quality monitor checks if predictions remain accurate. Lastly, the Bias Monitor finds unfair results in the output.

Sample Scenario

BotoGuard PH, a non-profit that finds and stops false information campaigns against Filipino elections, must know why its machine learning models mark some content as possibly false. They want clarity to earn trust with the public, a fictional COMELEC oversight board, and their own analysts. Basic metrics like accuracy or precision do not suffice; they need to identify which features of a social media post, news article, or video lead the model to choose.

Solution:

- **Use Amazon SageMaker Clarify** via '`ModelExplainabilityMonitor`' to receive information about the model's decision-making with potential bias. (See diagram on the next page)
- For instance, in the diagram, we can see a normal SageMaker workflow from data cleaning, feature engineering, and creating endpoints, but we can also see a "Model Explainability" workflow, which aims to monitor potential drifts from the model itself.
- A baseline has been established. Then, for every hour, it will return a result of potential drifts.
- A log will be provided and stored in S3.


SocMed Sentinel
(Fictional Social Media Stream)
BalitaCheck PH
(Fact-Checking Database)
TrollTrace DB
(List of known database accounts)
AWS Cloud
BotoGuard Pilipinas
SageMaker Data Wrangler
Tokenization, Sentiment Analysis, POS Tagging
PII Removal (Data Privacy Act of 2012)
Feature Engineering
FakeNews Score: Probability of disinformation
Partisan Alignment: Political learning based on source analysis
Emotion Amplification Index: Intensity of emotional language.
[Optional] SageMaker Feature Store
Online Store (for real-time scoring)
Offline Store (for batch analysis and retraining)
SageMaker Endpoint
Deployed Model for Content Classification via BERT / RoBERTa
SageMaker "ModelExplainabilityMonitor"
Baselining Job: Analyze training data to establish SHAP baseline
Monitoring Schedule: Hourly analysis of model predictions and feature attribution drift
Batch Transform Input/Endpoint Input: Specifies data source for monitoring
Post-Analytics Processor Script: Custom analysis and reporting (e.g., flagging potential biases)
Output to S3: Reports and visualizations
Available Roles and Agendas
BotoGuard Analysts: Review explanations, identify vulnerabilities
Regulators: Audit model fairness and transparency
Public: Increased trust in AI-driven content moderation

Amazon SageMaker Neo

Amazon SageMaker Neo converts machine learning models into improved versions that work efficiently on several device endpoints. The system automatically makes changes to each model and adjusts it to boost prediction speed without manual input. This approach fixes deployment plans that require efficiency and low cost.

Key Features and Functions:

- **Model Optimization for Hardware Platforms**
  - Neo changes models to work better on specific devices (CPUs, GPUs along with Edge units).
  - It accepts ML frameworks such as TensorFlow, PyTorch, MXNet or XGBoost.
  - Then, a user deploys models to AWS Inferentia, Intel, NVIDIA, ARM plus several edge systems.
- **Increased Inference Speed and Efficiency**
  - The process of Neo makes models smaller as well as less complex. A reduced size leads to quicker responses. The adjustments result in lower resource needs, which cuts costs on cloud or edge units.
- **Cross-Platform Deployment**
  - Models run on many platforms after one adjustment. The targets include cloud servers, IoT gadgets, besides local systems - no extra changes needed. The process fits into CI/CD pipelines for fast deployments to various platforms.
- **Integration with SageMaker Pipelines and Endpoints**
  - The Neo framework operates in Sagemaker pipelines, where it automatically deploys optimized models to Sagemaker endpoints. A user's inference requests respond much faster along with better speed results, especially in real-time apps.
- **Edge Deployment**
  - Neo excels at edge computing tasks or runs models on devices that possess limited storage plus processing power, e.g., mobile phones next to IoT sensors.
  - The optimized models integrate well into AWS services such as IoT Greengrass.

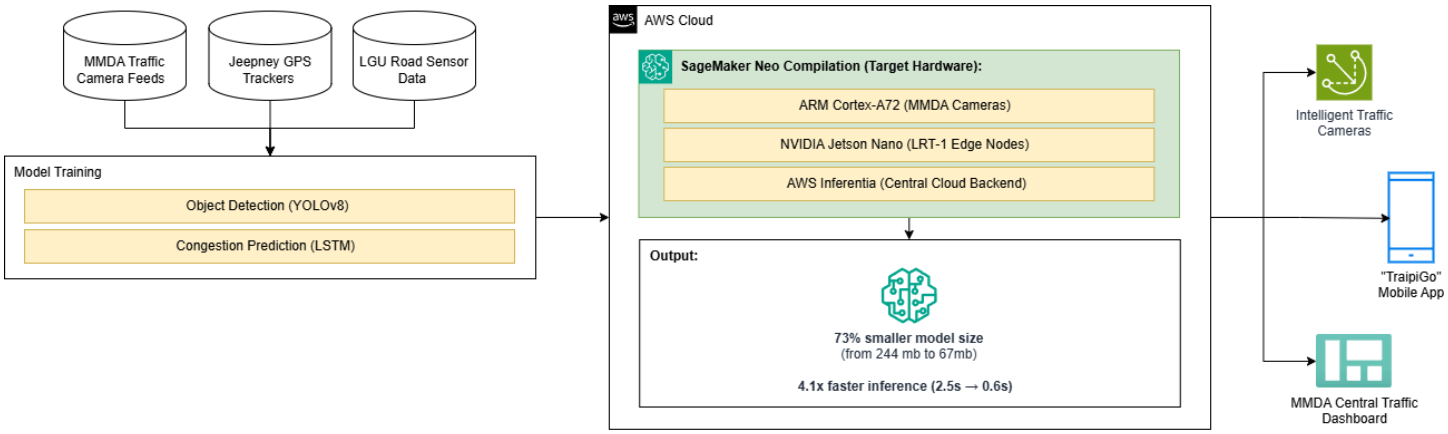
Sample Scenario

TrapiCam AI uses visual recognition models on local devices (traffic cameras, jeepney dashcams) to ease traffic in cities like Cebu and Davao. Their current YOLOv8-based model for spotting vehicles has trouble with:

- **High Delay:** 2.5-second waits on low-power traffic cameras
- **Device Differences:** Uneven work across devices (MMDA cameras vs. LRT-1 sensors)
- **High Cost:** Costly cloud GPUs for real-time traffic predictions

Solution:

The flowchart below describes the potential, improved model performance after passing it to the SageMaker Neo. Here, the optimized hardware for deployment has been identified, then, as an output, the model size was reduced by up to 73%. The inference got faster into 2 seconds only.



Amazon SageMaker Pipelines

Amazon SageMaker Pipelines is a CI/CD service dedicated to machine learning that controls the complete ML lifecycle. The service manages everything from raw data preparation to model release along with performance tracking. A highly structured service helps teams execute repeatable ML procedures with excellent reliability. The platform stands as an indispensable tool for organizations that need systematic as well as scalable machine learning operations.

Key Features and Functions

- **Workflow Automation**

- Amazon SageMaker Pipelines defines workflows for machine learning tasks in sequential steps based on the ML lifecycle.
- A typical workflow includes the following: data preparation, feature engineering, model training, evaluation, deployment, and monitoring.
- The platform includes conditional execution and branching, which adapt to multiple technical scenarios.

- **Integration with Other AWS Services**
  - **Data Preparation** - A workflow starts at Amazon S3, AWS Glue, and Data Wrangler to execute ETL (Extract, Transform, Load) tasks.
  - **Model Training** - The training process runs on SageMaker with built-in algorithms or custom code.
  - **Model Deployment** - Automatic deployment occurs to SageMaker endpoints, batch jobs, as edge hardware.
  - **Monitoring** - The SageMaker Model Monitor checks drift or problems and starts required actions.
- **Reusability and Version Control**
  - The SageMaker Pipelines system tracks versions and lets users reuse steps. Teams collaborate faster by sharing common procedures in various model types. Each project runs with parts from previous successful implementations. This modular structure reduces duplicate work or mistakes.
- **Integration with CI/CD Tools**
  - The service connects to AWS CodePipeline or AWS Step Functions. It also works with external tools such as Jenkins. This combination allows automatic updates of ML models. A change in data sets program code or settings leads to new model versions along with updates to production systems.
- **Scalability and Resource Management**
  - SageMaker resources appear on demand as needed for each pipeline. This includes training segments, access points, or data handling. The system runs multiple tasks at once, which makes ML procedures run faster.
- **Model Lineage Tracking**

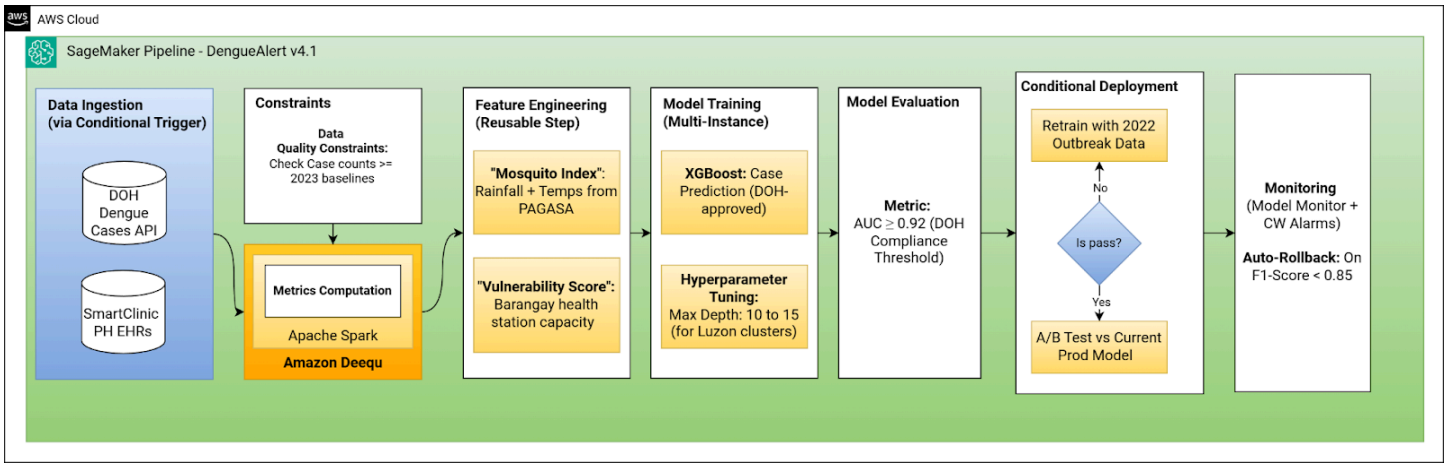
- The system records all details about models without manual input. This includes information like data versions, settings, or performance numbers. Each detail stays in a log for later checks, repeated runs, and regulations.

Sample Scenario

MediCare AI PH builds machine learning models that foresee dengue spread and assign ICU beds in Philippine hospitals. **Their manual fragmented systems result in:**

- **Late fixes:** A 3-week model update cycle during the monsoon season
- **Unsteady data:** Inconsistent electronic records from PGH, St. Luke's, and provincial hospitals
- **Rule issues:** Absent audit trails for DOH AI guidelines

Solution with SageMaker Pipelines:



Let's view the **SageMaker Pipeline** flowchart from left to right. First, in the Data Ingestion, we fictionally received an API from DOH about the anonymized cases of dengue across the Philippines and some electronic Healthcare data from various Hospitals. It will pass through Amazon Deequ for data inspection, checking the case counts if it started from the year 2023 onwards, then the metrics created from the condition will continue to further feature engineering. In this phase, new data shall be extracted, such as the Mosquito Index and Vulnerability Score, which serves as the KPI for our model.

Basically, it is a regression task for forecasting the number of dengue cases in a particular area and a classification task for identifying the ICU Bed Allocation. Next, in the model evaluation, we use AUC-ROC as a metric, with more than or equal to 0.92 for it to be accepted. If it fails, we retrain it; if not, we continue with AB/Testing. Then, lastly, further monitoring is needed.

How exactly did SageMaker Pipeline help?

- **Disaster Response Speed** - It can potentially cut the model from 21 days to 2 days only.
- A retraining function shall be automated in case the condition is not met.
- An automated rollback response has been created in case of anomalies.
- Enables parallel processing in 17 regional models, considering the scalability of AWS resources.

Amazon SageMaker Processing Jobs

Amazon SageMaker Processing Jobs handle data preparation along with feature engineering or model evaluation in a scalable setup. The service also manages post inference operations through automated infrastructure. A processing task coordinates each stage of ML procedures as well as executes large data computations in a very reliable manner.

Key Features and Functions

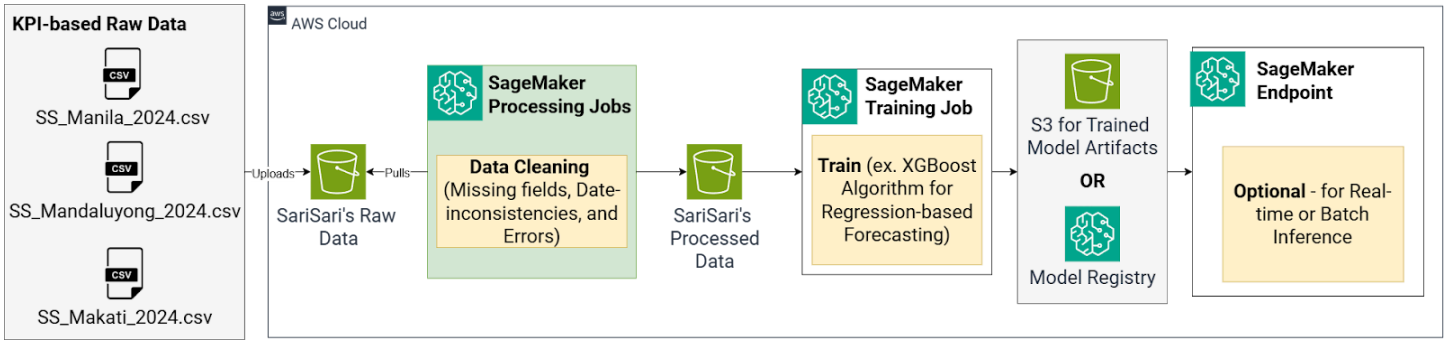
- **Data Preprocessing**
  - The system processes data through ETL tasks, which include the removal of errors or missing info from datasets, the conversion of raw numbers into a standard format along with data encoding, and the creation of additional dataset columns based on present information
  - A variety of formats work with the system: CSV, JSON, Parquet, as well as Avro.
- **Custom Scripts for Processing**
  - The platform runs Python, R, or Spark code on datasets via automated resources.
  - Users execute complex data changes without technical setup requirements.
- **Batch Data Processing**
  - The system manages extensive data tasks, such as converting historical info before AI model creation and analysis of big datasets after model release.
- **Integration with SageMaker Pipelines**
  - A Processing Job fits into a SageMaker Pipeline schedule or chain. The service runs data prep steps automatically, along with model training plus assessment steps.
- **Resource Scalability**
  - Automatically provisions and scales resources (CPU, GPU) based on the processing requirements. It ensures efficient handling of both small and large datasets.
- **Integration with AWS Services**

- Works with services like Amazon S3, Glue, Redshift, and Athena for seamless data ingestion and storage. Outputs from Processing Jobs can be fed directly into training jobs or batch transform jobs.

Sample Scenario

A client wants to implement an inventory forecasting model. For it to happen, he wants to analyze the growing grocery chain in NCR, Philippines, called SariSari. The client wants to implement a transactional data analysis from his stores in Manila, Mandaluyong, and Makati to forecast peak shopping periods, which he can use for inventory restocking and the most popular product from his stores.

His business intelligence team collects raw data from each store's customer transactions. These include daily sales, product information via SKUs, and customer purchase patterns, which is stored directly at Amazon S3. Upon checking, the datasets are filled with missing fields, date-formatting is inconsistent, and a lot of data errors, which are commonly caused by data entry issues. Also, the business analyst suggests implementing new features from the data to serve as the KPIs for the machine learning model. To help the team with their problem, you can do the following steps:



Using Amazon SageMaker Processing Jobs

1. **Data Ingestion** - the datasets in a form of large CSV files per branch are uploaded in a bucket named `s3://sarisari-raw-data/`
2. **Processing Script for SageMaker Processing Jobs**
   - A Python automation pipeline script was created by the business intelligence team to clean and standardize the data.
   - Now, the data can be cleaned by having a consistent date, removing errors, and fixing the invalid sales format.
3. **Setting Up the Processing Job**

- A processing job is now available, pulling data from an S3 bucket.
- A docker container has been created that contains the necessary Python Libraries.
- The processing job will provision the compute resources, for instance, automatically setting up an `` ml.m5.xlarge instance` `` for cleaning and feature engineering for large data.

4. **Output and Storage**

   - As an output, the cleaned data can be stored in an S3 location.
   - The cleaned data can be used for training a Job and then having an ML endpoint for inference.

Benefits:

- **Scalability** - the client can process several quantities of branch data in an effective manner via parallel processing.
- **Automation** - no need for manual scripts for provisioning the cloud servers.
- **Better Insights** - due to high-quality feature extraction, which came from the cleaned dataset, better insights from the model can be achieved.

Amazon SageMaker Serverless Inference

Amazon SageMaker Serverless Inference offers a deployment solution for ML models that scales automatically. The service adapts well to unpredictable traffic along with irregular request patterns. A team needs no infrastructure management or provisioning tasks, which leads to lower expenses as well as reduced complexity. The system takes care of the model setup, besides scaling operations, in a very straightforward way.

Key Features and Functions

- **Serverless Model Deployment**
  - The system places models on servers automatically. A user needs no instance setup or maintenance. This approach fits apps with irregular traffic loads or few requests.
- **Automatic Scaling**
  - Resources adapt themselves to workload peaks and drop to zero if inactive. Users pay just for actual processing minutes during predictions.
- **Seamless Integration with SageMaker Pipelines**
  - A pipeline includes Serverless Inference as one task, which puts models into production right after tests or training. Setup time drops significantly, plus workflows become simpler to handle and maintain.
- **Low Latency for Real-Time Inference**

    - The service delivers quick answers for instant calculations, which makes it perfect for interactive tools such as chatbots or suggestion engines.

- **Cost Efficiency**
    - A pay-per-use model considers processing time along with data volume, so businesses save money when traffic fluctuates.
- **Simplified CI/CD Integration**
    - The platform connects smoothly to AWS CodePipeline or an external CI/CD service. This allows automatic model updates as part of a continuous development process.

Sample Scenario:

A bakery shop in Valenzuela, Philippines, called "Juan's Delights," became significantly popular during festive holidays and local fiestas around the city. Besides its popular products, part of its success is by having developed a web and mobile application, which automatically assigns an order from their POS systems. The current server is just a rented, small data center from a local computer area built in 2020.

However, the dilemma lies down with the lack of server scalability. Now that the store has become popular in 2025, due to the unpredictable spikes that their application is receiving during peak times, the store owner wants to implement a pastry recommendation model.

It aims to be able to suggest seasonal or best-selling cakes to her online customers based on given criteria: (Preferences, Allergies, and Past Purchases, and Seasonality). The owner is planning to increase the rented data server to a larger size but doesn't want to pay for the idle servers if the traffic isn't at peak. A minimal infrastructure management is preferable, so the owner won't have to hire a dedicated software engineer and let her staff focus more on baking pastries.

Benefits of applying SageMaker Serverless Inference:

- **Easy Deployment**
    - The store owner simply hired a freelance ML developer to train a recommendation system uploaded to Amazon SageMaker for integration. Here, there is no need for a high-level data scientist because the instance configurations are already provisioned by SageMaker Serverless Inference.
- **On-Demand Scalability**
    - SageMaker Serverless Inference can scale up the provisioned instances to a higher level to handle traffic spikes during model inference.
    - During idle hours from the restaurant, serverless inference scales down to zero, which means, it doesn't incur any cost.
- **Low Latency for Real-Time Suggestions**

- When a user adds their preferences through the search bar and filters, it instantly responds with the most relevant suggestions.

- **Cost Efficiency**
  - Since serverless inference is a cloud component, you only pay the operational expense; whenever the model runs, that's the only time it will incur a cost.
- **CI/CD and Deployment**
  - They can use CodePipeline or CI/CD models to provision an automatic update once a model update has been released.

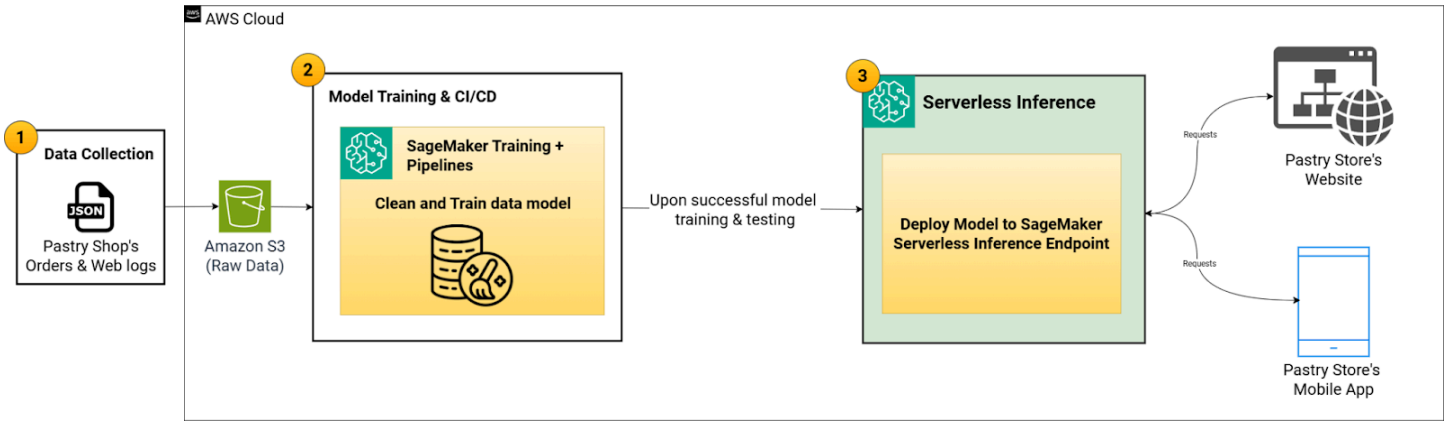


Diagram Explanation

1. **Data Collection**
   - The Pastry Store Website gathers all the customer information through a JSON file. It is then uploaded through Amazon S3. This will make the data available in the AWS infrastructure.

2. **Model Training & CI/CD**
   - A created pipeline through SageMaker Pipelines and potentially a CodePipeline.
   - Within the dataset, it underwent data cleaning and feature extraction to be fed in a particular model (e.g., XGBoost, GBM, etc.)
   - After a model has been created for training and testing, we upload it to SageMaker Serverless Inference.

3. **Serverless Inference Endpoint**
   - The Pastry Store's website sends requests to the serverless endpoint.
   - When the demand for recommendation systems increases, which leads to a surge, the Serverless Inference scales the infrastructure to manage the traffic.
   - On the other hand, it scales to zero if there's no request.

AWS CodeArtifact

A fully managed repository named AWS CodeArtifact maintains distributed code items as well as dependencies. These include program libraries, packages, plus custom components. ML engineers or developers use it to keep track of everything their systems need. The service helps teams control program versions in a very straightforward way. It also supports automation next to the smooth operation of machine learning processes.

Key Features and Functions

- **Centralized Dependency Management** - The system stores PyPI, Conda, npm, Maven or NuGet packages in one place. ML teams reuse standard libraries across different models, which creates uniform training plus deployment settings.
- **Version Control** - A precise package version system makes evaluation or training use matching dependencies. This reduces problems linked to unstable package combinations, also known as "dependency hell".
- **Integration with CI/CD Workflows** - The setup connects to AWS CodePipeline next to AWS CodeBuild, along with AWS CodeDeploy. It really speeds up dependency retrieval in SageMaker Pipelines or other CI/CD processes. The result: ML steps become highly repeatable.
- **Access Control and Security** - The service assigns detailed IAM permissions to determine each user's rights to post, change, or read artifacts. All connections stay encrypted to keep ML packages safe.
- **Cross-Account and Multi-Region Access** - Teams share packages securely between AWS accounts or regions, and large companies with remote ML teams really benefit from this feature.
- **Caching and Performance Optimization** - A local cache stores popular packages, which reduces delays when ML models need dependencies. Automated tasks run faster since packages install quickly in multiple jobs.

Sample Scenario

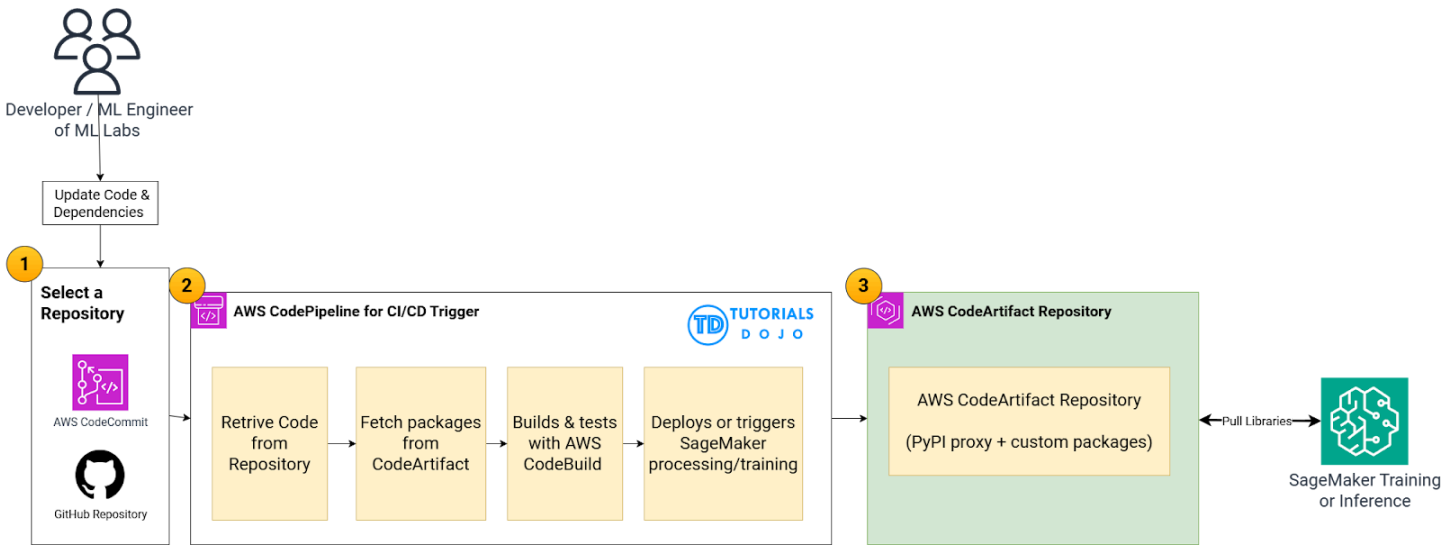
Jobert works as a team lead at a small startup company called ML Labs, known for providing outsourced ML projects for various companies. His team is constantly having a dilemma to manage consistent Python

dependencies, library versions, and custom packages in different environments, which includes both local devices, SageMaker Training, and CI/CD workflows. Each deployed ML service ends with incompatible library errors.

Key requirements to solve this pain point:

- Provide a centralized repository to streamline Python libraries and ML frameworks.
- Implement a consistent version control and document it as well.
- Enable simple integration with CI/CD pipelines (CodePipeline, SageMaker Pipelines)
- Configure your ACLs to properly distribute library access within AWS accounts.

Solution:



Based on the diagram, the Developer starts first by committing a code in a Python Script to a source repository, which can be in BitBucket or GitHub Repository. Next, the AWS CodePipeline for CI/CD triggers is important as well for detecting new commits and starting the CI/CD workflow. The CodeBuild phase helps you install the necessary dependencies and libraries, which were initially mentioned in CodeArtifact. Lastly, in the CodeArtifact repo, this is where our proposal mainly applies, where we internally store and proxy standard PyPi packages and include consistent versions for each build or training job. Lastly is the SageMaker, wherein we can train or infer the model, which will pull the exact libraries from CodeArtifact for a consistent ML environment across the testing phase until production.

AWS CodeBuild

AWS CodeBuild includes all functions needed to manage continuous integration (CI). It compiles code, executes tests, and creates build artifacts. The service helps automate machine learning tasks along with processes. It proves very useful when a project needs to orchestrate model training, run tests, package results, or deploy ML solutions through CI/CD pipelines.

Key Features and Functions:

- **Automated Build and Testing**
  - AWS CodeBuild runs the compilation of Python ML scripts along with automated tests to check model training pipelines.
  - The system performs unit tests, integration tests, plus data validation to maintain high code standards.
- **Integration with Sagemaker**
  - A direct connection to SageMaker Pipelines executes model training, evaluation, or deployment automatically. The service builds custom containers for SageMaker training as well as inference tasks.
- **Support for ML Dependency Management**
  - The build process includes downloads of required Python libraries from AWS CodeArtifact or public sources. Each pipeline run receives the exact package versions needed for execution.
- **Artifact Generation**
  - The process creates model files, Docker images, plus setup documents. A user stores these files in Amazon S3 or applies them in AWS CodePipeline for extra steps like deployment.
- **Parallel and Distributed Builds**
  - AWS runs several build tasks at once, which speeds up the training of models with varied hyperparameters and tests across multiple data sets or settings.
- **Integration with CI/CD Pipelines**

    - The system works alongside AWS CodePipeline and other AWS tools to manage ML procedures from code updates to set up. Source code changes or fresh data sets trigger new builds automatically.

- **Custom environment support**
    - The software accepts various build environments (such as Docker images) to mirror machine learning training setups. A stable or isolated workspace results in reproducible outputs. Developers establish consistent outcomes when they rerun identical code.

Sample Scenario

PalayPredict, a Filipino agricultural analytics startup, uses AWS CodeBuild to simplify machine learning tasks for models that predict rice yield in Luzon, Visayas, as well as Mindanao. By linking CodeBuild with SageMaker Pipelines and AWS CodeArtifact, PalayPredict fixes dependency issues, speeds up regional model training, and guarantees reliable builds for data from different regions.

Current Dilemma: Inconsistent Model Training for Philippine Rice Varieties

PalayPredict's ML models look at soil quality and weather trends next to past yields to help farmers plan when to plant. Their hand-run build and deploy work slowed things down:

- **Dependency Version Conflicts:** Developers picked different versions of Python libraries (for example, Pandas 1.5 and 2.0), which stopped the models from training.
- **Slow Regional Model Iterations:** They tested each set of hyperparameters for Luzon, Visayas, as well as Mindanao one after the other, and it took more than 12 hours.
- **Unreliable Artifact Packaging**: They built Docker containers by hand for SageMaker training jobs and furthermore often missed the scripts needed for data preparation.
- **Lack of Automated Testing:** They checked the soil sensor data only now and then, which led to wrong predictions.

Solution with CodeBuild: AWs CodeBuild-Powered CI/CD Pipeline

Step 1: Centralized Dependency Management with CodeArtifact

PalayPredict set CodeBuild to fetch Python dependencies from AWS CodeArtifact. This ensured that developers use the same library versions. A requirements.txt file fixed versions of key ML tools such as scikit-learn and TensorFlow.

Implementation:

- CodeBuild's `` `install` `` phase installs packages from a private CodeArtifact repository.

- If a build fails, Lambda sends alerts to developers when dependency resolution fails.

```
# buildspec.yml
phases:
  install:
    commands:
      - echo "Installing dependencies..."
      - pip config set global.index-url https://aws-codeartifact-login-endpoint
      - pip install -r requirements.txt
```

Step 2: Hyperparameter Adjustments for Regional Models in Parallel

CodeBuild's batch jobs ran training tasks at the same time for Luzon, Visayas, as well as Mindanao datasets using varied hyperparameters such as learning rates plus epoch counts. Every task applied a custom Docker image that copied SageMaker's training setup.

Workflow:

- **Signal:** A push to the regional-models branch starts three CodeBuild tasks side by side.
- **Tailored Build Environments:** Each task launches a GPU-enabled Docker container with CUDA drivers already in place.
- **Artifact Creation:** The finished models (.joblib files) and evaluation metrics (RMSE scores) are stored in Amazon S3.

Bash Command:

```
# AWS CLI command for batch builds
aws codebuild start-build-batch --project-name palay-regional-models \
  --environment-variables-override name=REGION,value=visayas,type=PLAINTEXT
```

Step 3: Automated Testing in addition to Validation

CodeBuild performs three sets of tests during the build step:

1. **Unit Tests:** Check that data preprocessing functions work correctly (for example adjusting pH levels).
2. **Integration Tests:** Confirm that model results match the outputs of SageMaker batch transform jobs.

3. **Data Quality Checks:** Remove datasets that lack soil nitrogen entries.

If any test fails, the pipeline stops immediately, and teams receive a message through Amazon SNS.

```
# buildspec.yml
phases:
  build:
    commands:
      - pytest tests/unit_tests.py
      - python validate_dataset.py --region $REGION
artifacts:
  files:
    - 'model_artifacts/**/*'
```

Step 4: Use SageMaker Pipeline

Approved models start SageMaker Pipelines via CodePipeline to:

- Place models on local SageMaker hubs (for example, mindanao-yield-predictor).
- Refresh Lambda-based farmer alert systems with new model versions.

Results:

- **Regional work became four times quicker:** Simultaneous builds cut model adjustments from 12 hours to 3 hours.
- **No clashes in dependencies:** CodeArtifact merge removed 92 % of errors that only happened on one computer.
- **Better accuracy:** Machine-driven data check boosted model correctness by 18 % on Luzon trial farms.

Conclusions:

AWS CodeBuild helped PalayPredict run ML workflows automatically while solving regional farming problems in the Philippines. By using batch builds CodeArtifact next to SageMaker integration, the team created models that train the same way every time, sped up cycles, and ensured deployments work reliably. This case shows how CodeBuild's CI power can expand ML solutions designed for different areas, plus farming needs.

AWS CodeDeploy

AWS CodeDeploy functions as a deployment automation service that moves applications or ML models to multiple destinations: SageMaker Endpoints, EC2 instances, and Lambda functions, along with edge devices. The service operates without manual intervention as well as places ML models on servers in a highly reliable manner. A very consistent process reduces system interruptions or failures. The automated sequence helps teams build plus maintain large-scale ML operations.

Key Features and Functions

- **Automated Model Deployment**
  - CodeDeploy automates how trained models move to SageMaker Endpoints or batch transform jobs along with real-time inference services.
  - The system connects to AWS CodePipeline for continuous delivery of updates.

- **Deployment Strategies**
  - **Blue/Green method:** A new model version starts up next to the active older version, which allows rollbacks.
  - **Canary release:** Traffic shifts step by step to new versions to check stability before full release.
  - **Rolling updates:** The system refreshes small groups of instances at once to keep services running.

- **Integration with SageMaker Endpoints**
  - The platform works together with the Amazon SageMaker Model Registry to move approved models automatically.
  - A tracking system maintains all model versions as well as deploys the newest and best performing ones.

- **Monitoring and Rollback Support**
  - Integrates with Amazon CloudWatch to monitor metrics such as latency, error rates, and inference throughput.
  - Automatically triggers rollbacks if the deployment fails or if performance metrics degrade.

Sample Scenario

A Filipino Startup named Rice Guard wants to use CodeDeploy to help them post their machine learning model predictions for different rice diseases. The ML Ops Engineers chose AWS CodeDeploy with Amazon SageMaker endpoints and CodePipeline, allowing Rice Guard to perform smooth updates and checkups next to send the system back to a previous state when problems arise. They can help farmers get accurate results even when they upgrade the model.

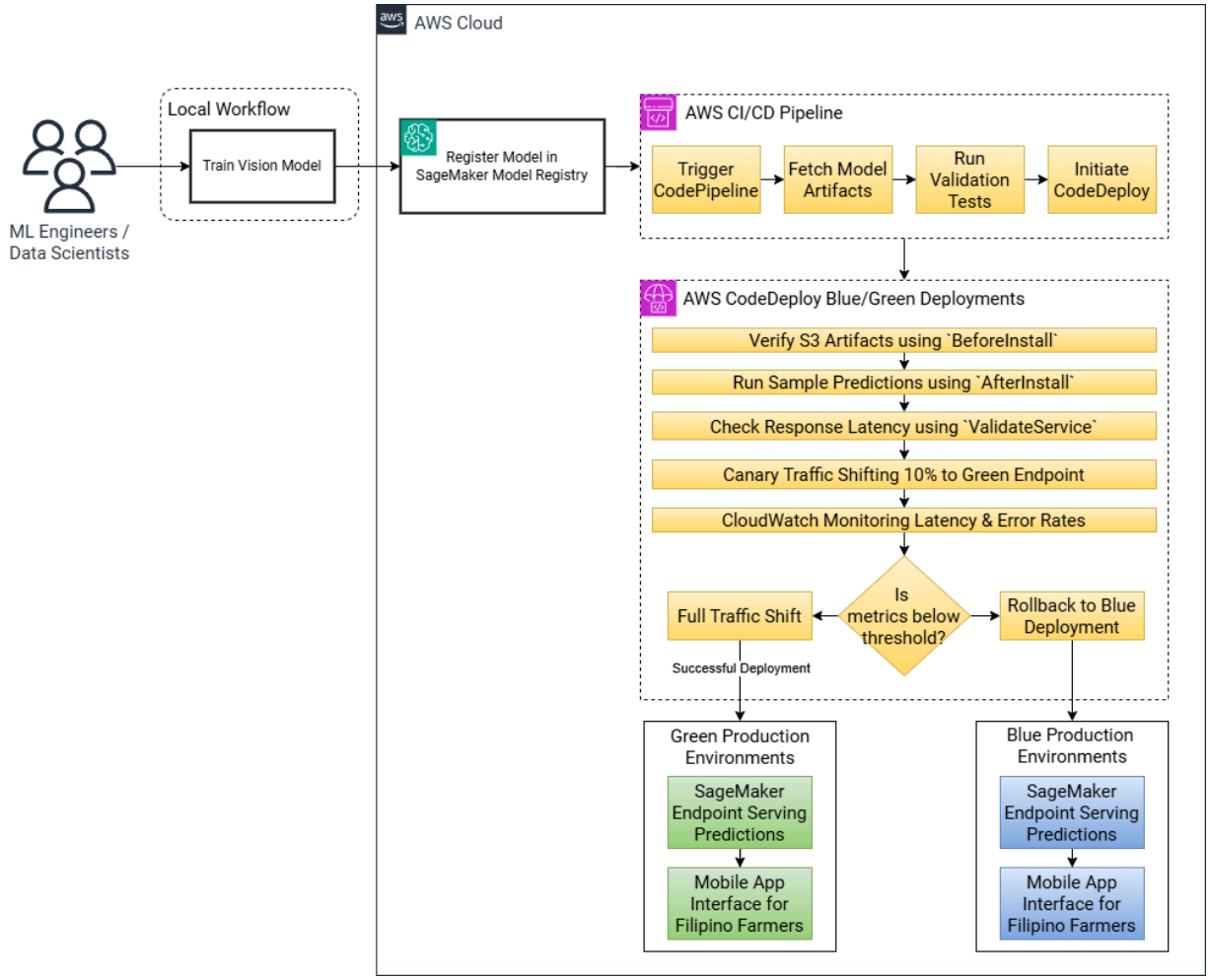
The Pain Point of RiceGuard's ML-Powered Crop Health Monitoring

RiceGuard works in Central Luzon, a key area for rice farming in the Philippines. They built a mobile app that lets farmers send photos that show rice problems caused by fungi and bacteria. The system then uses a machine learning model on Amazon SageMaker to check these photos for early hints of disease, such as bacterial leaf blight and brown spots next to sheath blight. When the model gets updated by hand, farmers face long periods without access and mixed-up results that slow down their actions.

Key Challenges:

- **Service Breaks During Updates** - Manually installing new changes led to service breaks that affected the productivity of farmers during important growing times.
- **Risk of Bad Launches** - New versions sometimes brought mistakes, and switching back took several hours.
- **No Quick Monitoring** - The team had trouble checking the model's speed besides its error counts.

Solution: Automated Deployment Pipeline with AWS Deploy



Stage 1: Blue/Green Deployment for SageMaker Endpoints

- RiceGuard chose AWS CodeDeploy's blue/green deployment strategy to prevent any halt in service. The current model ("blue") works together with the new one ("green") when updates occur. The system directs traffic only once an automatic check proves that the green model holds steady.
- **Model Enrollment:** Accepted models receive a version in the SageMaker Model Registry. A CI/CD pipeline starts when a new model is entered.
- **Integration with CodePipeline**: AWS CodePipeline collects the latest model files, runs tests to check their quality and begins a CodeDeploy rollout.

- **Traffic Shifting:** CodeDeploy sends 10 % of prediction queries to the green endpoint (test phase). If mistakes stay under 5 % for 15 minutes, the system moves all traffic there.

Sample Appspec.yml syntax

```
# appspec.yml for SageMaker endpoint deployment
version: 0.0
Resources:
  - TargetService:
      Type: AWS::SageMaker::Endpoint
      Properties:
        EndpointConfigName: "riceguard-v2"
        TrafficRoutingConfig:
          Type: ALL_AT_ONCE  # Shift all traffic after validation
```

Stage 2: Automatic Reversals with CloudWatch Alerts

CodeDeploy works with Amazon CloudWatch to check the green endpoint's performance. Alerts cause reversals when:

- Inference delay goes over 500ms (limit: 3 minutes).
- Error rates go above 10 %.

Python syntax

```
# CloudWatch alarm for high error rates
{
  "AlarmName": "HighErrorRate-RiceGuard",
  "MetricName": "5XXError",
  "Namespace": "AWS/SageMaker",
  "Threshold": 10,
  "ComparisonOperator": "GreaterThanThreshold",
  "EvaluationPeriods": 1
}
```

Stage 3: Checks Before Deployment

Before shifting traffic, CodeDeploy carries out checks to validate the new model:

- **BeforeInstall:** Looks for the model file in Amazon S3.
- **AfterInstall:** Uses a sample prediction to test its accuracy.
- **ValidateService:** Checks that the endpoint gives replies within a reasonable delay.

Bash script for CodeDeploy

```
# AfterInstall script for model validation
aws sagemaker invoke-endpoint \
  --endpoint-name riceguard-v2 \
  --body file://sample_input.json \
  --content-type application/json \
  --output output.json
```

Results:

- **Continuous Operation:** Farmers did not face service stops during updates, as requests moved smoothly between blue plus green endpoints.
- **Quicker Reversals:** Defective model versions changed within five minutes, instead of the previous two-hour wait.
- **Better Accuracy:** Instant checks helped RiceGuard improve the models and cut incorrect alerts by 22 %.

Findings:

AWS CodeDeploy helped RiceGuard run ML model updates automatically while keeping up with the steady work patterns needed in farming. Using blue/green deployments, lifecycle hooks, and CloudWatch, the team made sure that farmers got steady, correct results even when systems changed. This example shows how AWS CodeDeploy's automation tools can meet practical needs in fields where continuous service and careful work are essential.

AWS CodePipeline

AWS CodePipeline is a complete CI/CD solution that manages the entire process from the creation to the deployment of machine learning models. This service handles tasks like the construction, validation, or implementation of ML models without manual oversight. The system orchestrates ML processes in a structured way, along with timely retraining steps, which results in reliable workflows for developers. The service really excels at coordinating development cycles as well as maintaining consistency throughout model updates.

Key Features and Functions

- **Automated Workflow Orchestration**
    - Data cleanup or transformation of raw features (a.k.a feature engineering)
    - Step-by-step model training, along with quality testing and model packaging
    - Direct rollout to SageMaker servers, scheduled tasks, or remote units

- **Multi-Stage Pipeline Support**
    - In relevance to ML workflows, CodePipeline offers the following stages:
        - **Source** - A stage wherein it retrieves the updated source code or model artifacts, representing the model information itself.
        - **Build** - A stage that uses AWS CodeBuild to install the aforementioned dependencies, compile the source code, and run dedicated tests.
        - **Training** - Triggers **SageMaker Pipelines** or **SageMaker Processing Jobs** for model retraining.
        - **Deployment** - It has the capability to deploy models with **SageMaker Endpoints** using AWS Code Deploy.
- **Event-Driven Pipeline Execution**
    - The name itself means it automatically triggers an action or execution in response to events.

- For instance, it may trigger when new data is arriving in Amazon S3, code changes from a pushed repository, and updated model artifacts such as SageMaker Model Registry.

- **Integration with SageMaker Pipelines**
  - It can also orchestrate SageMaker Pipelines, ensuring that ML workflows will contain processes such as data preparation, model training, evaluation, and monitoring.
  - It has the capability to perform automation enhancements by enabling CodePipeline to manage workflow dependencies and task sequencing.
- **Continuous Deployment**
  - Continuous deployment/delivery of ML models to production environments became possible due to CodePipeline.
  - Strategies such as blue/green canary and rolling updates are now possible.
- **Artifact Management**
  - Artifacts, such as trained models, are now stored in Amazon S3, which enables ML pipelines to consume and produce artifacts, giving assurance to reproducibility during training, testing, and deployment phases.
- **Monitoring and Notifications**
  - CodePipeline can be integrated with Amazon CloudWatch and SNS for real-time inspection and alarms. Hence, if a stage fails, it will send a notification.

Sample Scenario

BayaniBank, a leading Filipino digital bank, leverages AWS CodeDeploy to deploy machine learning (ML) models that detect fraudulent transactions in real time. By integrating blue/green deployments with SageMaker endpoints and AWS CodePipeline, BayaniBank ensures seamless updates to its fraud detection system, which serves 5 million customers across Luzon, Visayas, and Mindanao.

Current Dilemma: Frequent Outages While Updating the Fraud Model

BayaniBank's mobile app depends on a machine-learning model to mark suspicious transactions, like ATM withdrawals that deviate from the norm or transfers with high amounts. Manual update methods led to severe problems:

- **Service Interruptions:** Installing new versions took over 30 minutes, and during that time, the system did not check for fraud.
- **Rollback Failures:** A broken update (v2.1) misclassified 12 % of regular transactions, and switching back to v2.0 consumed 2 hours of no service.

Solution: Pipeline That Uses CodeDeploy and Blue/Green Deployment

Step 1: Connect the Model Registry with SageMaker

BayaniBank uses AWS CodePipeline to run a four-part machine learning process:

- **Source:** When models get approved, the SageMaker Model Registry sends new versions.
- **Build:** AWS CodeBuild puts the model in a container with a GPU-friendly Docker image.
- **Training:** SageMaker Pipelines train models again each week with transaction data from Amazon S3.
- **Deployment:** CodeDeploy moves traffic among SageMaker endpoints with a blue/green plan.

Bash

```
# CodePipeline structure for fraud detection
Stages:
  - Name: Source
    Actions:
      - Name: SourceAction
        ActionTypeId:
          Category: Source
          Owner: AWS
          Provider: SageMaker
  - Name: Deploy
    Actions:
      - Name: CodeDeployAction
        ActionTypeId:
          Category: Deploy
          Owner: AWS
          Provider: CodeDeploy
        Configuration:
          ApplicationName: "fraud-detection-app"
          DeploymentGroupName: "prod-endpoints"
```

Step 2: Shifting Traffic with Lifecycle Actions

CodeDeploy runs check scripts during deployment steps:

- **BeforeInstall:** Make sure the new SageMaker endpoint handles 100 transactions in a simple test.
- **AfterInstall:** Looks at historical data to see if the false-positive figures of the green model (v2.2) and blue model (v2.1) differ.
- **ValidateService:** Checks that the green endpoint keeps its response time under 200ms during busy times (7–9 PM PHT).

```
# ValidateService script snippet
aws cloudwatch get-metric-statistics \
  --namespace "AWS/SageMaker" \
  --metric-name "ModelLatency" \
  --dimensions Name=EndpointName,Value=fraud-detection-green \
  --start-time "2025-03-02T19:00:00" \
  --end-time "2025-03-02T21:00:00" \
  --period 300 \
  --statistics "Average"
```

Step 3: Automatic Reversals with CloudWatch Alerts

CodeDeploy works with CloudWatch to check measures after changes are made:

- **Alert 1:** Starts reversal when the fraud check passes drop under 95 %.
- **Alert 2:** Undoes changes if the green point's CPU use goes over 80 % for 10 minutes.

CloudWatch Script

```
# CloudWatch alarm for success rate
{
  "AlarmName": "FraudDetectionSuccessRate",
  "MetricName": "SuccessfulPredictions",
  "Namespace": "Custom/BayaniBank",
  "Threshold": 95,
  "ComparisonOperator": "LessThanThreshold",
  "EvaluationPeriods": 2
}
```

Results

- **No Service Breaks:** Blue/green updates stopped service stops during 14 model changes in Q1 2025.
- **Quicker Fixes:** Bad updates (like v2.3's memory problem) came undone in 8 minutes instead of 2 hours before.
- **Better Accuracy:** Automatic checks cut wrong alerts by 17%, keeping ₱23 million per month in customer reimbursements.

Feedback:

AWS CodeDeploy enabled BayaniBank to automate fraud detection model deployments while maintaining 24/7 service availability for Filipino mobile banking users. By leveraging SageMaker integration, lifecycle hooks, and CloudWatch alarms, the bank achieved reliable updates and rapid rollbacks—critical for maintaining trust in a high-stakes financial environment.

AWS Step Functions

A fully managed serverless orchestration service, which enables you, as a developer, to coordinate the components involving ML pipelines, to serverless workflows for ease of automation process and handle end-to-end ML models, from preprocessing, training, evaluation, deployment, and monitoring.

Key Features and Functions

- **Orchestrating ML Pipeline Steps:** The Step Function enables you to create sequential, parallel, and branching workflows for ML tasks, which may include:
    - Data Ingestion and Preprocessing
    - Model training and evaluation
    - Model deployment to SageMaker Endpoints.
    - Monitoring, and that triggers retraining or additional updates.

- **State Machine Execution**
    - State machines are available in step functions with the purpose of defining and managing the flow for each step in the ML pipeline.
    - It may include invocation of SageMaker Processing Jobs, Training Jobs, or Lambda Functions.

- **Integration with SageMaker and Other AWS Services**
  - The system connects to SageMaker through basic integration methods to complete model training and deployment tasks. It links with AWS Lambda functions, S3 storage, SNS messaging, SQS queues, plus Step Functions activities. These connections let users run code segments or start other AWS services as part of their process flow.
- **Error Handling and Retry Logic**
  - Robust error handling, retry logic, and catch mechanisms to ensure that ML workflows
- **Visual Workflow Monitoring**
  - Step Functions provide a visualization for designing and ideating workflows. It became beneficial for teams to quickly understand and debug complex ML workflows.
- **Human Approval and Manual Interventions**
  - Supports manual steps in delegating workflows, with the inclusion of human approval before deployment, through SNS, SQS, or Lambda.
- **Cross-Service Orchestration**
  - Step functions help us coordinate interactions in multiple AWS Services for comprehensive orchestration of processes, starting from data preparation up to deployment.

Sample Scenario: Automated Dengue Outbreak Prediction for Metro Manila Public Health Using AWS Step Functions.

HealthWatchPH, a public health agency in Metro Manila, uses Step Functions to run dengue fever prediction models that work with real-time data from 17 cities. By running SageMaker jobs, Lambda data cleaners, and manual CDC approvals, HealthWatchPH cut outbreak response time from 14 days to 48 hours while handling unstable data quality common in Philippine municipal health reports.

Problem: Broken ML Workflows for Disease Monitoring

Metro Manila sees over 15,000 dengue cases each year. HealthWatchPH uses a manual ML process that faces critical problems:

- **Data Pipeline Issues:** 23 % of weekly reports from Quezon City and Caloocan lack key fields like patient age plus fever duration. This stops the model from starting.
- **Slow Human Check:** Health officers take 3 to 5 days to review model outputs before use.
- **Lack of Retry Options:** When SageMaker processing fails because spot instances end, the job must start over. This wastes 8 to 12 computer hours.

Solution: Step Functions Led ML Coordination

Stage 1: Reliable Data Collection Process

HealthWatchPH built a Step Functions state machine that has error checks to read raw CSV reports from three sources:

- **Amazon S3:** City health offices send daily case files.
- **Lambda Data Validators:** Look for needed fields with plain matching rules.
- **Parallel Recovery Paths:**
    - **Retries with Jitter** - Try re-reading corrupted files with retry rules that use a full jitter method and a maximum delay of 300 seconds.
    - **SNS Alerts** - Inform Manila CDC staff if verifications fail three times or more.

```
"Validate_Data": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "Retry": [{
    "ErrorEquals": ["Lambda.ServiceException"],
    "IntervalSeconds": 10,
    "MaxAttempts": 3,
    "JitterStrategy": "FULL",
    "MaxDelaySeconds": 300
  }],
  "Next": "Preprocess_Data"
}
```

Stage 2: Model Training with Conditional Branching

A Step Functions state machine orchestrates:

- **SageMaker Processing Job** - Cleans data using Scikit-learn imputation for missing values.
- **Model Training Parallelism** - Runs 3 competing algorithms (XGBoost, Random Forest, Prophet) via SageMaker training jobs.
- **Evaluation Gateway** - Selects the model with the highest F1-score using an AWS Batch-based metric evaluator.

```
# CloudWatch alarm for success rate
```

```
{
  "AlarmName": "FraudDetectionSuccessRate",
  "MetricName": "SuccessfulPredictions",
  "Namespace": "Custom/BayaniBank",
  "Threshold": 95,
  "ComparisonOperator": "LessThanThreshold",
  "EvaluationPeriods": 2
}
```

Stage 3: Human-in-the-Loop Deployment Approval

Step Functions integrates manual CDC reviews using:

- **SNS Notifications:** Alert epidemiologists when models achieve >85% accuracy.
- **SQS Wait States:** Pause workflow for 72 hours max awaiting approval.
- **SageMaker Deployment**: If approved, deploy to endpoints serving Barangay health apps.
- **Workflow Diagram Source:** Adapted from AWS Step Functions Documentation

Results:

- **Faster Outbreak Detection:** Step Functions cut down model update time from 2 weeks to 2 days.
- **Improved Resilience:** Jittered retries lowered data input failures by 67 % during monsoon network interruptions.
- **Cost Savings**: Running models side by side dropped SageMaker fees by 41 % compared to running them one after the other.

Conclusion

AWS Step Functions let HealthWatchPH run dengue prediction tasks automatically while solving the Philippines' difficulties with uneven health data and required human checks. By using state machines and decision splitting next to SageMaker links, the team built a system that can grow and be used for other illnesses, such as leptospirosis and measles.

Now that we have finished introducing you to the general AWS services for domain three, let's now identify how they are being used for a particular subtopic within this domain.

A. Selecting Deployment Infrastructure for ML Models

Chapter 3.1 Deployment Best Practices

As an aspiring Machine Learning Associate, having the right knowledge to deploy best practices in the context of Amazon Web Services is important for you to be prepared for the exam. When deploying machine learning models, especially for production-level use, versioning and rollback strategies are crucial to maintaining reliability and ensuring recovery in case of unwanted production failures.

Keep in mind that **versioning** refers to the proper model labeling for ensuring traceability, while **rollback strategies** refer to managing failed model deployments and rolling back to their baseline or benchmark versions. Luckily, AWS provides tools from Amazon SageMaker, AWS CodePipelines, and Amazon ECR that enable implementation strategies, which will be discussed later.

Versioning and ML Model and Artifacts

Remember that through **versioning**, each ML model, data pipeline, or automation script needs a proper version tag, which lets teams have a label that notes all modifications plus allows them to go back to older versions when needed. In AWS Cloud Infrastructure, they offer various services that enable versioning features for you to add to your documentation.

Why does it matter?

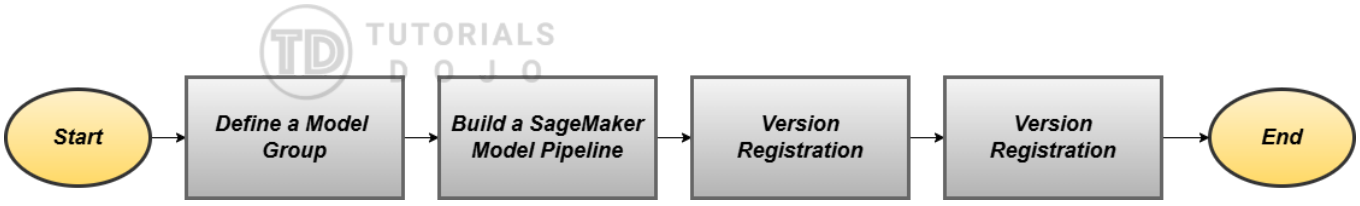
- **Traceability:** Quickly show what was altered between versions.
- **Reproducibility:** Always return to a particular model version to repeat outcomes.
- **Collaboration:** Allow several data scientists along with engineers to work together without disturbing each other's tasks.

How to Implement Model Versioning in AWS?

Amazon SageMaker Model Registry

Amazon SageMaker Model Registry acts as a streamlined repository for handling model versions, metadata, and approval statuses. It helps you deploy your models to production and automate deployment with CI/CD. Furthermore, it has an option to share models across different AWS accounts and implement it with SageMaker model cards.

According to AWS, a typical workflow of using SageMaker Model Registry would be the following:



- **First,** we begin by establishing a group to organize related models.
- **Second,** we develop a pipeline to automate the steps of model training and deployment.
- **Third,** we register each model's iteration per model within the model group after every execution in the pipeline.
- **Lastly,** we incorporate the Model group to the model registry, which are collections for centralized management and monitoring.

S3-Based Versioning for Artifacts – Besides the Model Registry, you save the training code, data, as well as other files in versioned S3 buckets or code repositories (GitHub, BitBucket, etc.).

SageMaker Projects (Optional Advanced) – Provides a fixed MLOps template (for example, pipelines, version control) for teams that want a solid DevOps approach.

Practical Tips for Versioning

- **Use simple names for each version -** For example, modelName_major.minor.patch or modelName_v1.0.
- **Set up automatic updates** - Use AWS CodePipeline or Jenkins to start new version entries every time you update the model.
- **Keep a record of the model's history** - Save hyperparameters training data pointers plus performance numbers so you see each change between versions.

Deployment Environments Best Practices (Test vs. Production Environments)

Part of creating innovative ML solutions is deploying them strategically for usability. As a common practice, ML engineers or developers usually settle first on two environments: testing and production. Hence, this section focuses on the best practices to properly provision and handle your cloud resources in several stages of ML model deployment for cost efficiency and security purposes.

Key Components and Features

- **Characteristics of Test Environment**
    - Resources can be in lower capacity, as long as it meets the requirements.
    - Security controls are highly flexible for testing.
    - Prioritizes development configurations.
    - Configured for rapid iteration for updates and testing.
- **Characteristics of Production Environment**
    - High availability requirements, depending on the client.
    - Must achieve SLAs if certain tickets arise.
    - Strict security controls to mitigate risks.
    - Prioritizes optimization for model performance.
    - Settled for high availability via auto-scaling capabilities.

Best Practices for Test Environment:

- Downsize your instance types when deploying in test environments.
- Leverage your spot instances to utilize unused resources efficiently.
- Enable instant or rapid deployment methods and rollback.
- Maintain lower costs by using pricing calculators and monitoring, especially when using on-demand instances.

Best Practices for Production Environments:

- Upscale your resources depending on the workload for efficiency and high availability.
- Auto-scaling groups are necessary to reduce downtimes.
- Implement reserved instances for continuous runtime.
- Set your cloudwatch alarms and logs for proper monitoring.

Implementation Considerations:

To improve security, it is important to use tight IAM roles in the live system. Test also live systems should reside within different VPCs. In addition, activate thorough monitoring of the live system and put into place correct data encryption.

- About scaling, correct auto-scaling rules are needed.
- Good capacity planning is a requirement.
- Observe how resources are used.
- Put proper load testing into practice.
- For cost control employ cost allocation tags.
- Set up budget alerts.
- Track resource usage.
- Choose optimal instances.

Primary Services:

Amazon SageMaker AI

- **Endpoints configuration** – Endpoint setup means you determine, besides handling, the points where your machine learning models connect to ask for data. It covers setting the endpoint label, its security, and its network.
- **Instance management** – SageMaker helps you to handle the EC2 instances. These instances deliver your endpoints. You simply select the kind of instance, set its storage, and hold its existence.
- **Auto-scaling policies** – You define automatic scaling rules. Such rules shift the count of running instances as demand alters. Your endpoints usually cope with incoming data but also it controls expenses.

AWS Auto Scaling

- **Capacity management** means you specify the number of instances you want active in your Auto Scaling group. Auto Scaling adds or removes instances to keep the number you set.
- **Scaling policies** automatically let you modify your Autoscaling group's capacity when demand changes. These policies ensure that the application handles every incoming request. It helps in cost control.
- For **target tracking,** you define the target value for a precise metric, like CPU use or request delay. Auto

Scaling modifies the capacity for the group to match the metric and the target.

Supporting Services:

Amazon CloudWatch

- **Monitoring and alerts:** CloudWatch allows you to monitor various metrics for your AWS resources, including CPU utilization, memory utilization, and network traffic. You can also configure CloudWatch to send you alerts when specific metrics exceed or fall below certain thresholds.
- **Metrics collection:** CloudWatch can collect metrics from a variety of sources, including AWS services, custom applications, and log files. You can then use these metrics to monitor the performance of your applications and to identify potential problems.
- **Log management:** CloudWatch can also be used to store and manage log files. This can be useful for troubleshooting problems and for auditing purposes.

AWS IAM

- **Access control:** AWS Identity and Access Management permits you to administer access to Amazon Web Services resources. You build user groups next to roles. You assign permissions to these items. These permissions define what actions they execute on your resources.
- **Role management:** IAM roles permit you to temporarily access AWS resources to items lacking permanent AWS credentials for role administration. This proves helpful when you provide access to apps or services. These apps or services require temporary access to your resources.
- **Security policies:** IAM policies exist as JSON documents. They establish the permissions an item has for AWS resources. With policies, you give or reject access to resources. In addition, you grant or deny access to actions.

Sample SageMaker Endpoint Configuration [For Viewing Only]

The code below is given for your reference only. It sets up configurations for deploying a machine learning model on AWS SageMaker in production and test environments. It uses boto3, the AWS SDK for Python, to create two distinct endpoint configurations with different resource allocations.

The production configuration uses more powerful ml.c5.xlarge instances with two servers and includes serverless capabilities, allowing up to 50 concurrent requests with 2GB memory. The test configuration uses a smaller ml.c5.large example with just one server. After defining these configurations, the code creates a

SageMaker client and registers both configurations with AWS, preparing them for actual deployment (though the final endpoint creation step isn't included in this snippet).

```
import boto3
import sagemaker

# Production endpoint configuration
production_endpoint_config = {
    'EndpointConfigName': 'production-endpoint-config',
    'ProductionVariants': [{
        'VariantName': 'production-variant',
        'ModelName': 'model-name',
        'InstanceType': 'ml.c5.xlarge',
        'InitialInstanceCount': 2,
        'VariantWeight': 1.0,
        'ServerlessConfig': {
            'MaxConcurrency': 50,
            'MemorySizeInMB': 2048
        }
    }]
}

# Test endpoint configuration
test_endpoint_config = {
    'EndpointConfigName': 'test-endpoint-config',
    'ProductionVariants': [{
        'VariantName': 'test-variant',
        'ModelName': 'model-name',
        'InstanceType': 'ml.c5.large',
        'InitialInstanceCount': 1,
        'VariantWeight': 1.0
    }]
}

# Create SageMaker client
sagemaker_client = boto3.client('sagemaker')

# Create endpoint configurations
sagemaker_client.create_endpoint_config(**production_endpoint_config)
sagemaker_client.create_endpoint_config(**test_endpoint_config)
```

Implementation Tips:

- Always use separate endpoint configurations for test and production
- Implement proper tagging for cost-tracking
- Configure appropriate auto-scaling policies
- Monitor endpoint performance metrics
- Implement proper security controls
- Use appropriate instance types based on workload requirements

Sample Scenario: Here's a sample Method of Procedure (MOP) that provisions a test and production environment for the scenario. **Deployment Environment Setup for an E-Commerce Platform:** A major Filipino retail chain like SM is developing their e-commerce platform to compete with existing online marketplaces.

- For the **development environment**, small t3.medium instances work well for developers. The setup needs restricted database size and it must exist in the ap-southeast-1 (Singapore) area. Developers do payment integration testing via sandbox accounts.
- For **testing or staging**, the system needs medium t3.large instances. The environment features moderate database size. It also recreates user flows similar to busy times in the mall. The setup does integration and testing with payment processors such as GCash or PayMaya.
- The **production environment** uses an auto-scaling group with t3.2xlarge instances. It exists for excellent availability across several availability zones. This environment uses complete database capacity with read replicas. Live payment integration with payment gateway occurs as well.

Container Options for Model Deployment

Containerization in AWS Machine Learning puts machine learning models and their necessities into manageable, independent things, which are containers. This method guarantees stable placement and use in varied places. AWS has two chief ways to containerize machine learning.

- **Provided (Built-in) Containers:** AWS gives many ready-made containers. These are good for certain machine learning frameworks plus algorithms. Such containers include pre-installed libraries and needs. They simplify placement and needless special setup. They function well with AWS ML services plus base structure.

- **Customized Containers:** If more specific needs exist, developers create custom containers. These containers suit particular machine learning models and needs. Such a method supplies better choice plus command over the container setup. It permits the addition of specified libraries, frameworks or versions. With tools like Docker, custom containers become built. After this they get deployed on AWS container services.

Key Benefits of Containerization in AWS ML:

Models, plus their dependencies, fit neatly inside containers, so they act the same way whether in a test or after deployment. AWS provides container orchestration with services, such as Amazon ECS plus Amazon EKS. With them, you raise or lower ML workloads according to demand. Containers also conserve resources. They share the operating system's core; they do better than virtual machines. For isolation, containers keep processes separate so that various ML models or dependencies don't interfere with one another. In addition, container images support version control - this feature simplifies recovery and ensures ML experiments are repeatable.

Key Components and Features

- **Ready-made / Provided containers** offer pre-configured images that work well with SageMaker. They include help for regular ML structures, such as TensorFlow and PyTorch. AWS infrastructure gains automated enhancements and this makes deployment easier. Security fixes plus current versions receive administration.

- **Customized containers** grant total control over the environment setup. They do support unique dependencies plus architectures. They permit the use of personalized algorithms. Because they mesh with current ML procedures, specialized needs enjoy versatility.

Optimizing for Performance, Cost, and Latency

Model production deployment requires careful consideration of performance, cost, and latency. This means developers must balance computational needs with quick response times plus budget limitations, all as they uphold model accuracy plus consistent service. For this reason, the main objective includes optimizing these aspects based on unique business demands and relevant uses.

Key Components and Features of SageMaker Hosting

- **Performance Metrics:** focus on model accuracy, throughput, resource utilization, response time next to scaling capabilities.
- **Cost Factors** are about instance type, instance size, storage fees, data transfer costs, autoscaling rules as well as instance types.
- Regarding **Latency Considerations**, these include inference time, network delay, model complexity given the model size, if it is batch or real-time processing, also geographic distribution plays a role.

Best Practices

To choose the right resources, utilize SageMaker Inference Recommender, multi-model endpoints and auto-scaling. For cost reduction, apply tagging, get SageMaker Savings Plans, use Spot Instances along with exploring AWS Cost Explorer. Regarding latency, locate deployments close to users, apply caching, lessen model size along with enable asynchronous inference.

Implementation Considerations

- Deployment Strategy: real-time vs batch, high availability, geographic needs, compliance
- You handle automatic scaling, first starts as well as limits.

Related AWS Services

- Several Amazon Web Services pertain to this.
- The main ones involve Amazon SageMaker, CloudWatch, Auto Scaling along with Cost Explorer.
- Others are S3, EFS, ECR, Lambda along with API Gateway.

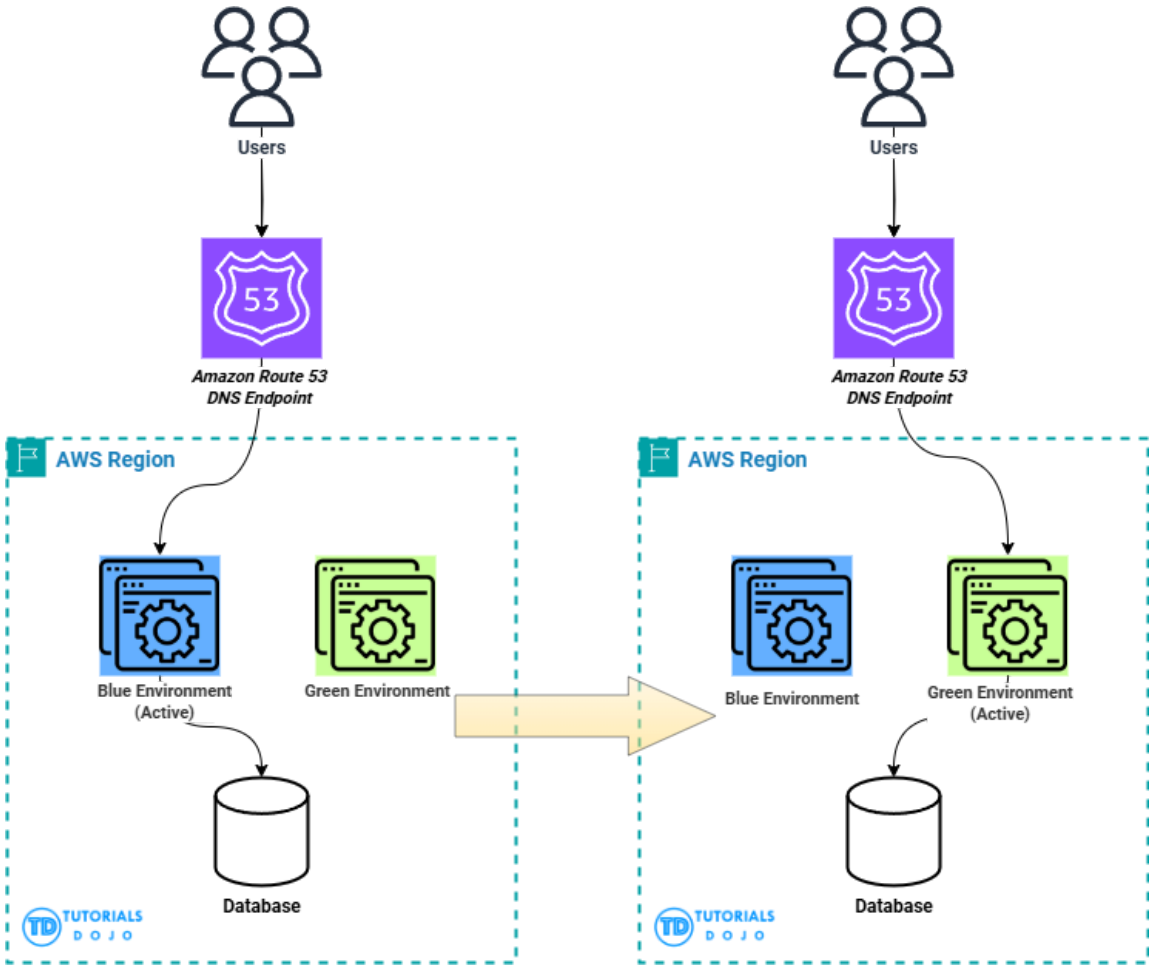
Rollback Strategies

A rollback is the process of bringing a specific version baseline of a system architecture that is stable. This is triggered by a failed deployment due to misconfiguration or dependency mismatch. It is important to minimize downtime to avoid a negative impact to business in terms of model unavailability from the platform. At the moment of model malfunction, which causes misinformation in model prediction, it may risk losing client trust.

How to Manage Failed Deployments and Rollbacks?

Blue/Green Deployment

A deployment strategy where you systematically create two separate parallel environments. A **Blue** environment runs the current version of the application, and a **Green** environment runs the new application version. The idea behind this deployment is to shift traffic between these two parallel environments, offering almost zero downtime and easy rollback capabilities. Once the test cases have been completed, the application will go live directly to the green environment, making the blue environment deprecated. Best combined with **AWS CodeDeploy** and **CodePipeline** for automated transitions.



Canary Deployment

Similar to the Blue/Green Deployment strategy, the **Canary Deployment** uses two models simultaneously. However, canary deployments release the new model incrementally until the traffic moves to the new version of the model.

In other words, it's a deployment option for you to route a small percentage of traffic to the new model version while monitoring how it performs. In relation to AWS, you can specify the traffic distribution from the `*InitialVariantWeight*` objects from the **AWS SageMaker endpoint** argument.

SageMaker Model Registry's "Approval Status"

A model version can be provided by the SageMaker model registry, which you, as a user, can mark as "Approved" or "Rejected". If rejected, with sufficient supporting metrics, you can switch to the previously approved version. Otherwise, if approved, it will initiate the CI/CD deployment section of the model.

According to AWS, in the "Update the Approval Status of a Model" section, you can either approve the model version status manually or via a condition step when you use a SageMaker pipeline. When a template has been used in SageMaker AI, the following set of processes occur during change.

- `PendingManualApproval` to `Approved` - This initiates the CI/CD deployment for the approved model version.
- `PendingManualApproval` to `Rejected` - wherein it is expected for no further action.
- `Rejected` to `Approved` - wherein it initiates the CI/CD pipeline for the approved model.
- `Approved` to `Rejected` - wherein it initiates the CI/CD pipeline for the previously approved model.

The approval status can be changed using the following options:

- Python (Boto3)
- Amazon SageMaker Studio console
- Conditional via SageMaker AI Pipeline

Infrastructure as a Code

By simply leveraging declarative IaC tools such as CloudFormation, Terraform, or AWS CDK, cloud configurations can be kept as a template. Rolling back is much more feasible, similarly as we redeploy the previous template.

Best Practices as a Rollback

- **Automate Monitoring** - use CloudWatch alarms to watch for anomalies or performance dips. This is to ensure that costs are inspected well. Call for a rollback if thresholds are exceeded.
- **Maintain a Stable Baseline** - Always maintain a stable baseline available if rollbacks are expected.
- **Orchestrate via CI/CD** - Tools like AWS CodePipeline or Jenkins can help in managing your deploy-and-rollback process automatically.

Related to AWS Services:

Let's now connect the AWS services mentioned earlier to this topic. If you're not yet familiar with it, kindly view our AWS list mentioned in the first part of this domain.

1. **Amazon SageMaker Model Registry** - Manages new model versions, records changes, and approves models.
2. **AWS CodePipeline / CodeDeploy** - Runs build test and release tasks. It supports canary release or blue/green methods.
3. **Amazon CloudWatch** - Checks performance sets alarms on unexpected results such as delay or error. It links rollback tasks when limits are passed.
4. **AWS CloudFormation / AWS CDK** - Lists AWS resources in a script. This lets you roll back using an earlier template copy.

Key Takeaways:

- Versioning makes sure every model state can be recreated or found.
- Rollback methods (blue/green, canary, or model registry checks) lower downtime while keeping user trust solid.
- AWS Services such as SageMaker Model Registry, CodeDeploy, CodePipeline, plus CloudWatch build the base for a dependable, automatic deployment with fallback options.
- Regular checks help you spot if a new version fails so you can switch back fast to a working version.

Chapter 3.2 AWS Deployment Services for ML Models

Overview of Amazon SageMaker and other AWS services for deployment.

Core Concept

Amazon SageMaker is AWS's complete service for machine learning. It helps you label data, choose ready-made algorithms, perform training, then deploy models. With SageMaker Model Hosting, you can deploy trained models as managed endpoints with ease. SageMaker sets up the needed hardware, keeps it working well, plus adjusts capacity whenthe load shifts.

Reasons to Use SageMaker for Deployment?

- **Innate Scaling** - SageMaker's Endpoints change capacity automatically based on request count; this makes costs match actual use.
- **Reduced Ops Overhead** - SageMaker manages servers, adjusts network settings plus handles updates.
- **Tracking & Records** - It works with Amazon CloudWatch to show measures like delay and error numbers.
- **Manage Versions & Rollbacks** - You have control over model versions via SageMaker Model Registry. If problems occur, you can return to an earlier version quickly.

Deployment Options within SageMaker

- **Real-Time Endpoints** - To make fast, low-latent, steady predictions.
- **Batch Transform** - Use when you check many data items, for example every hour.
- **Asynchronous Inference -** It is best when requests need extra time so that work can continue with other requests.
- **Serverless Inference** - Adjusts the capacity to nothing when not needed, then adds capacity when work grows.

Other AWS Services for ML Model Deployment

AWS Lambda Function - Serverless compute service that responds to events by running code. Use this when there is a need for small, immediate tasks or custom checks before or after predictions. Here, you only pay for each call; no need for server upkeep. However, the only caveat is it is not fit for big models or fast, high-volume predictions.

Amazon Elastic Container Service (ECS) / Elastic Kubernetes Service (EKS) - it is a Platform-as-a-Service that manages containers (ECS handles AWS-based setup; EKS manages Kubernetes). You can use this when you

require custom container setups or a microservices layout that needs tighter control over networks, scaling, or container setups. The advantage of this is that you can create custom ML environments; it joins smoothly with AWS identity and access. However, it requires more daily work compared to SageMaker.

AWS Elastic Beanstalk - It is a Platform-as-a-Service that helps you launch web apps quickly by handling the setup work. Use this service if your web app holds an ML model and you want to deploy it without managing technology details. Its advantage is that the setup is easy, and it prepares EC2, load balancers, etc. However, it does not focus on ML tasks like SageMaker.

AWS App Runner - A service that runs containerized apps or web services directly. Use it if your model is in a container and you need a simple method for web-based predictions. For its advantage, It scales on its own; setup is simple. However, it misses several ML features found in SageMaker, such as built-in monitoring of model parameters.

Understanding the role of SageMaker endpoints for model hosting.

This is where a SageMaker endpoint puts your ML model online as a REST service that you may call for real-time answers. For its endpoint configuration, it lists facts about the model file, instance kind, and initial scale plans, plus details about encryption when necessary.

Key Features

- **Autoscaling** - The system changes the number of servers based on request count or delay goals.
- **Multi-Model Endpoints** - It places several models at one address, cutting cost if you have many smaller ones.
- **Traffic Shifting** - It sends part of the traffic to a new model during an initial run.
- **Secure Access** - It works with VPC, uses SSL encryption plus IAM-based entry.

SageMaker Endpoints for Managed ML Hosting

Amazon SageMaker Endpoints furnish a managed space where machine learning models reside. They make deployment and scaling simple with reduced operational burden. Through a REST API, these endpoints give current predictions and present a dependable method for serving models widely. SageMaker manages infrastructure setup, capacity handling along with auto scaling because of actual traffic. With this you only pay for resource usage. Multi-model endpoints drive cost savings since many models use one endpoint. This lowers infrastructure costs, especially with smaller or less used models.

Advantages:

- SageMaker manages servers completely, so you don't need to. It handles scaling also infrastructure, thereby saving work.
- The system changes the number of instances according to incoming traffic or delay targets. It adapts to variable workload.
- For smooth transitions during model updates, traffic moves slowly toward new models. As a result updates cause minimal disturbance.
- With encryption and integration with IAM, access has been controlled. This bolsters security for production.

AWS Lambda for Serverless Inference

AWS Lambda works as a serverless platform and suits simple inference jobs done upon request. Lambda lets a user run code when particular events happen, thus infrastructure management becomes unnecessary. Payments cover only the computing time actually used. Because of this Lambda offers a good value for quick, streamlined predictions. It suits little models best or situations where actions trigger forecasts, like API requests, alterations to a database or similar prompts from AWS.

However, Lambda does not work well for huge models or instances needing major computing power, such as deep learning models. Lambda is less effective for numerous predictions because Lambda functions function better with tasks driven by events that happen less frequently.

Advantages:

- It is economical because payment is only for the time used when the function runs.
- Lambda hides the base infrastructure. This simplifies deploying small ML models with speed.
- It scales on its own to manage several parallel requests. No one needs to do it by hand.

Lambda works best if fast, small predictions matter. In addition, it supports a serverless, event-driven build.

Comparative Analysis: When to Choose Each Option

Deployment Option	Scalability	Cost	Control	Complexity
SageMaker Endpoints	Auto-scaling based on traffic	**Pay-as-you-go** for resources	Limited control over infrastructure	Low (fully managed)
AWS Lambda	Scalable for event-driven tasks	Low cost for small workloads	Minimal control over execution environment	Low (serverless)
ECS/EKS (Kubernetes)	Highly scalable with custom settings	Can be higher	Full control over containers and infrastructure	High (requires container management knowledge)

Employ SageMaker endpoints when a fully managed service for real-time inference becomes a priority. SageMaker fits well for scalable deployments plus it assures reduced operational work. It works well in situations where server management lacks appeal. Quick, dependable model hosting with simple scaling takes precedence.

Lambda presents a useful tool for event-driven setups where the goal involves running basic inference tasks. Server management also becomes less of a problem. In essence Lambda works in cases which do not call for many resources or large amounts of predictions.

For advanced requirements, pick ECS or EKS. Such requirements include those ML models that SageMaker or Lambda cannot easily handle. This option becomes preferable for custom ML settings besides hybrid structures. It functions when a person seeks significant control over containers as well as microservices. It involves greater knowledge besides greater oversight.

Summary:

- **SageMaker Endpoints** suit managed, scalable ML hosting. This setup works great when low-latency predictions plus automatic resource management become important.
- **AWS Lambda** stands as the better selection for serverless, event-driven tasks. This option gives a cheaper answer for simple inference but it cannot manage big or popular models very well.

- **ECS/EKS** present bigger flexibility plus control. You must do extra management. Because of this they exist as a good selection for teams with serious container orchestration needs or certain resource demands.

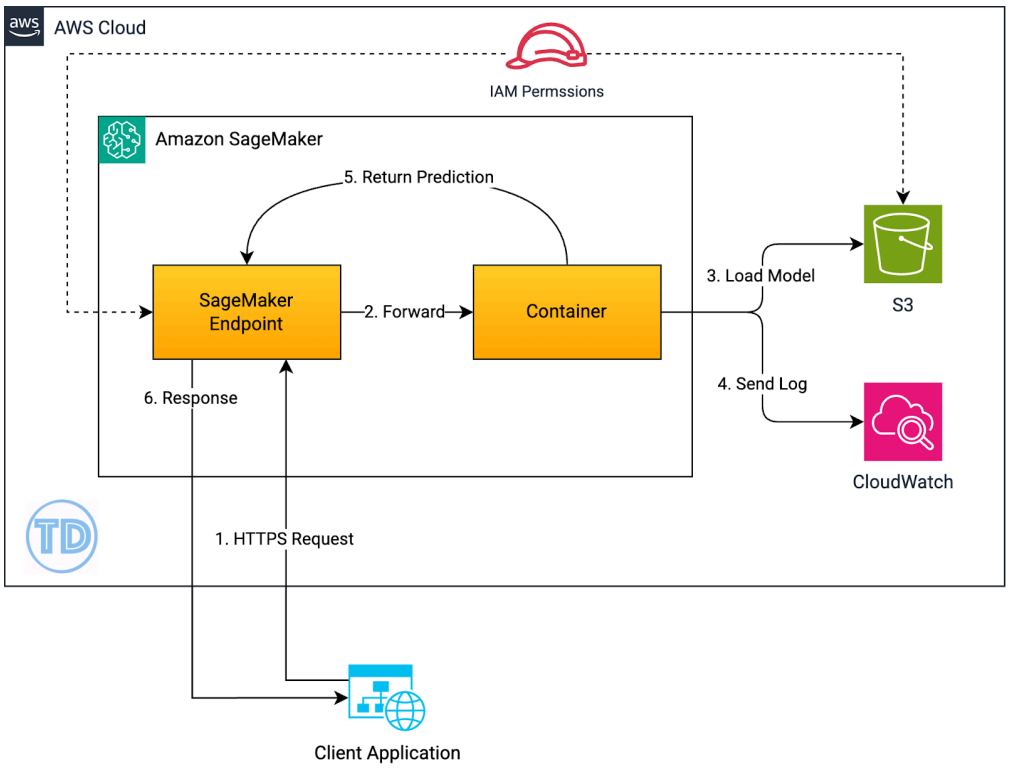
Each option has positive aspects according to your scalability demands, monetary limits next to the amount of control you want over your infrastructure.

Chapter 3.3 Serving ML Models in Real-Time and Batch Modes

Real-time serving: Using SageMaker real-time endpoints for low-latency predictions.

What is Real-Time Inference?

Real-time inference refers to a method that delivers predictions at the exact moment an inquiry has been requested. This type of modality is critical for systems that require little to no delay in predictions. This may include fraud checks for finance and emails. Furthermore, it guarantees a smooth user experience as well especially during urgent times. It boosts the responsiveness of interactive apps like chat bots or personalized ads.



Based on the implementation above, this is a sample AWS workflow that implements real-time inference on multiple models from the backend and will return the request to the client application. It simply utilizes a single endpoint which makes it reduced

How To Implement Real-Time Endpoints in AWS SageMaker

1. **Prepare Your Model** – Develop a new model which can come from you SageMaker or utilize pre-trained model artifacts via API call from PyTorch or Tensorflow ML frameworks. Save the model in an Amazon S3.
2. **Build a Model Object** – Perform a packaging action for the model and its inference prompts inside a Docker container, then link the container with the model artifacts stored in Amazon s3.
3. **Set Up an Endpoint** - Since it can endpoints can Choose the compute instance types *(e.g., ml.m5.large, ml.c5.xlarge)* that best fit your load as well as model difficulty. For an optional choice, you can turn on auto scaling to change the number of instances when traffic shifts.
4. **Set Up Deployment** - Once the endpoint is built, you get an HTTPS URL for sending inference requests. Send a POST request that contains the input data to your endpoint. Then, sageMaker will return the prediction in real time.

Best Practices for Real-Time Inference

- **Autoscaling** – Configure Amazon SageMaker Automatic Scaling to handle sudden traffic increase efficiently.
- **Endpoint Monitoring** – Monitor the endpoint using Amazon CloudWatch to track latencies errors plus instance performance.
- **Multi-Model Endpoints** – Keep several models behind a single endpoint to reduce operational work if you have many models. This reduces the operational overhead.
- **Deployment Tactics** – You can implement canary or blue/green deployments to test new model versions with a fraction of the traffic.
- **Security** – You can secure the endpoint using AWS IAM, VPC endpoints, and optionally, SageMaker PrivateLink to manage incoming traffic.

Batch inference - Using batch transform for large-scale, non-time-sensitive predictions.

Batch inference (or Batch Transform) is optimized for processing large data sets in groups rather than processing requests in real time. It is suitable when there is a need for non-time critical predictions, such as generating periodic risk reports, performing seasonal trend reviews, or processing terabytes of data in scheduled jobs.

Why use batch transform:

- **High-Volume Processing** - It works with large data sets such as terabytes of text, images, or CSV files.
- **No Persistent Endpoint Needed** - You pay only until the group task ends; there is no need for a constantly active endpoint.
- **Easy Integration** - It fits offline tasks like checking risks, boosting the value of data, or evaluating complete data sets.

How to implement batch transform in AWS SageMaker

1. **Prepare by keeping your raw data or CSV file in Amazon S3.**
2. **Create a Batch Transform Job**
    - Point to where the model artifacts reside (trained in SageMaker or made somewhere else).
    - Give the path to the input data in S3 along with the needed output path.
3. **Select Compute Resources** - choose suitable instance types for memory and CPU/GPU requirements.
4. **Run the Job** - SageMaker starts a compute cluster, works on all data in groups, and saves predictions in the given S3 folder.
5. **Tear Down (Automatic)** - After the job completes, SageMaker stops the compute resources by itself.

Best Practices for Batch Transform

- **Efficient Data Splitting** - If your dataset is very large, break it into smaller parts (for example by date or ID) to quicken the process.
- **Parallel Processing** - Start several instances so they work at the same time; this cuts process duration.
- **Cleanup** - Erase or store temporary data in S3 that you do not need once the job ends.
- **Cost Management** - Use the same model files for several batch jobs if the model stays identical; you do not need to create new endpoints each time.

Deciding Between Real-Time and Batch Inference

Use Real-time Inference when:

- The program or user expects a reply soon (less than 100 ms).

- Delay matters in cases like fraud checks, simple conversation bots, tailored offers.
- Load stays steady or shifts at will while you accept auto-scaling.

Otherwise, use Batch Inference when:

- Predictions appear overnight or at a set time.
- A large dataset calls for a cost-wise solution without a steady endpoint.
- Immediate outcomes are unnecessary for tasks like monthly risk reports or seasonal trend reviews.

Related AWS Services and Features

- **Amazon SageMaker**
  - **Real-Time Endpoints** - Endpoints show low delay, work without pause.
  - **Batch Transform** - Run tasks on set schedules or as needed to cut costs.
- **Amazon CloudWatch -** Watch measures like CPU, memory, error counts, delay at each endpoint.
- **AWS Lambda** - Answer real events to perform tasks before or after contacting SageMaker endpoints.
- **AWS Step Functions** - Handle several batch tasks or layered data jobs.
- **SageMaker Model Monitor** - Spots data shifts or unexpected issues in timely or batch setups.

Chapter 3.4 Provisioning Compute Resources

In this chapter, we'll be discussing more about the CPU and GPU instances that we can provision for training inference. This is useful for having additional knowledge in selecting the correct instances for efficient yet cost-effective training time. This covers a decision guide based on characteristics in selecting a particular instance.

Choosing between CPU and GPU for training and inference.

CPU instances only rely on the virtual computer's processor during training and model inference. On the other hand, GPU instances rely on the virtual computer's graphic processor for the same tasks. The information below describes when we should only be using this type of instance based on AWS documentation as well.

CPU Instances:

- **General Purpose** - Works for basic machine learning tasks, cleans data, or handles small jobs.
- **Lower Cost** - Costs less than GPUs, so it fits development, light tests, or early experiments.
- **Wide Availability** - Offers several groups (such as T3, M5, C5 on AWS) to suit various compute needs and memory needs.

When to Use CPUs

- **Light Models** - Models like simple regression, decision trees, or small neural networks run fast on CPUs.
- **Prototyping and Development -** Suits short tests, data work, or creating first versions of products.
- **Low-Throughput Inference** - When inference requests come rarely or batches stay small, CPU endpoints save money.

GPU Instances:

- **Parallel Computing Design** - GPUs work with matrices plus vectors. These tasks form the base of deep learning.
- **High Throughput** - They speed up the process of teaching large neural network models, such as convolution models or transformer models.
- **Specialized Instance Groups** - AWS offers the P-series (NVIDIA GPUs) plus the G-series, which suit deep learning lessons with large-scale use.

Decision Factors: CPU vs. GPU

- **Deep Learning** - Build large neural networks for computer vision, natural language processing, or recommendation systems.
- **High-Volume, Low-Latency Inference** - Produce live predictions at scale with large or complex models such as BERT or GPT language models.
- **Time-Sensitive Projects** - When compute demand is high or lesson periods are short, using GPUs is worth the extra cost.

Managing compute resources for both test and production environments.

Distinct Setups

- For **development and testing,** it is wise to use smaller, CPU-based instances like t3.medium or m5.large. To keep expenses minimal, consider spot instances.
- The **staging setup** must mirror the production setup but with a smaller size or fewer instances. This is done to assess performance.
- For the **production setup,** scale up or out by using robust CPU or GPU clusters. This action is based on performance and cost needs.

Elastic Scalability

- **Automatic Scaling for Real-Time Endpoints**

    - Set up Amazon SageMaker Automatic Scaling for the endpoints.
    - Change the count of instances based on CPU or GPU usage, the number of invocations, or latency.

- **Workloads in Batches**
    - To handle large, offline inferences or extensive data training, start clusters of GPU instances just during the job runtime.
    - Employ Amazon EC2 Spot Instances to save on expenses for tasks that are not critical or can be retried. Development training jobs or internal analytics are examples.

Monitoring and Optimization

Amazon CloudWatch Metrics

- Observe CPU/GPU use, memory, delay as well as I/O functions.
- Establish alerts. These will discover times when usage is more than or less than a limit. The alerts trigger scaling or setup changes.

Experimentation

- Employ Amazon SageMaker Experiments or record data manually to follow training setups. Track setups such as instance type and hyperparameters against performance measures.
- Find the best balance of expense versus output.

Chapter 3.5 Model and Endpoint Requirements

In this chapter, we'll be discussing the requirements involving endpoints. These requirements are important non-functionality factors based on the client's needs. In Amazon SageMaker, while a model describes the architecture and capabilities of the model to predict, an endpoint, on the other hand, functions as a hosted environment (Mobile App, Web app, IoT Device, etc.). A model must be deployed to an endpoint before applications can transmit data to it. The following information below describes how they are being selected based on their endpoint types.

SageMaker Endpoint Types

Real-Time – This type of endpoint prioritizes fast response times and should also manage many requests. These points function well when programs require immediate predictions. Users can select a particular instance kind and also can configure automated scaling, and the amount of traffic manages the scaling.

- **Amazon SageMaker Real-Time Endpoints**, which are called *persistent* or *provisioned endpoints*, exist for apps. These apps require low latency plus fast inference. The endpoints stay fully active. They ensure quick responses to requests that come in.

When to Use Realtime Endpoints:

- **High-Volume Applications** – Realtime endpoints suit situations that require a constant need for inferences. For instance, consider e-commerce systems suggesting items or systems to detect fraudulent activity. It is critical to keep response times of just milliseconds, especially for service level agreements (SLAs)

Key Benefits

- **Steady Performance** – These provisioned endpoints offer an anticipated latency, which is key for uses that have strict demands when it comes to speed.
- **Auto Scaling Capabilities** - SageMaker's endpoint auto-scaling feature lets the amount of instances go up or down through horizontal scale. It does this based on current traffic, wherein it optimizes how resources get used, and it keeps the speed good even when the load changes.
- **Automatic Adjustment** – The automatic adjustment function in SageMaker lets the number of instances go up or down. It does this based on current traffic. It optimizes how resources get used, and it keeps the speed good even when the load changes.

Limitations

- **Ongoing Expenses** – Given that the systems function at all times, expenses also accumulate at all times. It is true even when there is less than normal demand. This situation could create higher expenses during low traffic.

- **Scaling Management** – Auto-scaling changes the count of items dependent on demand. Configuration and command of adjustment strategies are still needed. This confirms that the system strength is on par with flow needs. It stops likely slowdowns or using too much support.

Serverless – This is good for intermittent or hard-to-guess amounts of traffic. Serverless options automatically adjust resources. This adjustment is rooted in current needs, where it takes away any need for human assistance. One good example of service here is Amazon SageMaker Serverless Inference.

- **Amazon SageMaker Serverless** from Amazon offers a fully managed environment to deploy machine learning models easily. The hassle of manual provisioning or through infrastructure as a code became less as it does not require managing the infrastructure. AWS handles supplying and adjusting computing power. This happens based on the request amount. It gives a cost-saving method to handle changes in traffic.

When to Use Serverless Endpoints:

- **Hard-to-Guess Traffic Patterns** - Serverless endpoints are preferable when inference needs are immeasurable due to randomness.
- **Small to Medium Models** - For models that don't need to be active all the time, serverless options get rid of resource consumption when in an idle state. This introduces cost-effectiveness, making it economical.

Key Benefits

- **No Instance Management** - AWS takes care of supplying and adjusting calculation sources. This removes the difficulty of dealing with instances.
- **Payment for Use**: Expenses happen with the time spent processing and the data amount handled. There are no fees when idle.

Limitations

- **Cold Start Latency** - First requests could have greater delays as AWS supplies sources. This could matter for applications sensitive to delay.

- **Restricted Sizes and Memory** - Serverless endpoints work with certain memory sizes (1 GB to 6 GB in 1 GB steps). They might not fit all model needs.

Asynchronous Inference

Imagine you're finding an option to anticipate inference, which will only be processed at a later time, especially for longer processing requirements. You look no further as you realize that there is a SageMaker Asynchronous Inference. This option is beneficial when there isn't a need for immediate response and there are a lot of processes going through.

When to Use Asynchronous Inference:

- **Prolonged Inference Time —** If the model serves for a long-term task (ex. Batch Processing, High Quality Image Creation, Video Development, or Full Writing Content), you may want to use this option as it focuses on quality rather than speed.

Key Benefits:

- **Avoidance of Blocking Operations —** in case a lengthy inference has been done through asynchronous operation, it will be settled aside in its backend, relieving these connections.
- **Receive Notifications and Queuing —** through the model's asynchronous capability to process data, a feature of being notified and having callback functionalities are possible.

Limitations:

- **Expect for a higher latency** – the queuing algorithm may introduce more delays than usual. Therefore, expect a higher latency.
- **Additional Storage Requirements** - you must configure higher storage solutions such as Amazon S3 in storing input data and pass the expected output.

Importance of Endpoint Requirements:

- **Scalability** – Endpoints should manage changes in inference demand. This needs settings that permit scale increases when demand is high. Reductions in scale at times of low demand are needed.
- **Performance** – Picking suitable instance types and setting auto-scaling rules make sure endpoints achieve latency and throughput goals. Timely predictions are supplied.
- **Cost Efficiency** – Correct settings for the devices aid in the optimization of expenses. This is done by matching resource assignment with real application habits, which keeps excessive supply from happening.
- **Dependability** – The devices must feature high availability and the tolerance of mistakes. That will make certain that inference services are consistent and reliable.

Chapter 3.6 Choosing Appropriate Containers

Choosing an appropriate container setup becomes key when machine learning models are put into deployment. This choice helps confirm reliability and good functioning next to cost efficiency. Containers give a steady setup. They make the handling, expansion, and coordination of machine learning actions.

Selecting Pre-built or Customized Containers

Pre-built Containers – SageMaker offers optimized pre-built containers that are optimized for ML frameworks such as TensorFlow, PyTorch, Scikit-learn, and XGBoost. They come with libraries and dependencies already configured in the configuration file of the container. Hence, it enables users to deploy faster deployment and reduces initial setup difficulties.

When to use Pre-built Containers?

- For leveraging standardized model architectures
- Instant testing and prototyping (good for hackathons)
- Project with significantly tight deadlines.

Customized Containers – this option gives you the control to manipulate the settings of your runtime environment depending on the project requirements. This enables the implementation of custom libraries, scripts, and tools to these containers.

When to use Customized Containers?

- Specialized ML frameworks, perhaps using private pre-trained models.
- Specific dependency requirements for ensuring system integrity which are not present in the pre-built containers.
- Enhanced security for better compliance.

For instance, a financial institution in the Philippines that uses a fraud detection model created via its own analytics software may gain from custom containers. This method permits exact control of dependencies and security settings. It can help to adhere to local rules, those set by the Bangko Sentral ng Pilipinas (BSP).

Containerization Options for Edge Devices with SageMaker Neo

For the complete discussion of SageMaker Neo, refer to the list of AWS resources mentioned at the start of Domain 3. However, for the overview, it is used to enable ML models to be optimized accordingly for edge deployment to a particular device for better performance.

Benefits of SageMaker Neo:

- Inference latency is reduced.
- Memory footprint decreases.
- Enhanced performance and efficiency.

Typical Use Cases for Edge Deployment:

- Remote healthcare assessments in rural areas.
- Traffic monitors occur in real time and improve optimization in Manila.
- Agricultural yield is observed in Mindanao.

How to Use SageMaker Neo for Edge Deployment

1. **Train a Model** - Use standard SageMaker training instances to train the model.
2. **Compile the Model** - Use SageMaker Neo to optimize the trained model for the hardware. Focus on a target such as ARM-based units or Raspberry Pi.
3. **Deploy to Edge Units** - Use SageMaker Edge Manager to deploy the compiled model. It is a secure and efficient model control on edge units.

Chapter 3.7 Optimizing Models for Edge Devices

When deploying machine learning models, it often requires supplementing the edge inference requirements. Edge computing involves processing data near to its source of generation. In the Philippines, this need is significant because of inconsistent web access in distant rural regions, so local resources are essential.

Understanding Edge Deployment

Machine learning models deployed on edge devices often face these challenges:

- **Restricted resources** – Low computational ability, memory capacity along with data storage for end devices.
- **Network limitations** – limited internet connectivity.
- **Latency Requirements** – oftentimes, real-time is needed when processing such predictions.
- **Energy limitations** – model operations for mobile devices require high energy consumption.

Common examples of Filipino-context edge computing scenarios:

- **Smart Farming in Nueva Ecija** – Agricultural sensors analyze soil health data on the device itself.
- **Real-time Traffic Monitoring in Metro Manila** – Device on the edge processes real-time video streams for better traffic control.
- **Bicol Region Disaster Alert Systems** – Devices capture data from sensors and provide alerts if thresholds are exceeded. This occurs without cloud connectivity.

This is the reason why optimization for edge Models are important.

Optimizing ML models for edge deployment ensures that in your inference:

- It ensures that the model is capable of performing well in limited hardware.
- It achieves low latency when deployed in the cloud.
- It saves computing resources.
- It operates effectively even with unstable internet connections due to proper distribution of traffic workloads.

Amazon SageMaker Neo: Edge Optimization Simplified

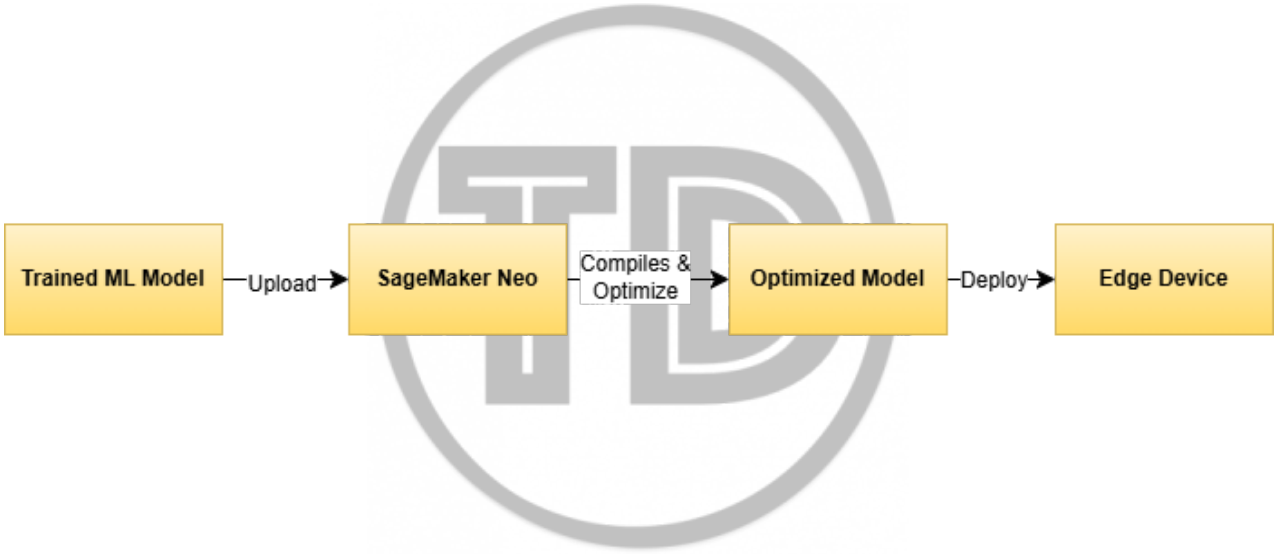
While SageMaker Neo has been discussed earlier in this domain, we'll do a deep dive on how SageMaker Neo works with regards to edge optimization. **Amazon SageMaker Neo** addresses the challenges mentioned earlier by having the ability to optimize ML models for deployment on the edge. It can compile models trained in various ML frameworks such as TensorFlow, PyTorch, and MXNet) into lightweight executables that run efficiently on broader hardware platforms.

Supported Frameworks and Hardware Platforms:

Frameworks Supported	Hardware Platforms
TensorFlow	ARM (Cortex-A, Cortex-M)
PyTorch	NVIDIA Jetson
MXNet	Intel-based devices
XGBoost, Scikit-learn	Raspberry Pi
ONNX	AWS Inferentia

How SageMaker Neo Works:

Let's take a look at this flowchart to demonstrate how SageMaker Neo can optimize your processes clearly.



Step-by-Step Hands-On Using SageMaker Neo

For this part, we'll be discussing the step by step hands-on guide in deploying models using SageMaker Neo. This is just the high-level overview for you to understand how it works generally. For this part, we assume that we already built an ML model using any ML frameworks that SageMaker Neo supports.

1. **Train your Model**
    a. Train using Amazon SageMaker built-in algorithms or custom frameworks.
2. **Compile Your Model using SageMaker Neo**
    a. Choose your model that is already trained from S3.
    b. Choose your target device platform.
    c. Use Sage Maker Neo to optimize your deployment packages dedicated to the ML framework you use and the target endpoint device. It will compile the model afterwards.
3. **Deploy on Edge Device**
    a. Download the compiled model artifacts, then you can deploy the model to your edge device via SageMaker Edge Manager or manually.

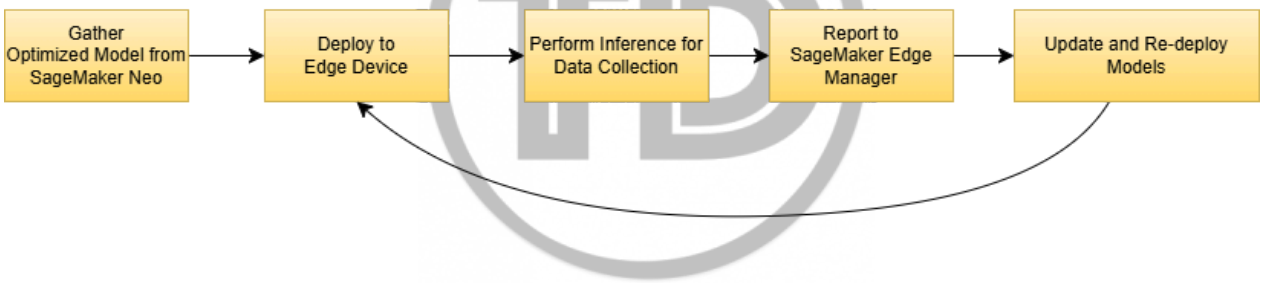
Sample cli command for optimization:

```
aws sagemaker create-compilation-job \
  --compilation-job-name optimize-td-edge \
  --role-arn arn:aws:iam::<your-account-id>:role/<sagemaker-role> \
  --input-config
'{"S3Uri":"s3://td-optimized-model/input/model.tar.gz","DataInputConfig":"<input_config
>","Framework":"TENSORFLOW"}' \
  --output-config
'{"S3OutputLocation":"s3://td-optimized-model/output/","TargetDevice":"rasp3b"}' \
  --stopping-condition '{"MaxRuntimeInSeconds":900}'
```

Monitoring and Managing Edge Models with SageMaker Edge Manager

Once set up, Amazon SageMaker Edge Manager lets you control your edge work reliably:

- Put edge models in place, keep them safe plus care for them.
- Check model shifts with their speed.
- Refresh models on collections of edge devices.



Best Practices for Optimizing Models on Edge Devices

- **Use small frameworks and plain models**: Pick neural designs that are simple or choose small models like MobileNet or EfficientNet-lite.
- **Quantize models:** Change 32-bit numbers to smaller ones such as 8-bit digits to shrink size, boost speed.
- **Model pruning:** Remove unnecessary neurons/weights to lower model complexity.
- **Monitor device health and performance:** Continuously monitor resource use, model correctness, delay.

Chapter 3.8 Skills for Deployment Infrastructure

Model efficiency is key as machine learning becomes complex over time. Correct settings for a model influences its performance, cost, in addition to what the user encounters. For exam candidates pursuing MLA-C01 certification, trade-off knowledge in deployment is vital, as is proper infrastructure choice.

Assessment of Performance, Cost as well as Latency Trade-offs for Placement

To place machine learning models effectively, one must assess the effects of performance, cost as well as latency on system efficacy. Such considerations aid in selecting the correct infrastructure for model placement.

Three Key Deployment Factors:

1. **Performance** - This assesses the model's efficiency in managing requests and creating precise forecasts based on current resources. Performance improves with model optimization, the use of proper hardware and inference strategy adjustments.

   Example: A Manila-based fintech company uses a real-time fraud detection model. High performance is crucial to detect fraudulent transactions without delay, ensuring the company's reputation and minimizing financial loss.

2. **Cost** - Deployment expenses involve the infrastructure, maintenance, storage along with computation capabilities needed to operate models extensively. It is vital to maintain expenses inside the organization's fiscal plan while satisfying the performance and time-delay needs.

   Example: a domestic e-commerce business, such as Lazada Philippines, uses suggestion systems. These systems should have good cost-effectiveness, as they must take into account the large number of product searches and the restricted computation resources available to smaller business structures.

3. **Latency** - Latency defines how long a model needs to furnish inference results after a request reaches it. A shorter duration is essential to applications functioning in real time, where fast resolutions matter. Autonomous vehicles, finance industry trade or custom advice systems depend on low latency.

   Example: For instance "Grab Philippines uses real-time traffic prediction models for their ride-hailing service." Considerable latency impairs user experience. Minimization of route optimization and fare calculation lags becomes important.

Balancing the Trade-offs:

Factor	Description	Considerations
Performance	It determines the operational speed and precision of the model.	Complex tasks will need more powerful hardware, refined models along with adaptation.
Cost	It affects long-term efficiency and viability.	Right instance types, such as CPU or GPU, must be carefully picked. One should use options that lower expenses, for example, reserved or spot instances.
Latency	For user experience, especially with real-time use, it is essential.	Ask yourself if it needs to be in real-time, and how do infrastructure choices impact response durations?

Example: A logistics company, located in Manila, uses machine learning models. It wants to make its delivery routes better in real time. The company must assess some things for this.

- **Performance** – The model has to work with current traffic data and change routes fast. Faster computers (GPU or custom hardware) could be important for quick processing of numerous data pieces.
- **Cost** – The logistics business must find a balance between computing expenses and what it can afford. GPUs are great for performance. But a simpler model may work better for both price and performance using CPU systems.

- **Latency** – Since the company depends on optimization of routes in real time, lowering the delay is vital. Without significant wait times, the model should give route changes. This ensures things go smoothly for the user.

Choosing the correct compute environment:

Choosing the right compute environment (CPU, GPU, or specialized hardware like AWS Inferentia) is key in balancing performance, cost, and latency.

CPU-based Instances

- **Optimal use:** Conventional machine learning patterns, basic patterns or patterns with modest needs for intensive computation.
- **Expenses:** It has the lowest price relative to GPUs and specialized equipment.
- **Capability:** It is adequate for patterns that have less demand for computation.
- **Delay:** The processing power causes a medium duration.

For example:"A Manila-based BPO uses a logistic regression model for customer churn prediction, which doesn't require heavy computation. A CPU instance gives value and also has good execution for this specific type of work.

GPU-based Instances

- **Best for** – Deep learning models such as those used for computer vision, NLP, or reinforcement learning.
- **Cost** – More expensive than CPU-based instances.
- **Performance** – High, especially for complex, high-volume models that require parallel processing (like CNNs for image processing).
- **Latency** – Low, providing quick inference for real-time applications.

Example: Grab Philippines deploys deep learning models for real-time image recognition in their app to enhance safety by detecting obstacles on the road. These models require GPU instances to achieve low latency and high performance.

AWS Inferentia

- **Best for** – Deep learning models at a large scale benefit most. It suits high-throughput inference tasks.
- **Cost** – Inferentia offers better value compared to GPUs when performing inference.

- **Performance** – Its performance is especially strong with large models similar to transformers (utilized with NLP).
- **Latency** – Inferentia offers minimal latency, which is important for inference performed in real time.

Example: In Metro Manila, a healthcare startup uses AWS Inferentia for their deep learning-based diagnostic models. These models give fast results for medical image analysis and the costs are less than if they selected GPUs.

Decision Matrix for Compute Environment Selection

Compute Type	Use Case	Cost	Performance	Latency
CPU	Traditional ML models for lightweight activities.	Low	Moderate	Moderate
GPU	Deep learning models for real-time inference.	High	High	Low
AWS Inferentia	Large-scale models (NLP, vision)	Moderate	High	Very Low

Selecting the Right Deployment Orchestrator

Selecting a suitable orchestrator is vital to automate, manage next to scale model deployments. Two common choices exist: Amazon SageMaker Pipelines plus Apache Airflow (through Amazon MWAA).

Amazon SageMaker Pipelines

- **A great fit for:** Comprehensive handling of machine learning procedures (data operations, model instruction as well as introduction) on AWS.
- **Strengths:** It is fully combined with other AWS services. It is simple for AWS-focused procedures. It uses automated model introduction and monitoring.

- **Example:** "Lazada Philippines uses SageMaker Pipelines to manage the continuous training besides deployment of its recommendation system, ensuring that the system stays up-to-date with new product trends."

Apache Airflow (via MWAA)

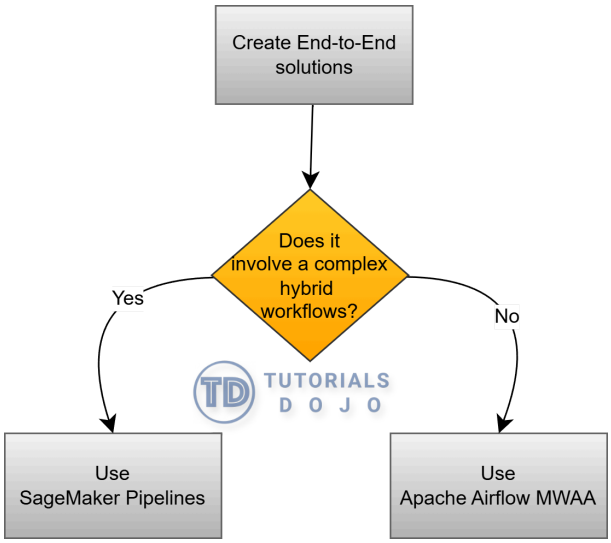
- **A great fit for:** Intricate, mixed procedures through AWS and on-premise solutions.
- **Strengths:** It is very customizable. It is well-suited for multi-stage procedures. It joins with several data origins and systems outside AWS.

Let's have a comparative analysis to better differentiate the correct deployment ML workflows:

Feature	SageMaker Pipelines	Apache Airflow (via MWAA)
Best Use Case	AWS-native ML workflows (end-to-end modeling)	Cross-platform workflows (AWS + external systems)
AWS Integration	Native integration with SageMaker, S3, Step Functions	Requires plugins or custom code
Learning Curve	Lower (Python SDK: sagemaker.workflow)	Higher (uses DAGs in Python, YAML config)
Extensibility	Less extensible outside AWS	Very extensible across services
Infrastructure Management	Fully managed by SageMaker	Managed via MWAA, but still more config needed
Monitoring & Debugging	Integrated via SageMaker Studio	Airflow UI, CloudWatch integration

Example: BDO employs Apache Airflow for automation of the fraud detection process. The process contains several steps. Data is extracted from diverse, local along with cloud-based databases.

Here's a flowchart you can follow to decide whether to use SageMaker Pipelines or Apache Airflow.



Multi-model or Multi-container Deployment Strategies

Multi-Model Endpoint (MME) – lets you put multiple machine learning models on a single endpoint. Because it uses shared resources, this lowers operational costs. It works well if different models are not used often or don't demand much computing power. It gives you a cheap way to deploy models at scale.

- **Advantages:** it is cost-effective, as shared resources diminish all infrastructure expenses. If models are not used often or request patterns change, this is beneficial. It is easier, because fewer endpoints mean less operational complexity.
- **For example:** A Manila-based restaurant chain uses multi-model endpoints to serve several menu recommendation models. Each model loads solely when necessary, reducing fees because of infrastructure sharing.
- **Related AWS Service:** Amazon SageMaker Multi-Model Endpoints supports loading when requested and model sharing to save money. Multi-model endpoints assist with models that have little traffic or unpredictable usage.

Considerations:

- **Performance Limitation**: Models use similar resources - hence, high usage by one model can affect others.

- **Cold Start Latency:** The first time a model loads, it may create small delays, particularly with models people do not access often.

Multi-Container Endpoint – Each model operates within its distinct container and this gives enhanced adaptability as well as separation. Ideal for models with varied hardware specifications or runtimes. For efficiency, it offers superior adaptability but with more expense from handling independent containers.

Advantages:

- **Isolation:** separate containers keep models apart, so one model's load does not affect other models.
- **Adaptability:** each container receives optimization for its model's individual needs. For example more processing power (GPU) goes to a deep learning model but less goes to a simpler regression model.
- **Scalability:** This setup works well for models where hardware needs differ or access happens at diverse rates.

For example: Ayala Corporation uses retail analytics models and puts each inside its own container. Because one specific model does sophisticated image recognition, it needs a GPU to perform well. Another model runs basic statistics, so it just needs a CPU.

AWS Services:

- **Amazon ECS** organizes these containers - therefore, it permits execution plus dimension change for applications housed inside.
- **Amazon ECR** stores and takes care of container images.

Considerations:

Because each container is separated, it does increase costs for resources and infrastructure. But taking care of several containers increases the complexity when contrasted with employing just a single point.

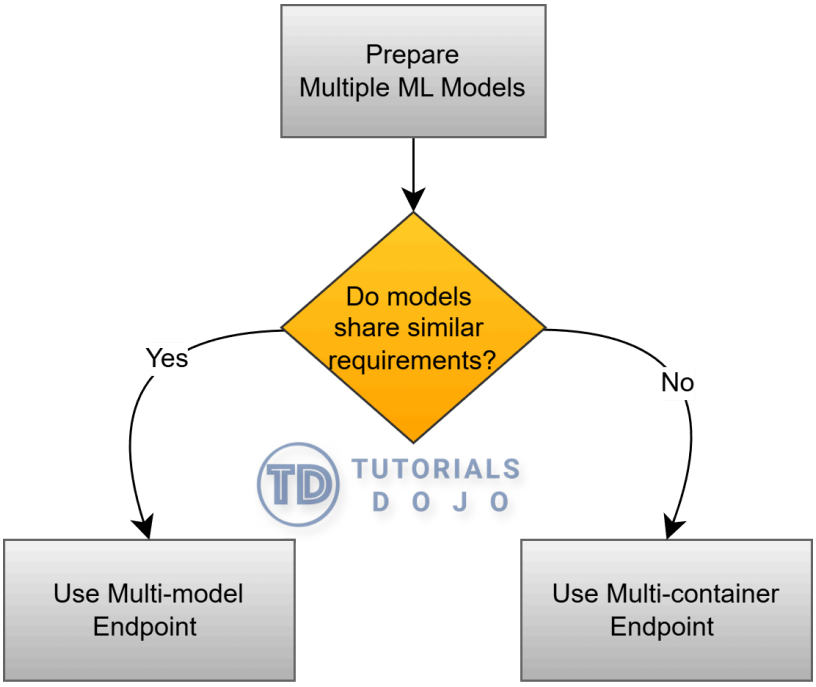
Cost vs. Performance Tradeoffs

Strategy	Cost	Performance	Best Fit
Multi-Model Endpoint	Low due to shared infrastructure	Varies due to cold starts and shared resources	Infrequent model usage, low resource needs
Multi-Container Endpoint	High due to isolated	High due to dedicated	High-performance

	resources per model.	resources per model.	models with varying requirements

Overall, **Multi-Model Endpoint** is cost-effective for infrequently-used models, but higher latencies are expected due to shared resources. **Multi-Container** offers better performance, but causes more expenses due to isolation of infrastructure.

Sample workflow:



B. Creating and Scripting Infrastructure for ML Models

Chapter 3.9 On-Demand vs Provisioned Resources

Choosing the proper resource model for cloud machine learning models is essential. You can pick on-demand or provisioned options. This chapter describes the variance among these models - it gives direction on scalability plus cost effects. Understanding this is vital for the MLA-C01 exam.

Differences Between On-Demand and Provisioned Resources

AWS offers two main methods to get resources:

- **On-Demand Resources:** With this choice, you pay solely for the computer power you utilize without any long-term commitments.
- **Provisioned Resources:** These are resources that are reserved for you ahead of time - typically with a commitment - usually at a lower rate. However, these resources are not flexible.

Quick Comparison Table:

Factor	On-Demand Resources	Provisioned Resources
Availability	Resources are immediately available when needed.	This requires prior reservation or allocation.
Pricing Model	Pay-as-you-go (usage-based)	Discounted rates for longer commitments
Cost Control	Flexible but can potentially offer higher overall costs	Predictable and but potentially lower overall cost
Scalability	Highly scalable, dynamic workloads	Limited scalability; fixed capacity

Understanding Scalability and Pricing Implications

Deciding between resources available when needed and resources set aside in advance has a big effect on how well your system expands and the complete cost.

1. **On-Demand Resources**

- **Scalability**: it works well for workloads that change quickly or that you cannot predict; They respond to high needs right away.
- **Pricing:** The pricing involves pay per-hour or per-second, and there's no upfront cost or commitment for each part.

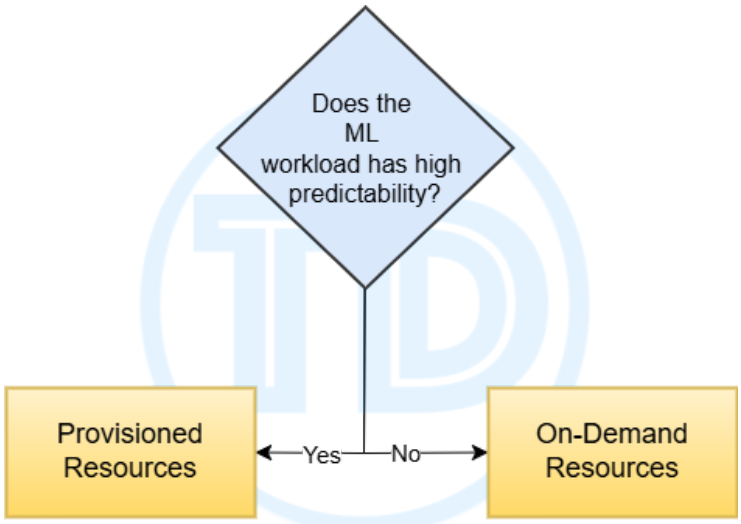
Example: Shopee Philippines, a popular e-commerce store, applies on-demand resources when needed during big sale days - 9.9 or 11.11. This allows them to deal with fast rises without preparing a lot of resources at the start.

2. **Provisioned (Reserved) Resources:**

- **Scalability:** It is limited to allocated, static resources, making it less than ideal for sudden scale but effective for steady, anticipated workloads.
- **Pricing:** Discounted hourly rates compared to on-demand and requires an upfront commitment (1-year or 3-year reservation).

Example: A Philippine bank, such as BPI, uses reserved resources to run its fraud detection models. Since the bank has predictable daily transactions, this makes reserved instances economical.

Decision Framework: Choosing Between On-Demand and Provisioned



Chapter 3.10 Scaling Policies and Their Tradeoffs

In cloud environments, the capacity to adapt to differing needs matters greatly. Auto-scaling is a robust method - it permits you to tweak the resources for your tasks per traffic or usage stats. This section examines auto-scaling rules, their advantages and trade-offs plus guidance on picking a right scaling system for your task.

What is Auto-Scaling?

Auto-scaling lets a cloud system change the quantity of computing power it uses depending on the usage of those resources. This resource includes things such as EC2 instances or containers, and then the system makes these changes on its own. It watches for certain situations based on alarms, such as how much traffic there is or what the status of the CPU usage is. AWS has several tools that allow autoscaling. The main ones are Amazon EC2 Auto Scaling plus Amazon ECS Auto Scaling.

Auto-scaling employs two main ways:

- **Vertical scaling** - also known as scaling up or down, changes the capacity of the current instances. For example, it adds more hardware CPU, memory or storage.

- **Horizontal scaling –** also known as scaling out and can also be scaling in, where it changes the number of instances because of traffic demands. After scaling, the traffic is loaded equally through an elastic load balancer.

Scaling Policies and Their Trade-offs

Selecting a proper scaling policy has importance when you put auto-scaling into action. It makes certain that workloads satisfy demands for performance without resources that are either too much or too few. Diverse scaling policies and their gains and losses get examined.

Types of Scaling Policies:

1. **Dynamic Scaling (Based on Metrics)** – This policy does a resource adjustment, a process based on metrics. CPU use, memory use, number of requests or specific metrics represent examples. For instance, there's upcoming traffic in business critical applications such as e-commerce platforms, or data processing jobs that have unexpected traffic.

   Advantages: It is expected for efficient resource utilization, where it will only scale when there is an increase in demand, and it's based on the usage itself.

   Trade-offs: May cause latencies when scaling up or scaling down when the metrics are not configured properly.

2. **Scheduled Scaling** – Scaling actions are taken depending on the schedule that has been fixed ahead of time. For instance, the instance will scale up only at 8 AM during business hours to accommodate

traffic properly. Ideal for predictable workloads such as processing in batches, jobs or applications that experience normal traffic patterns regardless whether it is weekends or weekdays.

Advantages: Low operation requirements, and has the opportunity to set up a fixed optimized resource allocation based on the known patterns.

Trade-offs*:* Not recommended for unpredictable traffic patterns.

3. **Target Tracking Scaling** – This policy adjusts resources to ensure that it maintains a certain metric based on a target value. For instance, you could set a policy to keep CPU utilization at 50% by adding or removing instances automatically. The alarms then can be based on the CloudWatch alarms. For instance, this is ideal for applications where a level of resource utilization is required.

   Advantages: It maintains the resources at an optimal level and in a steady state.

   Trade-offs*:* Selecting the target value before scaling in and out can be challenging, and cannot efficiently adapt to sudden traffic surges.

4. **Step Scaling** - Specific levels determine how resources are adjusted. For example a rule could add one EC2 instance when CPU use passes 70 % for ten minutes. It adds another when the use passes 85 %. This method works well when you anticipate big, quick traffic rises, like for flash sales or when new products are released.

   Advantages: This type of scaling aids in swift accommodation of peak times - additional resources appear in small steps.

   Trade-offs: If thresholds are too high during a sudden traffic increase, it may lead to under-provisioning. Else, if the thresholds are too low, it might lead to over-scaling and underutilized costs.

Chapter 3.11 Infrastructure as Code (IaC) Options

Machine Learning projects transition into production environments. You require a method for deployment and updates to the AWS infrastructure - it must be consistent, moreover scalable. Infrastructure as Code tools let you define the deployment environment through code. IaC creates secure ML infrastructures that reproduce easily. Two AWS tools support Infrastructure as Code: AWS CloudFormation plus AWS Cloud Development Kit, also known as CDK.

What is Infrastructure as a Code?

Infrastructure as Code involves managing and supplying your Amazon Web Services resources, like Amazon EC2 instances, Amazon S3 buckets and Amazon SageMaker endpoints, by using lines of command-like code through configuration files or scripts machines can read. With IaC you articulate your requirements in code. Instead of manual creation and setup of resources in the AWS Management Console - AWS later processes the details and sets up or amends, your infrastructure to match.

Benefits of Using IaC

- **Scalability** – With IaC, you replicate environments for various purposes such as development, testing, and production, without difficulty.
- **Reproducibility** – You can version-control your infrastructure, which allows you to roll back to a previous infrastructure state if needed.
- **Automation** – Reduces configuration by hand, cutting down errors from people.
- **Collaboration** – By letting teams work together on one code store for software and setup, IaC boosts cooperation, developing a "DevOps" or "ML Ops" approach.

Leveraging CloudFormation

AWS CloudFormation helps you describe AWS resources in template files - these files exist in JSON or YAML. As you send a template, CloudFormation provisions and configures the resources right away, without using manual operations, commonly known as “ClickOps”.

CloudFormation Foundations:

- **Templates** – A CloudFormation template holds the AWS resource configuration definitions within a document.
- **Stacks** – A single unit that consists of the current state of the AWS configuration, whether it’s deployed or roll-backed using CloudFormation. To produce a Stack, you upload the JSON or YAML template to CloudFormation.
- **Change Sets** – Before you update the resources, you have the option to review the adjustments to a running Stack before confirming the changes of the resources.

Pros	Cons
Declarative approach – where you describe what resources you need with lines of code before the AWS provisions them for you.	As the architecture becomes too complex, especially for ML pipeline architectures, the configuration might be harder to read.

Change Sets – safer deployments, and you only pay for the resources that you will provision. When a failure has been encountered, it will automatically roll back the deployment process.	**YAML / JSON** – isn't as flexible as a full programming language or other declarative IaC language.

Example: Amazon SageMaker Deployment with CloudFormation

A Filipino bank, *Juan-ML*, hopes for a uniform setting to host a credit risk model. In this case it can specify an Amazon SageMaker Endpoint, a VPC setup, also IAM Roles using only a CloudFormation template. This ensures every time they deploy a new environment, whether it's for development or production-level testing, this ensures every provisioning will have the same configuration.

[Optional] For this demo, we'll show only a part of the solution, which uses a cloud formation template that provisions an S3 bucket for your ML infrastructure needs. We kept it simple to follow if you also wish to test it. Copy the YAML syntax below, paste it into a text editor, and save it as a YAML file. Follow the next steps to upload it to CloudFormation.

Sample CloudFormation Template:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Simple CloudFormation template for ML infrastructure'

Resources:
  # S3 bucket for storing ML data, models, or artifacts
  MLStorageBucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'ml-storage-${AWS::AccountId}-${AWS::Region}'
      VersioningConfiguration:
        Status: Enabled
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256
      Tags:
        - Key: Project
          Value: MLInfrastructure
        - Key: Environment
```

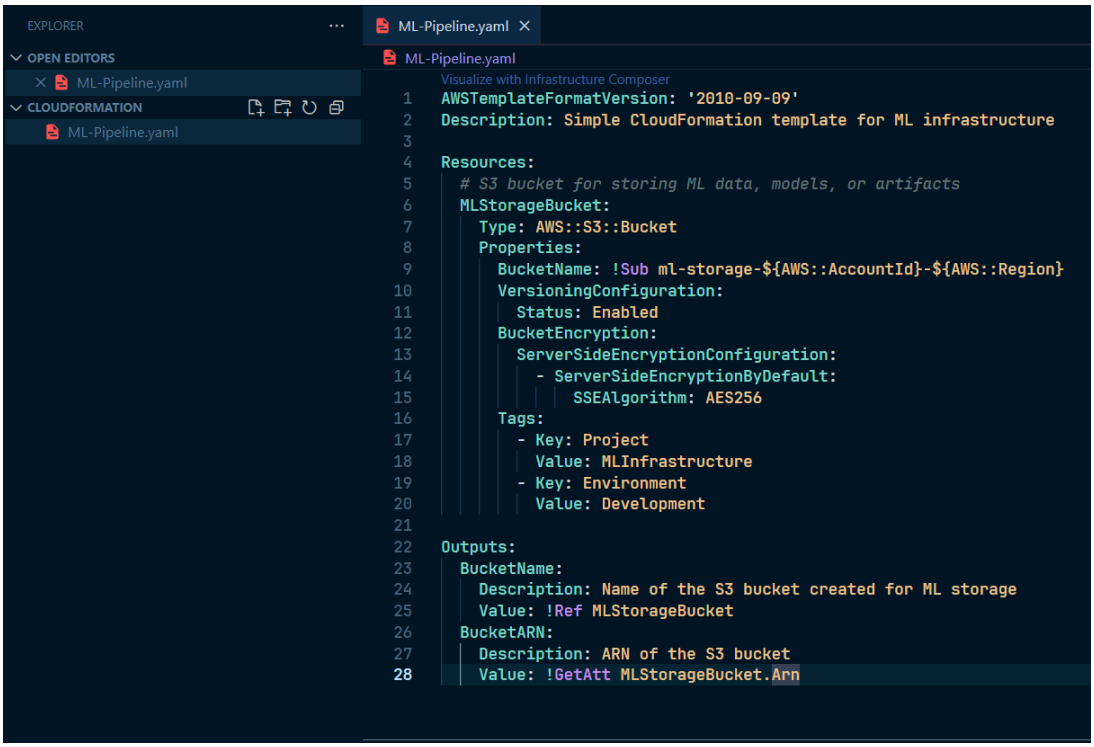
```
          Value: Development

Outputs:
  BucketName:    Description: 'Name of the S3 bucket created for ML storage'
    Value: !Ref MLStorageBucket
  BucketARN:
    Description: 'ARN of the S3 bucket'
    Value: !GetAtt MLStorageBucket.Arn
```

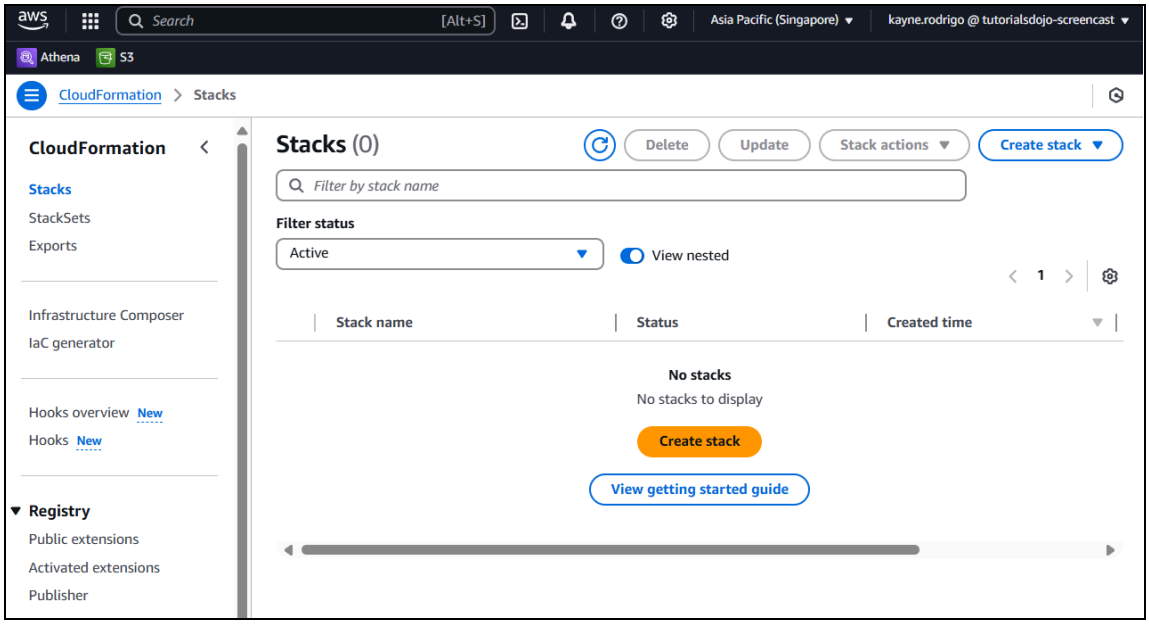
Steps to upload this CloudFormation template to AWS CloudFormation.

Note: You may want to open your AWS Management Console for this one, a free tier account will do. Also take note that the UI may change in the future as well.

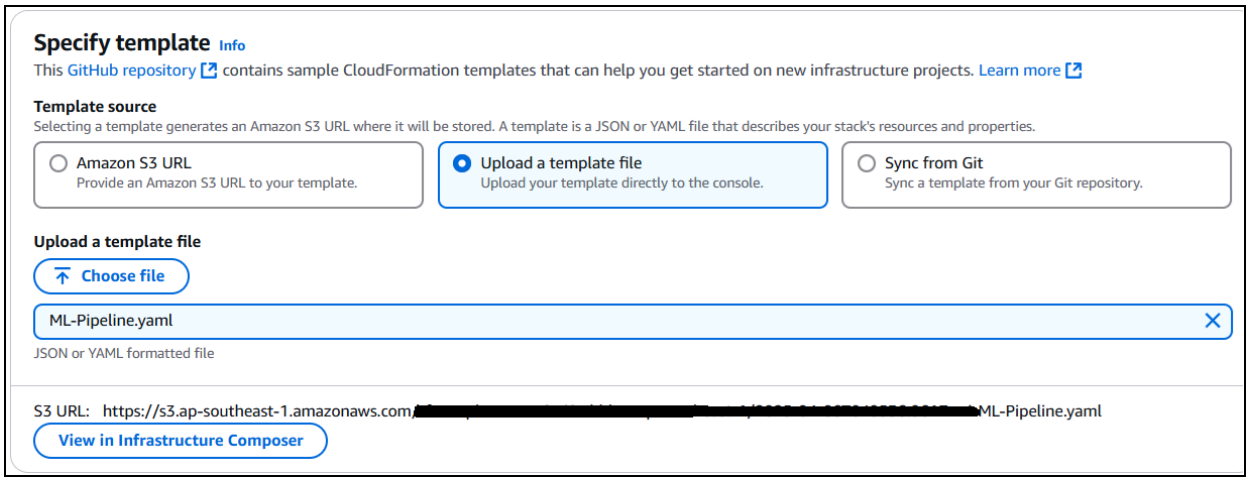
1. Copy the YAML syntax above, for context, it simply creates an S3 bucket for ML workloads.
2. Open any text editor (notepad, notepad++, visual studio code, etc.)
3. Paste the YAML syntax and save it as any filename you want. (ex. ML-Pipeline.yaml)



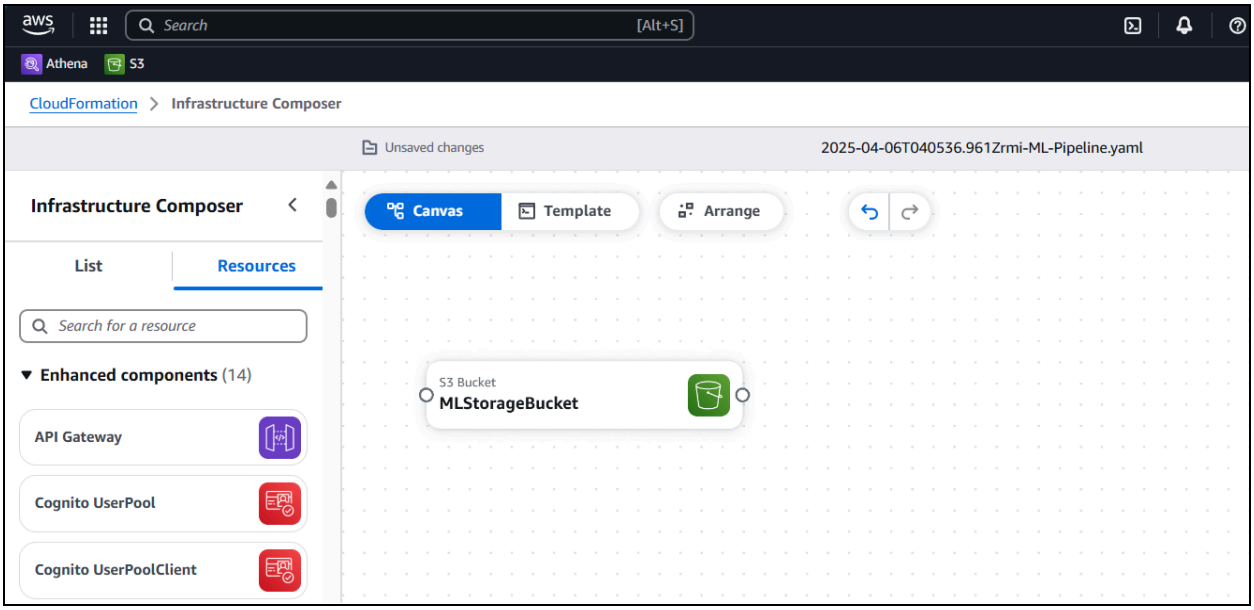
4. Go to your AWS Management Console, search for CloudFormation.



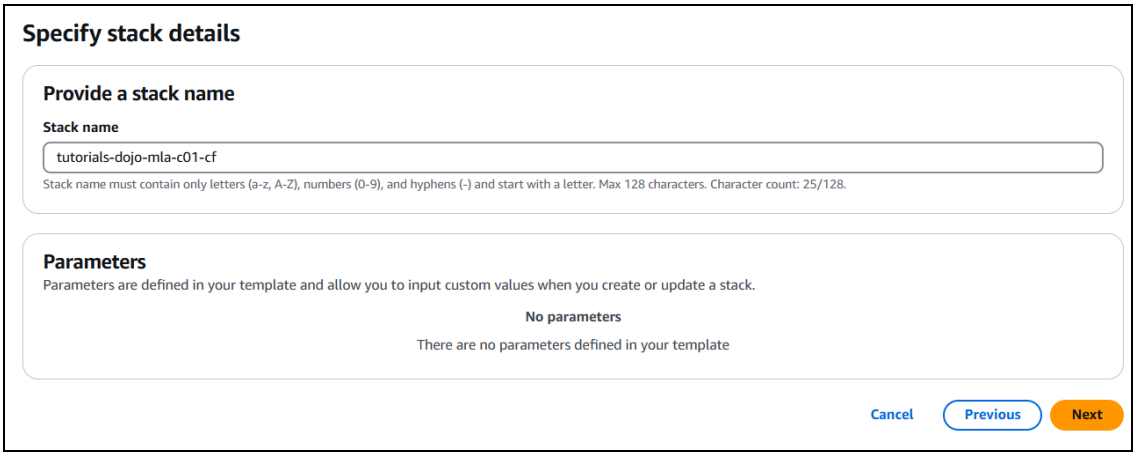
5. Click Create Stack, then select “Upload a Template File”.
6. Upload your template file.



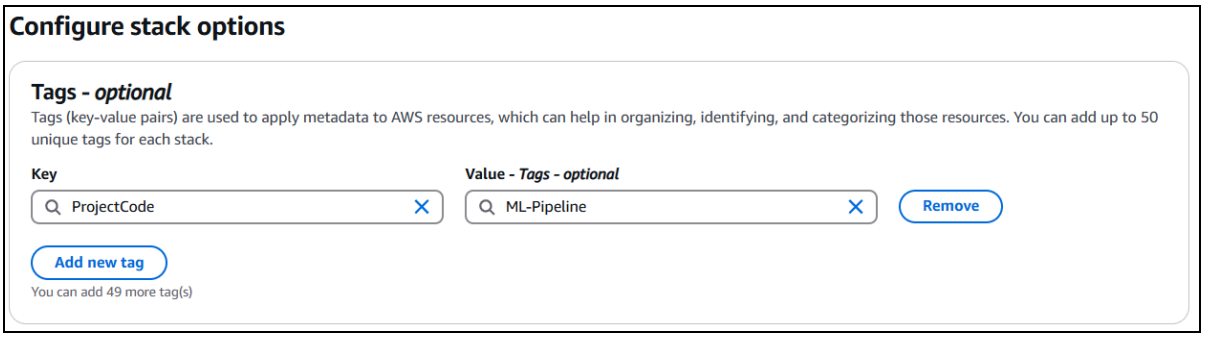
7. [OPTIONAL] You can view the setup in your Infrastructure Composer for a better UI.



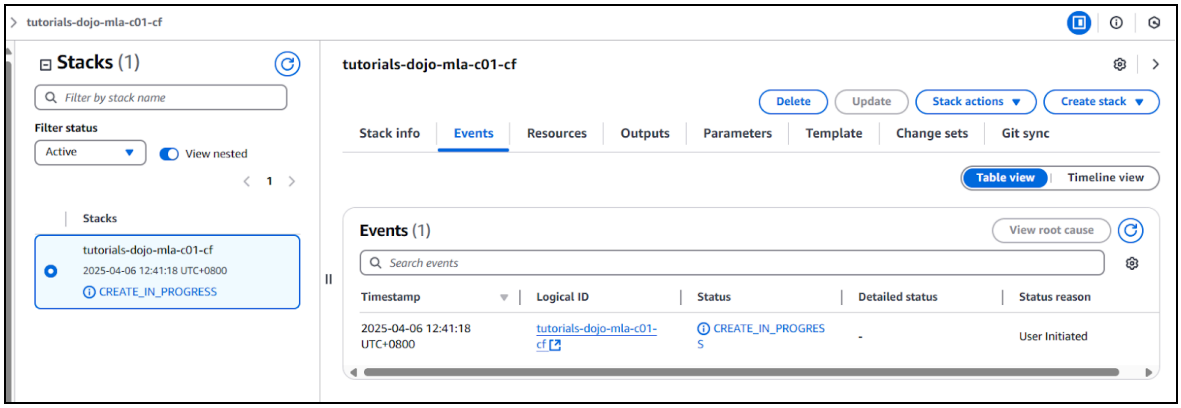
8. Give your stack a name, then click next.



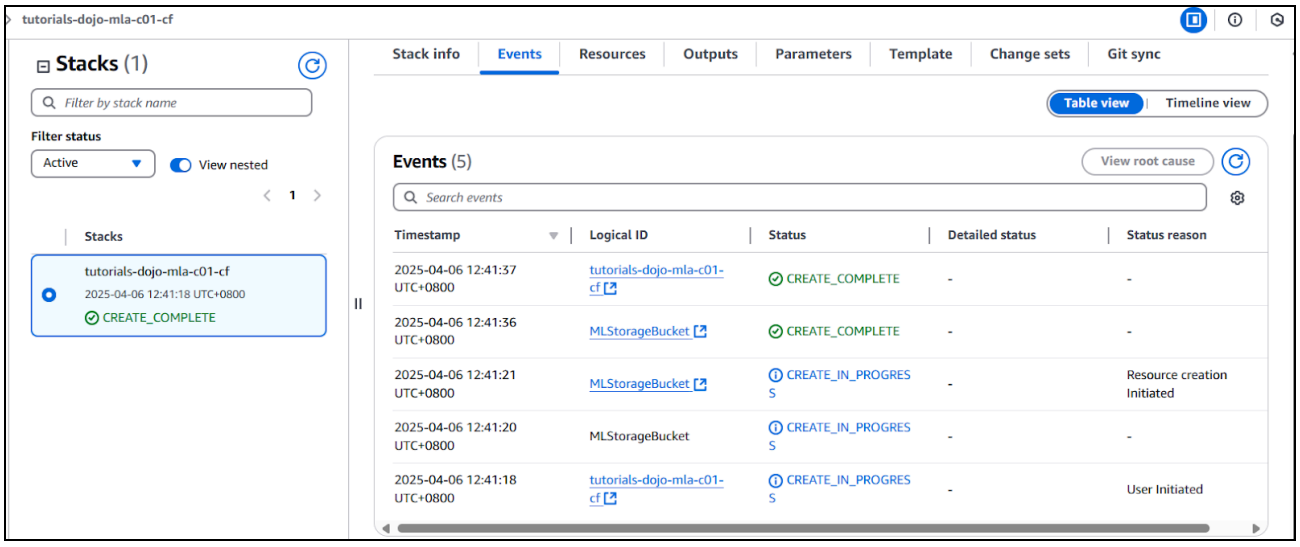
9. [Optional] Add tags if needed.



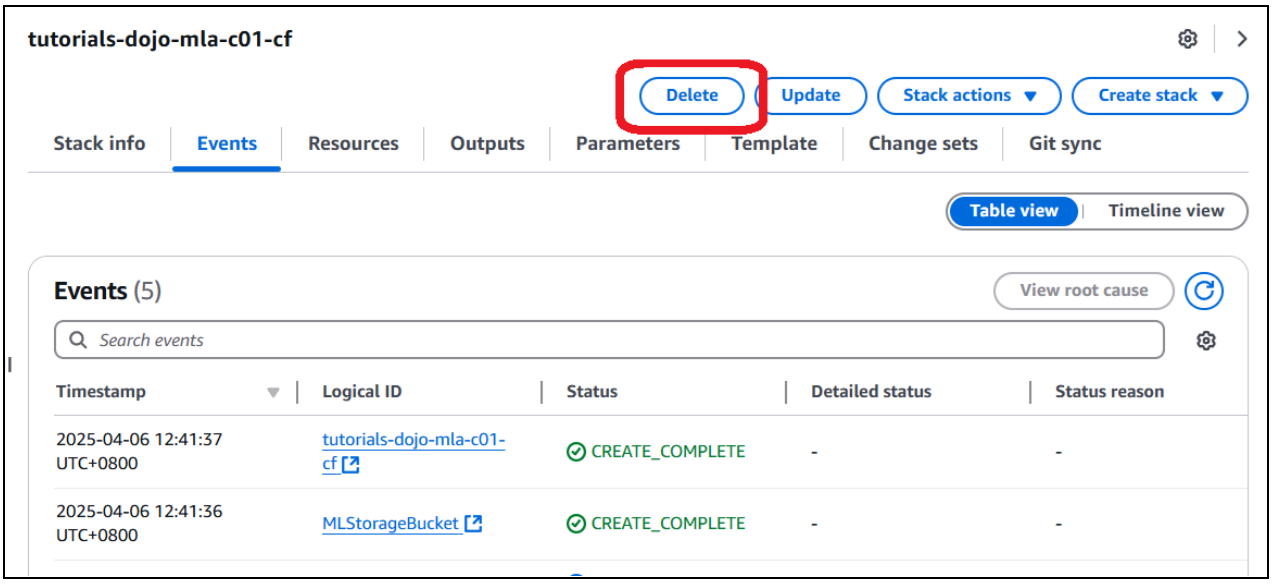
10. Click next, then under “Review and create” just click Submit for now.

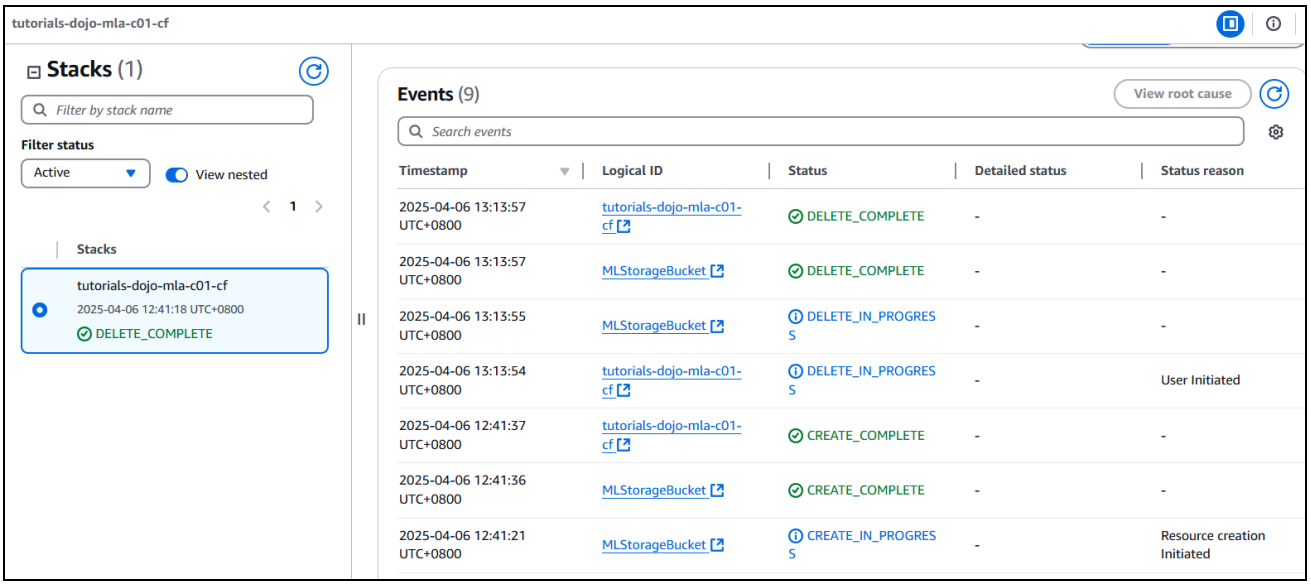


11. Wait until the Cloudformation rendering is complete. It will be labeled as “CREATE_COMPLETE”.



12. If finished, simply click “Delete”. The cloudformation status will be updated to “DELETE_COMPLETE”.





Chapter 3.12 Containerization Concepts

You can use containerization to package applications, even ML models with what they need, into units that move quickly. This capability makes them run consistently across different places where you compute. Because ML workloads greatly benefit from this approach, you can combine your specific libraries, frameworks

such as TensorFlow or PyTorch, and model binaries in one container image. AWS provides services that support the complete container lifecycle.

Why Containerize ML Models?

Portability – Containers offer stable environments starting from development up to production scale. As an example, it is feasible for someone to train a model in a local Docker setup, then execute that very container on Amazon ECS or EKS. Then only minimal changes can be expected.

Isolation – Every container operates separately, allowing you to isolate dependencies—like different library versions for different ML models.

Scalability – Container orchestrators, such as ECS, besides EKS, automatically enlarge or diminish containers. They base it on CPU, memory, or specialized data.

Standardization – Common container formats, such as Docker images, bring uniformity to how ML environments can be transferred to another team, making implementation almost seamless.

AWS Container Services

While these parts were introduced earlier, let's take a deep dive into each service to familiarize you.

Amazon Elastic Container Service (ECS)

Amazon Elastic Container Service (ECS) is a container service, expanding next to protected Docker containers. ECS works together with AWS services such as Amazon CloudWatch, AWS IAM along with Amazon VPC.

- **Launch Types** – For the **EC2 Launch Type**, you handle the EC2 instances where the containers operate. In contrast, the **Fargate Launch Type** is serverless, where AWS handles the compute operations and you state resource needs for containers.
- **Integration with SageMaker** – An application goes to a SageMaker endpoint. When a custom inference server is required, you can operate it on an ECS Service that reaches your model container.
- **Use Cases** – One use case involves sending custom ML inference code to the Docker containers that you handle. Another use case is operating batch data tasks, such as data cleaning or feature engineering, in containers.

Example: A Filipino-based e-commerce company, "TutorialDojo-ML," wants to spin up containerized inference tasks that handle sporadic traffic quickly. They use AWS Fargate to avoid managing EC2 clusters, paying only for the time their containers run.

Amazon Elastic Kubernetes Service (EKS)

Amazon EKS gives a controlled Kubernetes environment. It permits the operation of apps in containers on AWS. Kubernetes within AWS works as the system to orchestrate these apps. Kubernetes is an open-source system that automates the deployment, scaling, along with control of container applications. Similar to ECS on Fargate, EKS on Fargate lets you provision pods without running the underlying servers. If your ML work uses complex Kubernetes features, such as Kubeflow, then EKS could serve as a great option. This is the Kubernetes ecosystem at work.

Pros of EKS: any Kubernetes-based or third-party solution, like those for monitoring or growth, operates smoothly. There is an established free group that centers around ML tools, such as Kubeflow for ML pipelines and Spark on Kubernetes.

Cons of EKS: The learning process is a bit harder, since you must handle Kubernetes ideas, like pods, deployments next to services along with AWS details. EKS becomes excessive if you require simple container control for inference.

Example: A Quezon City data analytics startup called "TD-QC" plans to use Kubeflow for complex ML pipelines. To incorporate Kubeflow pipelines, they select EKS. Through EKS, TD-QC expects to execute hyperparameter tuning and multi-model scheduling.

Amazon Elastic Container Registry (ECR)

Amazon ECR exists as a container registry. Users store, manage, and deploy Docker container images with it. The repositories and images in ECR are private to your AWS account as a standard configuration.

- For security, IAM policies with precision dictate who accesses images for pulling or pushing actions.
- Tags give you the capability to version Docker images - model:v1 and model:v2 are examples.

In a usual workflow, one starts by developing a Dockerfile locally or in AWS CodeBuild. A user creates a container image. For example **docker build -t my-model:v1** fulfills this action. Authentication to ECR then occurs, followed by pushing the image. An example of this is **docker push <ECR_Repo>/my-model:v1**. After this, deploy the container image on ECS or EKS.

Sample: For an instance, a Filipino bank's ML engineer works on a XGBoost model. She creates a Docker image with a custom inference script. She then pushes the image to ECR. The engineer names it **credit-risk:v1**. ECS pulls this image from ECR when it starts new tasks.

[OPTIONAL] Demo: Building, Storing, and Managing Containers for ML

Here are some sample syntaxes in building, storing, and managing containers for an ML infrastructure.

Building your container image:

Dockerfile:

```
FROM python:3.9
RUN pip install boto3 sagemaker scikit-learn
COPY inference.py /app/inference.py
ENTRYPOINT [ "python", "/app/inference.py" ]
```

Local Testing: *(Test predictions with local requests)*

```
docker run -p 8080:8080 my-model:v1
```

Security: It is vital to prevent storage of secrets inside the Docker image. Perhaps you use AWS Secrets Manager or instead you use environment variables inside ECS or EKS. In addition, update base images often so you address possible vulnerabilities.

Storing Images in ECR

- **Command for creating your repository:** *aws ecr create-repository --repository-name <my-repo>*
- **Command for authenticating:** *aws ecr get-login-password --region ap-southeast-1 | docker login …*
- **Command for pushing:** *docker push <account_id>.dkr.ecr.ap-southeast-1.amazonaws.com/<my-repo>:latest*

Managing Images

- **Version Tagging**: You can tag your images via version or hashes, ex. `:v2`, `:commit123`.
- **Lifecycle Policies**: You can define your lifecycle policy for ECR.
- **Automated Scans**: ECR can assist you in scanning images, then inform you for any vulnerabilities.

Container-Based Model Deployment Patterns

Single-Model Container – One model exists inside each container image. It is simpler to manage, yet costs might rise when you have numerous models.

Multi-Model Container – You can package several models into a single image. You can keep them on Amazon S3 and then load them. This approach works well for a "model hub," though the container needs more complex instructions to direct requests toward the correct model.

Sidecar Container – For advanced configurations, you can operate a "model server" container plus a "logging or metrics" container. They work together on the same ECS task or Kubernetes pod.

Example: PAL-ML-Express, a logistics business, prepares a route-optimization model in SageMaker. The firm desires specific oversight of its inference endpoint.

1. **Build a Docker image.** It should include a `route_inference.py` and a simple web server.
2. **Push** image to ECR under `pal-ml-express/route-optimizer:v1`.
3. **Run** it on ECS Fargate that includes a single `task definition`, and the application load balancer is in front of a container.
4. During a traffic increase, ECS automatically provisions new tasks via auto-scaling. If this happens, they pull the container image from ECR then provisions it.

Chapter 3.13 Auto Scaling and Compute Provisioning

Good resource management cuts expenses a lot plus assures steady function for machine learning projects. This chapter shows how one sets up auto-scaling for Amazon SageMaker endpoints. It describes how one handles on-demand instances, instead of Spot Instances, for training or inference. It joins AWS Lambda with SageMaker, so model serving becomes economical.

Introduction to Auto Scaling in SageMaker

Amazon SageMaker includes features to help change infrastructure size in response to changing machine learning endpoint traffic.

- **Endpoint Auto Scaling** – adjusts the count of inference instances for real-time endpoints. The adjustment uses setup metrics or a set timetable.
- **AWS Application Auto Scaling** – functions under the surface. It handles coordination of scaling actions.

- **Scheduled Scaling** – lets you raise or lower capacity at specific times. An example involves working times against off-hours within the Philippines.

This is why this is significant:

- **For optimal performance:** the system supplies ample calculation power automatically, so delays stay minimal.
- **Cost optimization:** it occurs because the system prevents charges for unused resources amid slow periods. This works well for workloads with irregular or uneven use.

Configuring Auto-Scaling for SageMaker Endpoints

Overview of SageMaker Endpoints: If you establish a model on a SageMaker real-time endpoint, you determine the instance type, such as CPU, GPU or a special type like ml.inf1. You state the original number of instances, either the lowest count or the count you prefer. In addition, you determine the greatest instance number you permit.

Steps to Enable Auto Scaling:

- **Create an Auto Scaling policy** using the **Management Console CLI or the Application Auto Scaling API.** You may refer to this CLI command.

```
aws application-autoscaling register-scalable-target \
  --service-namespace sagemaker \
  --resource-id endpoint/<EndpointName>/variant/<VariantName> \
  --scalable-dimension sagemaker:variant:DesiredInstanceCount \
  --min-capacity 1 \
  --max-capacity 10
```

- **Create or update a scaling policy** that references the scalable target:
  - Develop a scaling policy plus connect it to the scalable target.
  - Determine the type of policy: it is either TargetTrackingScaling or StepScaling.
  - Pick a metric for Target Tracking. SageMakerVariantInvocationsPerInstance or CPUUtilization work well.

- Set **InvocationsPerInstance** to a target of 70 %. If the number of requests each instance faces goes beyond this point, the system increases capacity.

- **Validate by directing traffic to the endpoint.** Confirm that automatic scaling occurs.

Common Scaling Metrics

- **CPUUtilization:** For a CPU-bound model, target CPU usage, like 70 %, shows when the system requires added instances.
- **InvocationsPerInstance:** It is standard practice to check requests for each instance. Once a limit passes, it begins a scale-out.
- **Latency-based:** ModelLatency or OverloadRequests determine when an instance must get added.

Integrating Spot Instances for Cost Savings

Spot Instances in SageMaker

Spot Instances lower compute costs, sometimes significantly. In SageMaker, there are two usage patterns.

- **Managed Spot Training** for training jobs exists. When this option exists, SageMaker employs available EC2 capacity. This suits workloads prepared for disruptions - for example, ones which checkpoint a model.

- **Real-time endpoints** do not normally permit straight Spot employment for inference. But you can implement custom setups to run containers that reach SageMaker. For example you can use ECS or EKS with Spot. Replicate the inference process.

Best Practices

- **Checkpointing matters:** It saves a model's temporary state to Amazon S3. A job failure does not waste effort and a restart uses the latest saved data.

- **Model Partitioning:** it ensures data about partial results or partial training states stay safe using routine saves.

Example: In Manila, a new company utilizes Managed Spot Training with a text-classification model. The system saves progress to S3 at each epoch, so SageMaker begins from the last saved point if Spot Instances take back the resource. Because of this approach, the startup reduces training costs for its limited training schedule.

AWS Lambda for Model Serving

SageMaker is a key service to host plus auto-scale ML endpoints, but AWS Lambda with tailored containers or small frameworks works in certain situations. For example, models see low traffic. A SageMaker endpoint that runs continuously might be expensive if the model processes just a few requests per hour. Lambda functions can serve when S3 events or a queue triggers an inference job. The function loads the model from S3 or EFS, fulfills the inference and then deactivates.

Architecture:

- A Lambda function holds your model or a pointer to your container within ECR.
- Events in Amazon API Gateway or Amazon S3 create the trigger.
- Inline inference occurs. Because a model can get large, EFS use is advised. Partial data can receive caching in short-term storage.
- Automatic scaling occurs, which relies on concurrent Lambda invocations.

Trade-Offs:

- **Cold Starts:** Booting large models can cause lengthy initial delays.
- **Memory Limits:** Lambda provides a memory limit of 10GB, though that amount is possibly too small for significant machine-learning models.
- **Stateful vs. Stateless:** Because Lambda has no state, SageMaker or ECS/EKS could work better when the model demands heavy GPU computation or it relies on lasting context.

Multi-Variant and Multi-Endpoint Approaches

Multi-Variant Endpoints – A/B testing different model versions or hosting separate hyperparameter variants. Automatic scaling is adjustable for each variant. Separate scaling policies are applicable.

Blue/Green or Canary Deployments – deployed a new version plus call it "green." After that, user traffic will be moved gradually from the "blue" version. Take the "blue" version out of service, provided that the "green" version is stable. For Canary, send a small fraction of user traffic, such as 10 %, to the endpoint. After that, if everything functions correctly, more traffic will be routed.

Putting It All Together: A Real Example

Scenario: Consider TagalogClips, Inc. The company must run a captioning service driven by AI. They foresee large user volumes in the evenings.

1. A SageMaker Endpoint utilizes ml.c5.xlarge instances, starts with two units and can expand to ten.

2. Target tracking policy on InvocationsPerInstance=50.
3. Setup scheduled scaling from midnight to 6 AM (Philippine time), reduce max capacity to 3.
4. In addition Spot Training retrains the system each night and stores snapshots in S3.
5. When occasional uploads occur outside peak times, a Lambda function gives a second route for making guesses, especially if the endpoint is small. This handles VIP requests.
6. The result features lower costs, brief delays along with flexible scalability.

Chapter 3.14 Automating Provisioning with CloudFormation and AWS CDK

Automatic setup of AWS resources proves important for stability, size adjustment along with effective operation. AWS makes available two strong tools to automate resource setup: AWS CloudFormation and the AWS Cloud Development Kit (CDK). We examine good processes to use these tools well, so we make infrastructure control plus deployment faster.

A review to CloudFormation and AWS CDK

AWS CloudFormation plus AWS CDK are both Infrastructure as Code (IaC) tools. These tools define and provision AWS infrastructure with code and they confirm deployments stay consistent.

- **CloudFormation** is a declarative service, so it defines AWS infrastructure in templates. You can employ YAML or JSON. CloudFormation obtains these templates. It provisions the designated resources in the proper order with just one command.

- **AWS Cloud Development Kit (CDK)** is a higher-level, imperative IaC tool. This tool defines cloud resources using programming languages, like TypeScript, Python, Java or C#. Because of this CDK gives a flexible and rich approach compared to CloudFormation.

Effective CloudFormation Usage – The aim is template reuse besides easy upkeep. Divide templates into smaller, modular stacks. Each stack represents a rational part of your application. For example it can be networking, security or compute elements. An illustration includes distinct templates. They cover VPC, EC2 instances, RDS instances as well as IAM roles.

Leverage Parameters, Outputs, and Conditions – The target includes template adaptability plus adjustability. Use parameters - they accept values during runtime. Outputs reveal resources. Conditions define behavior specific to an environment. This relates to production against development. An example shows passing the

VPC CIDR block as a parameter. It involves output of an EC2 instance's DNS name for additional uses. In addition it uses conditional resource creation based on the environment.

Version Control and Change Sets – The aim is to handle CloudFormation templates like code. Store them inside Git, a version control system, so you follow alterations through time. Always employ change sets so you observe what will happen prior to use in an environment. For example, prior to an update to production resources, construct a change set. This lets you scrutinize any changes planned.

Automate Stack Updates with Change Sets – The target is to constantly manage resource updates with change sets. This process sidesteps sudden alterations. It also makes sure changes stay secure and reliable. As an example you can update a CloudFormation stack through a change set. Then you know it correctly modifies resources like security groups or IAM roles.

Implement Drift Detection – A purpose is to ensure resources equal their planned state. AWS CloudFormation Drift Detection enables a comparison. It contrasts resource states right now with the state outlined in a template.

Best Practices for Using AWS CDK

a. **Choose the Right Language** – The aim is to employ the programming language best suited for you and your team's abilities. CDK provides support for a few languages, such as TypeScript, Python, Java plus C#.

   Example: Suppose your team knows TypeScript well. Use CDK's TypeScript support. This action gives access to traits, like autocompletion, type verification as well as IDE assistance.

b. **Use Constructs for Reusability** – The aim here is to wrap typical infrastructure models into constructs that you reuse. This develops modularity, enhances reusability along with cuts down code repetition.

   Example: For example, create a construct. It should handle an application load balancer accompanied by a collection of EC2 instances. This construct then works again plus again in various services.

c. **Utilize CDK's strong constructs.** – The objective is to employ CDK's powerful constructs for services like Amazon S3, DynamoDB or Lambda. These constructs simplify many configuration details and they enable simpler creation of intricate resources using less code.

   Example: For instance you can use s3.Bucket or lambda.Function constructs. This defines resources with reasonable default settings.

d. **Deploy with pipelines.** – The intention is to automate infrastructure deployment via AWS CodePipeline. To that end you must use CDK to specify the pipeline stages. It guarantees that infrastructure is provisioned and updated on its own as a piece of a CI/CD process.

   Example: you can make a pipeline. When a code repository gets a change, this pipeline begins infrastructure provisioning.

e. **Test CDK Code Locally** – CDK lets you create CloudFormation templates on your computer. This helps greatly when you check modifications before the launch.

   Example: For example you can apply 'cdk synth' to make the CloudFormation template locally and you can employ 'cdk diff' to see what changes occur.

f. **Employ AWS CDK Pipelines for Multi-Account Deployments** – AWS CDK Pipelines supply a method, so you can put resources into diverse AWS accounts and regions. Because of that it becomes quite handy for launching infrastructure in various stages, like dev, test or production.

   Example: Automate deployment of infrastructure across multiple environments (e.g., separate accounts for staging and production) using CDK Pipelines.

General Best Practices for Both CloudFormation and AWS CDK

Automate with AWS CodePipeline – The aim is integration of CloudFormation or CDK with AWS CodePipeline. This provides complete automation. The pipeline then automates code deployments, infrastructure provisioning and testing. For example a CodePipeline starts when a repository change occurs. It updates CloudFormation stacks or deploys CDK applications.

Version Control Infrastructure – Another aim is use of Git or a different version control system. This tracks infrastructure code. It ensures collaboration and you follow changes made to your provisioning scripts. As an example, store CloudFormation or CDK code in a GitHub repository. Versioning, code review along with collaboration happen.

Watch besides follow the resources – The objective is to put monitoring into the setup routine. Use Amazon CloudWatch alongside AWS Config to watch the shape of resources set up by CloudFormation otherwise CDK. For example place CloudWatch alarms so it tells you once a set resource (like an EC2 instance or RDS database) passes a limit.

Set labels for cost control – The goal involves placing labels on resources at setup to follow costs and usage. Labels let you give costs and resources to separate teams, maybe departments. In other words use

cdk.Tags.of(resource).add('Environment', 'Production') inside CDK to label all resources and track costs inside AWS Cost Explorer.

Security plus Compliance – Adhere to the least privilege principle when you give IAM roles in CloudFormation or CDK. Access becomes restricted to needed resources using AWS Identity plus Access Management (IAM) policies. For example the tool iam.PolicyStatement helps restrict access to definite resources such as S3 buckets or EC2 instances in CDK.

Conclusion

Using AWS CloudFormation or AWS CDK to automate resource provisioning adds consistency, efficiency next to scalability to your cloud architecture. By adopting the recommended practices detailed in this chapter, you ensure your infrastructure remains maintainable and secure. As you use these tools, remember the value of version control, modular design along with continuous monitoring to keep your AWS environments functional. For instance CloudFormation gives a declarative method but AWS CDK provides a programmatic method. Both offer robust ways to automate your infrastructure.

Chapter 3.15 Hosting and Configuring SageMaker Endpoints

Putting machine learning models into use with Amazon SageMaker requires deployment as live endpoints. This chapter emphasizes good methods for setting up SageMaker endpoints and making sure they run safely inside a Virtual Private Cloud (VPC). It addresses optimizing endpoints for automated scaling. We explore how to set up significant performance measurements, for instance, model delay plus CPU use. These configurations permit automated endpoint scaling to satisfy shifting requirements.

Review to SageMaker Endpoints

Amazon SageMaker helps you put machine learning models into operation with ease by using SageMaker Endpoints. These endpoints are web services that exist to give fast predictions. They suit applications that want quick answers, for example, systems that detect fraud, recognize images along with giving instant recommendations.

- **Real-Time Inference**: Once a model concludes its training besides evaluation, you can install it as a SageMaker endpoint to deliver predictions.

- **Managed Infrastructure**: SageMaker deals with the supply, growth, in addition to upkeep of the systems required to serve the model.

Deploying SageMaker Endpoints Within a VPC for Enhanced Security

A virtual private cloud is important for the security of SageMaker endpoints. It establishes network separation for models. By using a VPC to host endpoints, administrators manage access. They use VPC Security Groups plus Network ACLs for added defense on top of the standard setups.

Benefits of VPC Hosting:

- It keeps SageMaker endpoints out of reach of the public internet.
- It determines which EC2 instances or services gain entry to the endpoint.
- It permits integration with other confidential VPC resources, for example, databases and internal apps.

Steps to Deploy a SageMaker Endpoint in a VPC

1. Construct a virtual private cloud. You need a VPC. It requires subnets and security groups. You have two options: use an older VPC or form a fresh one.
    - To keep your SageMaker endpoint secure, **situate it inside a private subnet**. This action stops any access from the public internet.
    - Set up **security groups** so they grant access only to safe, known sources. These sources could include application servers or internal services.
2. **Configure Endpoint Deployment**. During model deployment to SageMaker, you must say which VPC subnets and security groups the endpoint must employ. To specify these VPC settings, you choose the VpcConfig parameter in the SageMaker endpoint configuration.
3. **Grant SageMaker Necessary Permissions**. For SageMaker to get to items within the VPC, it needs permission. Assign the correct IAM Role. It must have the needed VPC permissions. You must link this role to your SageMaker endpoint configuration.
4. **Verify the Connection.** After deployment, confirm the endpoint works from the proper resources inside your VPC. Use VPC flow logs to watch traffic besides guarantee safe communication.

Key Metrics for Monitoring and Scaling

Good performance plus cost depend on observing your SageMaker endpoint's key measures. CPU utilization next to request count gives you this understanding. They show how well the model works and they guide choices about scaling.

Model Latency: Model latency is how long the model needs to send back a prediction after it gets a request. Observing latency tells you whether your model gives answers at a quick, acceptable rate.

CPU utilization: measures the amount of processing power the endpoint instance uses. If CPU use is high, you may need stronger instances or you could use auto-scaling.

Request Count: indicates the volume of incoming requests for predictions. When requests quickly increase, auto-scaling could become vital.

Configuring Auto-Scaling for SageMaker Endpoints

Auto-scaling adjusts automatically the count of instances that support your SageMaker endpoint. It does this based on how much work the system needs to perform. Through the use of AWS Application Auto Scaling, your SageMaker endpoint expands or shrinks as traffic changes.

Enable Auto-Scaling – To enable auto-scaling, go to the SageMaker Endpoint and use the Application Auto Scaling feature. You need to configure scaling policies - these policies specify the least besides greatest count of instances for the endpoint.

Define Scaling Metrics – the first one is CPU Utilization. You set a target for CPU usage (for example, 60 %). This target triggers actions related to scaling. Define a limit for model latency. For instance you can set a 200ms threshold. If the latency goes above this limit, it initiates scaling actions.

Scaling Policies – Describes how an endpoint changes its size. The **scale-out policy** tells when to make the endpoint bigger. For example it might grow when CPU usage rises above 70 % or when the number of requests goes past a set limit. A **scale-in policy** sets when to shrink the endpoint. For example it might become smaller when CPU usage falls under 30 %.

Test Auto-Scaling Behavior – To test auto-scaling, simulate activity. Send several requests to the endpoint - this checks if auto-scaling starts as it should. Use Amazon CloudWatch - this watches the scaling events plus confirms the endpoint scales according to the metrics.

Monitoring and Fine-Tuning Auto-Scaling

- CloudWatch Alarms should monitor main metrics like latency, CPU utilization next to request count. You receive notifications when the numbers pass set limits, this helps in tracking performance and scaling.
- Examine the scaling action. Change the scaling thresholds and policies for enhanced performance plus cost reduction.
- When your model runs with high efficiency, refine the scaling rules to be less reactive. This action lowers costs. However, if the model has substantial latency or CPU usage, expand resources faster.

Best Practices for Hosting SageMaker Endpoints

Use Multi-Model Endpoints for Cost Efficiency: To lower costs use multi-model endpoints in SageMaker. These permit you to locate several models on one endpoint, not deploying discrete endpoints for each model. This enhances resource use, besides.

Optimize Endpoint Instance Type: You must pick an accurate instance type for your model based on computational needs. Begin with lesser instances such as ml.m5.large, then make it bigger as needed. For real-time inference, implement GPU-enabled instances, ml.p3 or ml.g4dn, for complex learning models which demand important computational power.

Implement SageMaker Model Monitor: This lets you examine your deployed model's functionality. By doing so you can sense data skew or concept drift as time passes.

Conclusion:

Having SageMaker endpoints inside a VPC lets you handle your models safely and well. If you watch important stats like model delay, CPU usage next to the number of requests and then use those stats to make the system scale automatically, your endpoint stays quick, saves money along with growth as needed.

With the good methods this chapter gives, like putting endpoints in a VPC, turning on auto-scaling and picking the best instance types, you utilize SageMaker endpoints to make machine learning apps that work strongly besides handling real-world use.

Domain 3: Deployment and Orchestration of ML Workflows Sample Questions

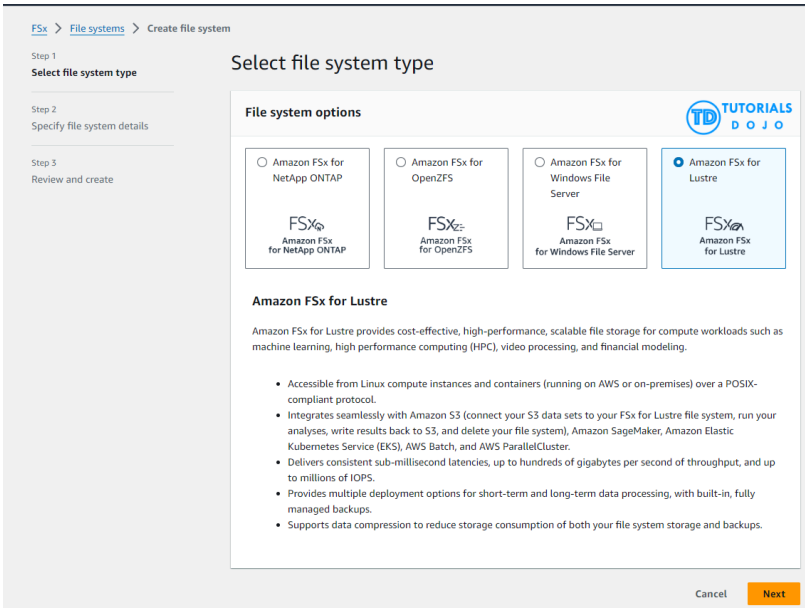
1. A data scientist is working for a company that specializes in developing machine learning models for large-scale video processing tasks. The current project involves training a deep-learning model using a massive dataset of high-resolution videos. The dataset is stored in a distributed file system, and the training process requires high throughput and low latency access to the data. Additionally, the training process involves frequent read and write operations, and the storage solution must be able to handle the high I/O demands efficiently.

   Which of the following storage services will meet the requirements?

   a. Amazon File Cache
   b. AWS Storage Gateway
   c. Amazon Elastic File System (Amazon EFS)
   d. Amazon FSx for Lustre

   Explanation:

   Amazon FSx for Lustre is designed specifically for high-performance computing (HPC) and machine learning workloads. It provides sub-millisecond latencies and supports hundreds of gigabytes per second of throughput, which is ideal for training deep learning models with large datasets.



FSx for Lustre efficiently handles the high I/O operations required for processing high-resolution videos. It is optimized for both sequential and random access workloads. It can seamlessly integrate with Amazon S3, allowing for efficient data access and management, which is beneficial when dealing with massive datasets typically used in machine learning.

Hence, the correct answer is: **Amazon FSx for Lustre.**

The option that says: **Amazon Elastic File System (Amazon EFS)** is incorrect. While Amazon EFS provides scalable file storage for use with AWS Cloud services and on-premises resources, it is designed more for general-purpose workloads rather than high-performance computing.

The option that says: **AWS Storage Gateway** is incorrect because it is primarily designed to connect on-premises environments to AWS storage services, providing seamless integration and hybrid cloud storage solutions. It is not optimized for high-performance computing or the specific needs of deep learning model training. Its primary use case is for data backup, archiving, and hybrid cloud storage scenarios.

The option that says: **Amazon File Cache** is incorrect because it is only used to cache frequently accessed data, which does not align with the requirements of continuous and intensive read/write operations needed for training deep learning models.

References:
https://docs.aws.amazon.com/fsx/latest/LustreGuide/what-is.html
https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html

Check out this Amazon FSx Cheat Sheet:
https://tutorialsdojo.com/amazon-fsx/

2. A data scientist at a healthcare company has developed multiple machine-learning models using R. The data scientist wants to deploy these models on Amazon SageMaker. The main goal is to minimize operational overhead while ensuring efficient deployment and management of the R models.

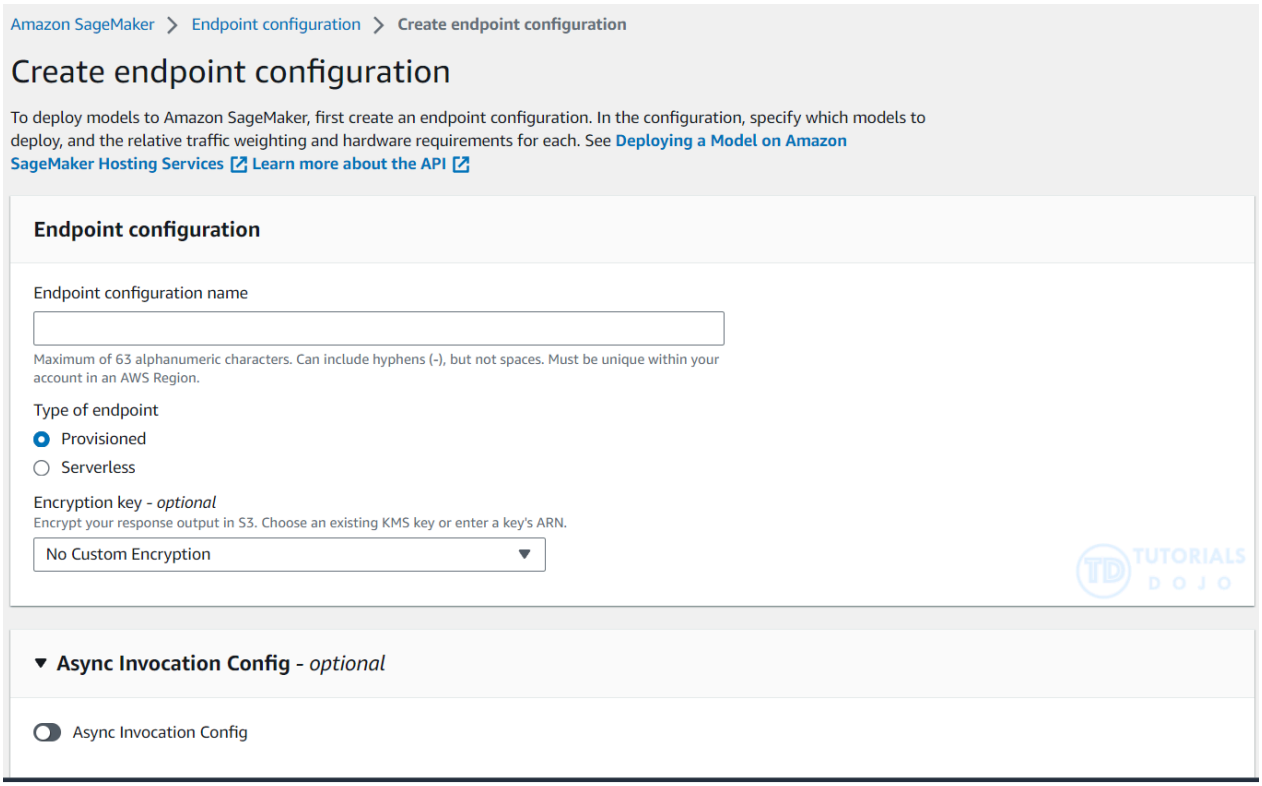
   Which solution should the data scientist choose?

   a. Use the SageMaker Python SDK to convert the R models into Python models and deploy them using SageMaker's built-in Python containers.
   b. Manually install R and all necessary dependencies on an EC2 instance, then use SageMaker's local mode to deploy the models.
   c. Use SageMaker's built-in R kernel to develop and deploy the models directly from a Jupyter notebook.

d. Create a custom R Dockerfile, build the Docker image locally, push it to Amazon ECR, and use it to create a SageMaker endpoint.

Explanation:

SageMaker notebook instances come with a pre-installed R kernel, which includes the reticulate library. This library provides an R to Python interface, enabling you to utilize the features of the SageMaker Python SDK directly within an R script. Additionally, Amazon SageMaker offers RStudio as a fully managed integrated development environment (IDE) within the SageMaker domain. With this integration, you can launch an RStudio environment to run your RStudio workflows on SageMaker resources seamlessly.



Hence, the correct answer is: **Create a custom R Dockerfile, build the Docker image locally, push it to Amazon ECR, and use it to create a SageMaker endpoint.** This option allows the data scientist to package all necessary R dependencies and the models into a Docker image. By pushing this image to Amazon ECR (Elastic Container Registry), it can be easily managed and deployed on SageMaker. This approach minimizes operational overhead because it leverages Docker's containerization capabilities, ensuring a consistent and reproducible environment for the R models.

The option that says: **Manually install R and all necessary dependencies on an EC2 instance, then use SageMaker's local mode to deploy the models** is incorrect because this option involves a lot of manual setup and maintenance, which typically increases operational overhead. Managing dependencies and configurations manually on an EC2 instance can be error-prone and time-consuming, making it less efficient than Docker containers.

The option that says: **Use the SageMaker Python SDK to convert the R models into Python models and deploy them using SageMaker's built-in Python containers** is incorrect because converting R models to Python models can be complex and may not always be feasible, especially if the models rely on R-specific libraries and functionalities. This approach also introduces additional steps and potential compatibility issues, increasing operational overhead.

The option that says: **Use SageMaker's built-in R kernel to develop and deploy the models directly from a Jupyter notebook** is incorrect. While using SageMaker's built-in R kernel might seem convenient, it is not primarily designed for the production deployment of models. This approach is more suitable for development and experimentation rather than efficient and scalable deployment, leading to higher operational overhead in a production environment.

References:
https://docs.aws.amazon.com/sagemaker/latest/dg/build-your-own-processing-container.html
https://docs.aws.amazon.com/sagemaker/latest/dg/r-guide.html

Check out this Amazon SageMaker Cheat Sheet:
https://tutorialsdojo.com/amazon-sagemaker/

References for Domain 3

Model Registration Deployment with Model Registry

Deployment guardrails for updating models in production

Working with deployment configurations in CodeDeploy

What is Amazon CloudWatch?

What is Amazon SageMaker AI?

Deploy models for inference

Real-time inference

What is AWS CodePipeline?

Batch transform for inference with Amazon SageMaker AI

Automatic scaling of Amazon SageMaker AI models

Configure security in Amazon SageMaker AI

Train a Model with Amazon SageMaker

Amazon EC2 Instance types

Model performance optimization with SageMaker Neo

Deploy models for inference

Supported Frameworks, Devices, Systems, and Architectures

Amazon SageMaker AI Cheat Sheet

Amazon EC2 Pricing

Amazon EC2 Reserved Instances

AWS Pricing Calculator

Docker Docs

What is Amazon Elastic Container Registry?

What is Amazon Elastic Container Service?

What is Amazon EKS?

AWS Samples

What is Application Auto Scaling?

Managed Spot Training in Amazon SageMaker AI

AWS Machine Learning Blog

ML SOLUTION MONITORING, MAINTENANCE AND SECURITY

Monitoring ML Model Inference

Monitoring and Optimizing Infrastructure and Costs

Securing AWS Resources in ML Workflows

Best Practices for Monitoring, Maintenance, and Security of ML Solutions

A. Monitoring ML Model Inference

Chapter 4.1 Understanding Drift in ML Models

Data Drift

Data drift denotes alterations to the input data's distribution as time passes. External elements or changing data patterns cause these modifications. Once feature distributions, like age, location, or weather, change significantly, the model probably fails to generate good predictions. It trained on data from a past distribution, leading to a performance drop.

Example: Think of a machine learning model. This model forecasts customer churn. It uses features such as age and income. If the income spread among customers changes significantly over time, for instance, the model could have problems because of economic shifts. This is because the training happened on a different spread. Without detecting this drift, the model gives less reliable predictions.

Data drift has types.

- **Covariate shift** means the distribution of features changes, yet the connection between what you want to predict and the features stays constant.
- **Prior probability** shifts happen when the percentage of various categories alters in classification tasks; however, the basic relationship does not suffer damage.

Concept Drift

Concept drift happens when the link between input features and the target variable shifts across time. This makes more trouble than data drift does because the basic grasp for the model about how inputs relate to outputs turns worthless. The model may still detect structures within the data, yet these structures don't produce the right forecasts anymore.

For example, picture a machine learning model trained to predict a person's chance to buy a product according to their age plus prior buying history. During this period, consumer behavior alters - consider that younger people start to complete more luxury buying. The model even had training to forecast purchases primarily on former data, in which older people acted as main purchasers. Whenever the connection involving age plus buy chance altars across time, concept drift happens.

Types of Concept Drift:

- **Sudden Drift** - means a fast shift in how data connects. For example, this happens with new laws or market changes.
- **Incremental Drift** - describes a slow alteration through time. It can be hard to spot, yet it builds up.
- **Recurring Drift** - refers to past designs that appear anew. This happens in cycles or across seasons. You must consider it when you create models.

Impact of Drift on Model Performance

Data plus concept drift harm a model's ability to work well. If someone does not spot a fix drift, the model has trouble. It gives bad or wrong outputs. Because it depends on old data distributions or connections, it becomes old. If the model is not re-trained with new data, it makes prejudiced guesses.

Detecting and Addressing Drift

Detecting plus dealing with drift is vital to keep the model exact. Implementing drift detection strategies is crucial.

- For instance, constant observation of the model's performance occurs across time. This uses metrics like accuracy, precision, and the F1 score.
- The model receives periodic retraining with fresh data. It adjusts to current trends plus distributions.
- Anomaly detection locates data patterns, and these vary from the training set in a large way.

With services like Amazon SageMaker Clarify plus AWS CloudWatch, one can find data plus concept drift. These watch prediction accuracies also locate variations in the feature distribution across time.

Chapter 4.2 Techniques for Monitoring Data Quality and Model Performance

Data quality and model performance must be monitored well for machine learning models to provide accurate plus dependable outcomes. Models need ongoing observation when in production to find performance decline or data problems. Amazon SageMaker includes solid tools for continual monitoring. These tools let data experts and engineers follow model health and preserve reliability.

Using SageMaker Model Monitor to Continuously Monitor Model Predictions in Production

Amazon SageMaker Model Monitor exists as a completely managed service. It helps in the observation of machine learning model quality after deployment. This service permits up-to-date surveillance of data quality and model performance. It gives warnings plus comprehensive reports when problems occur.

Function of SageMaker Model Monitor:

- **Data quality monitoring:** The SageMaker Model Monitor constantly watches the data put into the model when it is used. It checks for shifts or odd features in that data. The system automatically sees how the statistical spread of current data differs from the spread used for training. If any feature's distribution shifts significantly (data drift), the Model Monitor will raise an alert.
- **Model Performance Monitoring: I**n addition to tracking data quality, SageMaker Model Monitor also monitors model predictions, comparing them to expected outputs or ground truth (if available). The performance is assessed based on accuracy metrics, such as precision, recall, or AUC.
- **Monitoring Alerts:** When there is an issue with the incoming data (e.g., a feature has deviated from the expected distribution), or the model's performance drops below a defined threshold, Model Monitor can automatically trigger alerts. These alerts can help data scientists or ML engineers investigate the cause of the degradation and take corrective action.

SageMaker Model Monitor has important abilities.

- It builds a basic standard using the data used to train the model. This standard concerns both the input information and the model's anticipated results. This standard assists with judging future data and predictions.
- It identifies data drift. If the nature of production input data differs from the training data, Model Monitor sends alerts.
- **Model Performance Metrics:** It can automatically calculate performance metrics such as accuracy or log loss and flag if performance falls below an acceptable level.
- **Automated Reporting:** Reports are automatically generated to summarize the model’s performance over time, which makes it easy to track degradation or issues in the model's behavior.

Example: A company uses a fraud detection model. This model determines if a financial transaction exhibits fraud. It does this based on several factors, like transaction amount, customer ID, and location. The company employs SageMaker Model Monitor. It can observe how the model functions in real time as fresh data comes

in. When the arrangement of features, such as transaction amount, alters, the system creates a warning. The team has a reason to practice the model again on the new data.

Detecting Anomalies in Input Data or Prediction Output Using Automated Tools

Spotting unusual data in inputs or forecasts is vital because this reveals possible issues in machine learning setups. Anomalies point to data problems, model changes, or outside influences on forecasts. Various methods plus automated software exist - they support quick anomaly spotting.

Techniques for Anomaly Detection in Input Data:

- **Statistic Methods:** In input data anomaly, finding benefits from certain methods. Simple math processes, like figuring out average and variation, show when a feature differs much from what it should be. When a fraud model sees a quick rise in typical spending, that may present a data anomaly.
- **Z-Scores**: another approach spots anomalies with numbers. It checks how many standard deviations a data point rests from the average once a Z-score goes past a limit, like three standard deviations, that counts as an anomaly.
- **Isolation Forests:** Isolation Forests is an algorithm. It functions properly on complex data. Instead of profiling normal data, the algorithm isolates anomalies, so it is effective for outlier detection in datasets. These datasets must have features that are plentiful.
- **SageMaker Model Monitor for Anomaly Detection:** Also, SageMaker Model Monitor works with Amazon CloudWatch. It automatically flags anomalies in input data. Through data statistics and machine learning algorithms, it detects outliers.

Anomaly detection in prediction output has several approaches

- **Error analysis:** It involves comparing the model's predictions with the real results, when available. This comparison lets you compute error metrics, such as MAE or MSE. The process searches for data points that are far from the average error. When a sudden increase in error occurs, it possibly suggests an issue with the model's forecast.
- **Confidence intervals can assist:** If a model produces probabilities, like in classification tasks, then anomaly detection examines the confidence scores. Should the confidence score seem quite low for a particular prediction, this likely points to some anomaly in either the input or within the model itself.

- **SageMaker Model Monitor:** It computes performance metrics automatically. It comes with tools for discovering when these metrics fall beneath a specified limit. That event denotes that the model has problems.

Example: A healthcare prediction model forecasts patient results. A rapid rise in incorrect forecasts by the model (for example, a "no disease" prediction for a patient with a confirmed disease) suggests a problem. Automated anomaly detection, by way of SageMaker Model Monitor, helps in marking such instances. The team examines if the input data has changed or if the model requires retraining.

Best Practices for Monitoring Data Quality and Model Performance:

1. **Determine the key metrics:** The organization must specify performance benchmarks, such as accuracy plus AUC, and establish limits for data distribution.
2. **Use automatic monitoring:** For example, SageMaker Model Monitor helps by automating the review process and spots problems as they happen. This automation reduces manual checking.
3. **Periodic Model Evaluation**: Models need assessments with new data on a regular basis. This happens even if the model appears fine. This process helps make certain the model stays useful plus correct.
4. **Retrain the Model**: If the system finds data or prediction issues, the model needs retraining with new data. Updated data improves the model's abilities.
5. **Establish alert systems:** Automatic alerts warn about model issues, data changes, or anomalies. This action guarantees quick responses when problems appear.

With tools like SageMaker Model Monitor, firms put anomaly detection in place. They follow standards because these organizations hold good models. These models react to shifts in data and the operational environment.

Using SageMaker Clarify for Data Analysis and Bias Detection

Amazon SageMaker Clarify is a function. It finds possible bias in machine learning models. It gives model explainability. This supports data scientists and ML engineers to grasp how their models form predictions. Clarify supports fairness across diverse demographic categories. Throughout the ML lifecycle, Clarify works. This starts from data preparation and extends to post-deployment monitoring.

Key Components and Features: Metrics used before training investigate data bias. Other metrics used after training, assess bias within the model's predictions. It works with structured data plus also with unstructured data. It contains bias metrics, for example Class Imbalance plus Difference in Proportions of Labels.

B. Monitoring and Optimizing Infrastructure and Costs

Chapter 4.3 Key Performance Metrics for ML Infrastructure

Tracking machine learning infrastructure performance proves vital to confirm the system manages the necessities of training, deployment, and large-scale inference. Because machine learning tasks get more complex and extensive, following vital metrics matters more. These metrics help infrastructure performance optimization, scalability, and dependability. This chapter describes utilization, throughput, scalability, fault tolerance, and availability. They represent essential machine learning infrastructure performance metrics.

Utilization: Measuring Resource Usage (CPU, GPU, Memory and Storage)

Utilization shows how well the infrastructure's resources are put to work. Tracking resource utilization helps confirm system performance without hardware overload or underuse.

Key Metrics for Resource Utilization

CPU Utilization: CPU utilization displays the system's CPU resource percentage in use. For ML tasks, especially big model training, high CPU utilization is normal. Constant high CPU usage hints that the system lacks optimization, causing flaws or blockages.

Formula:

$$CPU\ Utilization = \left(\frac{CPU\ Usage}{Total\ CPU\ Capacity}\right) \times 100$$

Best Practices: Look at CPU usage when there is high activity and when there is low activity. If usage stays high, you might expand the infrastructure or refine the model training. Doing so should enable full use of multi-core processing.

GPU Utilization: Because many machine learning operations use GPUs to speed up calculations, especially operations with deep learning models, watch GPU usage. This confirms that the machine learning model uses all available GPU power, preventing slowdowns.

Formula:

$$GPU\ Utilization = \left(\frac{GPU\ Usage}{Total\ GPU\ Capacity}\right) \times 100$$

Best Practices: Good practice involves task distribution across several GPUs so GPU resources are not idle. One needs to watch GPU memory usage to confirm that models do not go over the allocated amount.

Memory utilization: The memory utilization metric records the quantity of RAM the system uses. If memory is insufficient, system performance can drop. This is especially apparent when large datasets are processed or intricate models are trained.

Formula:

$$Memory\ Utilization = \left(\frac{Memory\ Usage}{Total\ Memory}\right) \times 100$$

Best Practices:

- Confirm that the infrastructure has adequate memory for big datasets. This is particularly important when you train complex models.
- Memory-efficient algorithms, for example, mini-batch gradient descent, must be used. They keep memory overload at bay during training.

Storage utilization: Gives a measure of how much disk space is in use within the entire system. Because ML models plus datasets often need notable storage capacity, this becomes important, especially when systems deal with huge training sets or save several model versions.

Formula:

$$Storage\ Utilization = \left(\frac{Used\ Storage}{Total\ Storage\ Capacity}\right) \times 100$$

Some helpful habits are to watch disk I/O and storage consumption to avoid data slowdowns. It is good to routinely remove old models, in-between training results, and logs so storage space becomes available.

Tools for Monitoring Resource Utilization:

- **AWS CloudWatch**: Used for monitoring CPU, GPU, memory, and storage usage on EC2 instances.
- **NVIDIA nvidia-smi**: For GPU utilization and memory usage monitoring.
- **Prometheus & Grafana**: For real-time monitoring and visualization of resource usage.

Throughput and Scalability: Ensuring Infrastructure Can Handle Growing Data Volumes

Throughput and Scalability – Data handling rate besides growth capacity are important. They make certain the infrastructure deals with data at needed speeds. They let it expand with increasing data amounts.

Key Metrics for Throughput and Scalability:

The throughput rate indicates system data processing speed. For ML tasks, the data-handling rate matters a lot in the training and prediction phases. When we observe the data handling rate, we see if the structure handles large data amounts in a reasonable amount of time.

Formula:

$$Storage\ Utilization = \left(\frac{Total\ Data\ Processed}{Time\ Taken}\right)$$

Best Practices:

- Track the throughput of data processing pipelines (e.g., data ingestion, ETL processes).
- Ensure the system can handle increasing throughput requirements as data grows by scaling resources appropriately (e.g., moving to larger EC2 instances or adding more instances).

Scalability:

Scalability shows the capacity of an infrastructure to manage a greater load through more resources. A machine learning infrastructure must scale in two ways: vertically, by using stronger machines, and horizontally, by using a greater number of machines. This accommodates growing data amounts.

Key Considerations: For example, vertical scalability boosts resources on one machine. CPU, memory, or GPU increase. Horizontal scalability, in contrast, adds more instances to spread tasks. Amazon SageMaker Distributed Training or EC2 Auto Scaling do this.

Best Practices:

Employ auto-scaling groups in training and inference. The system then adds or removes instances automatically based on current demand. You should pick distributed computing frameworks like Apache Spark or Dask. These tools process extensive datasets across different nodes.

Tools for Monitoring Throughput and Scalability: Consider these tools when you track throughput and scalability. Amazon CloudWatch helps oversee throughput besides establishing auto-scaling. Amazon EC2 Auto Scaling also serves to adjust instances based on demand. SageMaker Distributed Training helps in scaling machine learning workloads across many instances.

Fault Tolerance and Availability: Maintaining System Uptime and Preventing Failures

Keeping systems up and running while stopping them from failing falls under fault tolerance and availability. Ensuring the ML system stays up and running without trouble even when there's a glitch in the equipment, issues with the software, or sudden increases in demand hinges on how well it can handle mistakes and stay available.

Key Metrics for Fault Tolerance and Availability:

Availability: how well a system stays up and running over time. In environments where things are being made, especially those needing real-time inference systems, always-on is key.

$$Availability = \left(1 - \frac{Downtime}{Total\ Time}\right) \times 100$$

Best Practices: For important services, such as Amazon SageMaker or EC2, utilize multi-AZ deployment. This confirms high availability in different areas. Set up load balancing, like Elastic Load Balancer (ELB), to send requests to different servers.

Fault Tolerance: Fault tolerance means a system can still work if it has problems. ML systems need designs that recover by themselves from hardware or software issues so that model performance stays steady.

Key techniques:

- **Replication -** data and model checkpoints need copies across many servers or regions, which safeguards data.
- **Automatic failover** uses things like Amazon RDS Multi-AZ or Amazon S3 cross-region replication. These confirm automatic transition if a server stops working.

Best Practices:

- Proper methods include configured scheduled backups plus snapshots of models plus data. This makes certain data return after a fault.
- Health checks are needed to find and then change defective parts by automation.

For fault tolerance plus availability supervision, consider these aids

- Amazon CloudWatch Alarms watch system strength besides establishing automatic fix tasks.
- Because AWS Elastic Load Balancer (ELB) spreads traffic equally, it confirms good availability.
- Through DNS-based routing, Amazon Route 53 helps with failover.

Summary:

Keeping an eye on utilization, throughput next to scalability, along with fault tolerance plus availability proves essential to hold up the function plus dependability of ML infrastructure. Organizations use tools such as Amazon CloudWatch, SageMaker, EC2 Auto Scaling, and AWS Load Balancing. These tools guarantee that their ML systems work in an optimal fashion. As data demands increase, these systems grow. They recover from errors in a fast manner.

Upon grasping and observing these performance metrics, organizations make certain that their ML systems fulfill business needs. At the same time, they have elevated performance, dependability, and with efficiency.

Chapter 4.5 Using CloudTrail for Monitoring and Retraining

In a machine learning (ML) lifecycle, security, transparency, and reliability for a model's development, deployment, and maintenance are important. AWS CloudTrail has a large part in watching, checking next to securing compliance for ML workflows. It tracks model retraining plus access to ML resources. This chapter will describe how to employ CloudTrail to record actions linked to model retraining. It explains how CloudTrail helps check access to ML resources so that compliance stays in place.

Setting Up CloudTrail for ML Workflows:

AWS CloudTrail lets you watch and save information on what occurs in your AWS account when someone takes actions on your AWS setup. CloudTrail gets and saves API requests made to AWS services. This includes what happens in Amazon SageMaker, which is often utilized for model training, retraining, and deployment.

Creating CloudTrail Trails:

- When you set up CloudTrail for machine learning workflows, it logs all API calls in your AWS account. This includes what people do with Amazon SageMaker, for example, when they begin a training job or trigger model retraining.
- To begin, you should make a trail in CloudTrail. This trail will get all management events plus data events for services such as SageMaker besides EC2.

Steps to Set Up CloudTrail:

- Begin within the AWS Management Console and locate the CloudTrail service.
- Start a trail creation process plus assign a name.
- If you want coverage across all regions, activate the global tracking setting.

- Select an S3 bucket - it will hold the log data.
- With that chosen, activate CloudWatch Logs integration because this provides immediate monitoring plus alarms about retraining actions.

Tracking Retraining Activities:

- CloudTrail records crucial activities. For example, it documents when a new model training task begins or a model's update process starts. Each record has specifics, such as the time it occurred, the IP address it came from, the user involved next to the exact API function invoked, like CreateTrainingJob or StartRetraining.

- Imagine a data system that sets off a model update when it spots a shift. CloudTrail then saves the request to launch the retraining task inside Amazon SageMaker.

Key Actions to Track:

- To begin a training job, use either CreateTrainingJob or StartTrainingJob.
- When a model requires retraining, use UpdateModel or StartModelTraining. Custom API calls that activate retraining workflows represent another option.
- After training, set up the model deployment by using CreateEndpoint or UpdateEndpoint.

Example Use Case: The system retrains models when Amazon SageMaker finds data drift over a set limit. You can then use CloudTrail. It observes these events. Doing so permits auditing of the exact time also individuals who started the retraining. This confirms that only permitted changes happen. The entire retraining procedure receives a log. The log assists with transparency and problem fixes.

Leveraging CloudTrail for Auditing Access and Maintaining Compliance in ML Workflows

Following the rules in machine learning setups is very important. This is true especially if the task involves private details or certain areas, such as banking or health. CloudTrail has strong abilities for checking access and making sure every interaction with the machine learning setup is permitted and noted.

Auditing Access:

CloudTrail logs can help you audit who has accessed ML resources and what actions they have performed. This is important for ensuring that only authorized personnel or services can trigger retraining, access sensitive data, or deploy models.

Access Control Audits:

- CloudTrail offers detailed logs of the specific request that a certain IAM user or role did. It notes if the request succeeded or failed.
- For instance, should a user try to start a training job within Amazon SageMaker, the log includes the details of the user's IAM role. It records the API call and whether the action got permitted or denied.

Monitoring Unauthorized Access: CloudTrail spots improper access trials on Machine Learning services. When an attacker gains entry to a harmed Identity plus Access Management user profile, for example, along with aims to redo model training, the action becomes recorded. You establish CloudWatch Alarms so you get warnings about any questionable behaviors.

Role-based Access Control (RBAC) Compliance: With CloudTrail along with IAM policies, you can observe if users get access to approved resources. If an ML engineer must retrain a model, they should not deploy it to a production environment. CloudTrail logs document any action outside the permitted operations.

Maintaining Compliance: In regulated fields such as finance or healthcare, it is essential that model retraining plus deployment are auditable and follow industry standards like GDPR and HIPAA.

1. **Tracking All Changes**: CloudTrail notes each modification done to your ML models, from data treatment to model deployment. It provides a comprehensive audit track. It can be checked during compliance studies or external audits.
2. **Ensuring Data Privacy**: Should your models handle personal details or health data, CloudTrail lets you record who viewed the data plus at what time. With AWS resources like AWS KMS, you enforce data encryption plus assure CloudTrail notes such actions.

Example Use Case: The financial services firm must observe who starts model retraining on confidential financial information to adhere to Sarbanes-Oxley (SOX) rules. CloudTrail lets the firm hold a thorough record of each training task. It holds the IAM user who began the retraining and the exact data reached. Because of this audit trail, all actions are trackable. It grants openness and safety.

3. CloudTrail Best Practices for Monitoring and Compliance in ML Workflows:

- **Activate multi-region trails**: This makes CloudTrail record actions throughout all sections of your AWS account. Comprehensive surveillance is important, particularly if machine learning resources exist in varied areas.

- **Employ CloudWatch Logs integration.** By this action, real-time log handling is possible, and alerts for suspect or disallowed behaviors get defined. One example involves unauthorized model retraining.

- **Establish strong IAM policies. A**dopt least privilege access standards, then specify detailed IAM roles. These actions control who starts retraining or views private information.
- **Set Retention Policies.** For data retention plus auditing access, establish suitable log-holding regulations. This fulfills the legal requests.

Conclusion:

AWS CloudTrail is useful to watch actions connected to model retraining and to check entries within machine learning processes. It makes your machine learning systems safer and more compliant. All deeds get noted, approval happens along with clarity remains. Then you hold power over your machine learning models and related items. You can link CloudTrail to more AWS tools. For example, IAM, CloudWatch next to SageMaker exist. This lets you build a tough plan to watch and also keep up your machine learning space.

Chapter 4.6 Instance Types and Their Impact on Performance

Choosing the correct instance type is crucial for machine learning models when you deploy them because it ensures the best performance, good value, and growth potential. Amazon Web Services has many instance types available, and each type is built to manage particular tasks well. Depending on memory, processing power, and inference functionality, those instance types fall into categories. In this section, we look at different instance types within Amazon Web Services, plus we examine how those affect model operation.

Memory-Optimized Instances: For Models with Large Memory Requirements

Memory-optimized instances suit workloads - these workloads demand plentiful memory for smooth performance. They are appropriate for data-heavy models when models handle huge datasets, execute complex data operations, or preserve big in-memory models.

Key Features:

Significant memory-to-CPU ratio exists: These instances present big memory configurations. These setups are crucial to execute models, so models need high memory. Deep learning models and major data handling, along with analytics workloads, are examples. Large models gain a benefit from these instances. These models involve extensive feature sets or big batch sizes and need high memory.

Use Cases:

- NLP models like BERT or GPT need abundant memory - they keep plus deal with substantial language datasets.
- Substantial data preprocessing uses much memory to control large data in a distributed method.

Example Instance Types:

- **R5 instance** within the R family gives superior memory capacity. As a result, it matches demanding databases, significant data analytics, and in-memory caches.
- **X1e instance** is ideal for memory-bound applications like HPC or in-memory databases.

Best Practices:

- For model needs of plenty of memory, you can select memory-optimized instances, so the model deals with or keeps intermediate results.
- But watch memory consumption closely since capacity overage degrades the instance's performance.

Compute-Optimized Instances: For CPU-Intensive Workloads

Compute-optimized instances have great processing power. They are good for tasks that use a lot of the CPU. These instances offer high CPU performance. They are optimized for heavy compute tasks like model training plus big simulations.

Key Features:

- **High CPU-to-memory ratio:** These instances are designed for tasks that require substantial computing power but do not necessarily need large amounts of memory.
- **For parallel processing, they work efficiently.** For example, they suit tasks that gain from multi-core processing, like training machine learning models or running simulations. Such simulations require many calculations each second.

Use Cases:

- Situations appear where models contain tricky algorithms. Some models use many CPU cores to handle much data simultaneously. The training of such models fits this requirement.

- For simpler models, data changes or model training might cause a delay because of the calculation. Such models encompass linear regression plus decision trees.

- With real-time analytics or scientific computing, the workload demands lots of calculation power but less memory.

Example Instance Types:

- Consider C5 instances. These instances, like c5.xlarge or c5.2xlarge, have good use for calculation-heavy ML tasks. For instance, you can train decision trees or handle big data with them.
- C6g instances use ARM architecture and, because of that, these instances deliver top performance for tasks that depend on the CPU. They cost less than typical instances built on x86.

Best Practices:

- It is good to choose compute-optimized instances if the model training relies a lot on the CPU and does not need huge memory.

- Code optimization for parallelism, like multi-threading, is good because it uses the full computing ability of these instances.

Inference-Optimized Instances: For Faster Prediction and Model Inference

Inference-optimized instances provide fast, efficient predictions or model inference. This design supports serving machine learning models during production. It also permits the delivery of real-time predictions.

Key Features:

- **Optimized for latency and throughput**: These instances handle many requests quickly. Because of this, they are well-suited for real-time inference.
- **Scalable to high traffic**: Inference-optimized instances manage many simultaneous prediction requests. They work well in production environments. So, they address the needs for numerous predictions.

Use Cases:

- Applications involve real-time suggestion systems. These systems need quick reactions to deliver custom content or propose items.
- Computer vision uses this. Image sort or item find need a speedy, effective process for users to feel good.
- For chatbots or digital helpers, short inference delays matter a lot. This sustains normal talk rhythm.

Example Instance Types:

- **Inf1:** These instances have a special design for deep learning inference duties. AWS Inferentia chips power them, and they offer economical, powerful inference.
- **G4ad:** These instances feature optimizations for inference duties based on GPUs. They are very useful for machine learning models that need substantial throughput and for multimedia data processing in real time.

Best Practices:

- Use specialized instances to serve prediction requests because that improves performance, mostly when response time is important.
- Your models must be refined so they perform well. This includes converting them to efficient formats like TensorFlow Lite or ONNX. This boosts performance on hardware built for prediction.

Summary:

- For ML tasks, especially those tied to memory needs, memory-focused instances become crucial. Examples include massive data work plus complex model training that demands substantial memory.
- Compute-focused instances do very well with tasks limited by CPU power, for instance, training algorithms that involve complex math. They are also useful to manage large simulations.
- Designed specifically for prediction tasks, inference-focused instances handle high volumes of requests with minimal delay. This ensures quick plus adjustable real-time prediction for deployed models.

Through proper instance selection, you improve model performance, cut expenses, and grow your ML applications. The selection requires a good understanding of the workload.

Chapter 4.7 Tools for Infrastructure Monitoring

Good infrastructure monitoring matters to confirm that machine learning (ML) models operate well in production. AWS provides various tools that help you to follow the health, performance, along with cost-effectiveness of your ML infrastructure. Within this chapter, we will see useful monitoring tools. They include CloudWatch Dashboards, EventBridge, SageMaker Inference Recommender, and AWS Compute Optimizer. These tools ensure that your ML systems remain responsive, optimized, and cost-efficient.

CloudWatch Dashboards: Building Custom Dashboards to Track Performance Metrics

Amazon CloudWatch, a monitoring tool from AWS, presents features that help you follow the condition and output of your infrastructure. It contains custom dashboards for display. With CloudWatch Dashboards, you can show KPIs, follow resource use, and watch system metrics right away in one place.

Important features of CloudWatch Dashboards

- CloudWatch lets you build personal dashboards that join metrics from various AWS services, for example, Amazon EC2, SageMaker, along CloudTrail. But you adjust these dashboards to center on important metrics for your ML workflows. Examples of these metrics are CPU usage, GPU utilization, memory usage, and model inference latency.
- You can watch resource use and performance at the present time with CloudWatch Dashboards. You make sure that people see any problems right away.

Example Use Case:

For example, a data scientist makes a specific CloudWatch Dashboard. It watches the GPU usage and memory consumption as an Amazon SageMaker model trains. When a metric passes a certain level, an alarm goes off. This alarm asks for adjustments to resource use or scaling of the infrastructure.

Best Practices:

- To get the most out of it, create **CloudWatch Alarms**. They notify you once crucial metrics, like memory or CPU use, go over defined values.
- Employ **CloudWatch Logs** to record detailed logs. These logs are useful to fix bad performance or failures.

EventBridge: Automating Responses to Infrastructure Changes and Performance Anomalies

Amazon EventBridge exists as a serverless event bus service. This service simplifies connections between applications with events. You can use it to automatically react to infrastructure changes and unusual performance within your ML systems.

Key Features of EventBridge:

With EventBridge, you record shifts in your infrastructure, such as SageMaker model retraining or resource scaling. It starts automated workflows or sends notifications as a response. EventBridge has a central quality: it offers an event-driven architecture. This architecture makes automatic responses to infrastructure changes

possible. For example, a new instance launches or a SageMaker model weakens in performance. EventBridge begins workflows to grow resources or start model retraining.

Rules for custom events are configurable. These rules establish responses to distinct event types. You tailor these rules to respond to triggers. For instance, it can be a fast increase in data processing demand.

EventBridge works with other AWS services. Step Functions next to SNS are a few examples. Because of this, tasks run on their own when events appear.

Example Use Case: To illustrate, consider machine learning deployment. EventBridge configuration leads to automated model retraining. This happens after it spots performance drift or input data changes. This activity starts a SageMaker retraining job. It makes certain the model stays exact.

Best Practices:

- For better results, automate routine infrastructure management with EventBridge. Backups or model updates are tasks to consider.
- With EventBridge plus AWS Lambda together, custom scripts or tasks operate. This happens because of particular infrastructure events. A model version update following retraining is one scenario.

SageMaker Inference Recommender: Optimizing Instance Selection for Inference Workloads

Key Features of SageMaker Inference Recommender:

- For cost-effective instance choices, the Recommender checks various instance types and gives advice on the ones that perform well and also keeps costs down. It does performance tests on different types. You find the right setup for your machine learning inference work.
- The service scales inference work besides giving advice about the amount of instances to manage production traffic. It ensures high availability and quick responses for real-time apps to occur.
- SageMaker Inference Recommender uses actual model data to test several instance types. It does that to decide what suits your specific needs best.

Example Use Case:

An e-commerce firm that uses a recommendation engine needs to determine the best EC2 instance type. The Inference Recommender helps them to find the option with a favorable cost. It also considers performance for product recommendations.

Best Practices:

- To assess your model's resource requirements, utilize Inference Recommender. It helps you select a good instance type for inference tasks.
- For changes in workload or new AWS instance types, review the suggestions often. By doing this you make sure the instance choice is still correct.

AWS Compute Optimizer: Recommending Appropriate Instance Types for Cost-Effective and Performant ML Workloads

AWS Compute Optimizer helps in the enhancement of compute resources. It does this by advising suitable EC2 instance types for machine learning tasks, using real usage information. Because the tool offers proposals for correct instance dimensions, it supports cost reduction, making sure resources are not over-supplied.

Key Features of AWS Compute Optimizer:.

- For example, Compute Optimizer analyzes past usage information plus advises EC2 instance types. These provide a strong mix of cost and output.
- It gives suggestions for CPU and memory improvements. You can pick the right EC2 instance for model training and inference tasks.
- Compute Optimizer spots underused instances, which can get smaller and it suggests scaling choices for a general system output boost.

Example Use Case:

A data scientist works with EC2 instances during model training through Amazon SageMaker. They can use Compute Optimizer to locate cheap instances for large models, for example, deep neural networks (DNNs) or transformers. With this tool, they prevent excessive resources plus cut down on expenses.

Best Practices:

- For ideal results, Compute Optimizer should be a regular element to evaluate instance use. This confirms that infrastructure remains affordable and well-performing.
- To manage training and inference instance optimization, you can integrate Compute Optimizer with SageMaker Inference Recommender.

Summary:

- CloudWatch Dashboards provide custom dashboards. These track key ML infrastructure metrics plus display system health.
- EventBridge automates infrastructure change plus anomaly responses. It does this when it starts workflows because of certain events.
- SageMaker Inference Recommender improves instance choice. This action assures cost-effective and powerful model inference workloads.
- AWS Compute Optimizer locates the ideal EC2 instance types. This helps you use resources in a useful and economic manner.

Because of these tools, you can maintain, improve, and grow your ML infrastructure. At the same time, they secure output besides economic operation.

Chapter 4.7 Tools for Infrastructure Monitoring

Reliable infrastructure monitoring remains vital for the proper execution of machine learning (ML) models in a production environment. AWS delivers several tools. They help users monitor the health, performance, and cost-efficiency of their machine learning infrastructure. In this chapter, we look at some helpful monitoring tools. These include CloudWatch Dashboards, EventBridge, SageMaker Inference Recommender, and AWS Compute Optimizer. With them, your ML systems stay responsive, optimized, and cost-efficient as time passes.

CloudWatch Dashboards: Building Custom Dashboards to Track Performance Metrics

Amazon CloudWatch is AWS's core monitoring tool. It contains features that help users in following the condition besides the function of infrastructure plus custom dashboards for clear visual information. Through CloudWatch Dashboards, a person can show key performance indicators (KPIs), follow resource use, and watch system metrics immediately from one area.

Key Features of CloudWatch Dashboards:

- **CloudWatch Dashboards:** CloudWatch permits you to make individual dashboards. These bring together metrics taken from varied AWS services, such as Amazon EC2, SageMaker, along CloudTrail. A person can adjust these dashboards to watch the most useful metrics for their ML workflows, like CPU use, GPU function, memory use, as well as how long a model inference took.

- **Real-Time Monitoring**: With CloudWatch Dashboards, you can immediately check resource use and operation. This confirms that any unusual behaviors receive immediate attention.
- **Alerts & Alarms**: Thresholds are configurable for metrics, plus notifications via Amazon SNS or email happen when those limits break.

Use Case: For example, a data scientist builds a personalized CloudWatch Dashboard to watch GPU use and model inference latency on Amazon SageMaker endpoints. Should those metrics pass predefined limits, CloudWatch causes alarms so someone deals with it quickly.

EventBridge: Automating Responses to Infrastructure Changes and Performance Anomalies

Amazon EventBridge is a serverless event bus. It links application data from your own programs, built-in SaaS programs, and AWS services. It permits event-driven reactions to changes within your AWS setup.

- Concerning automated workflows, you can set up rules. These rules listen for specific events, such as new instance launches or changes in SageMaker endpoints. The rules start workflows as a response.
- Through integration, EventBridge connects with AWS Lambda, Amazon Step Functions, and other services. With this, EventBridge runs tailored logic when an event takes place.
- When CloudWatch spots a performance anomaly in your ML base, EventBridge starts corrective actions on its own. For example, it can scale or restart resources.

SageMaker Inference Recommender: Optimizing Instance Selection for Inference Workloads

SageMaker Inference Recommender is built to assist users in picking suitable instance types when hosting their machine learning models in a live environment.

- The recommender tests models with different instance setups and finds the most suitable option for specific workloads.
- For example, users define acceptable latency and throughput levels, and the Inference Recommender suggests inexpensive options.
- This service executes performance tests plus creates a report. This report specifies cost and latency next to throughput for each suggested instance type.

Sample: A retail business utilizes Inference Recommender. The tool determines if ml.c5.xlarge, ml.g4dn.xlarge, or ml.inf1.xlarge is best. It considers the model for product suggestions. Cost plus actual speed are balanced in this choice.

AWS Compute Optimizer: Recommending Appropriate Instance Types for Cost-Effective and Performant ML Workloads

AWS Compute Optimizer examines how you use resources and then advises you to change instance sizes for greater effectiveness and lower expenses.

- For the historical utilization analysis, Compute Optimizer looks at CPU, memory, and network next to storage data to check if instances have too little or too much workload.
- It gives other EC2 instance type options that deliver improved performance or cost less, depending on your tasks.
- Because it consistently studies usage, Compute Optimizer helps you in maintaining a current infrastructure that matches evolving needs.

Use Case: As an example, if a compute-optimized instance runs at 30 % CPU use, Compute Optimizer could advise a reduced instance family, such as c5.large in place of c5.xlarge, so expenses go down but performance stays the same.

Summary:

CloudWatch Dashboards present current, flexible views of your machine learning infrastructure data. EventBridge starts automatic replies to abnormalities or updates to the system.

- SageMaker Inference Recommender helps in the choice of correct instance kinds made for machine learning inference jobs.
- AWS Compute Optimizer gives continuous advice to match computing power with need - it confirms efficient cost and strong work.

With these tools used together, machine learning systems stay adjustable, have a good cost, and withstand changes in demand.

C. Securing AWS Resources in ML Workflows

Chapter 4.8 IAM Roles, Policies, and Groups for AWS Access

Strong identity plus access management is essential for AWS resource security plus compliance within a machine learning environment. AWS Identity plus Access Management lets you manage access for AWS services and resources. Within this section, we examine how to set up IAM roles and policies so they give needed permissions. We show how to make IAM groups for streamlined access administration for several users on ML systems.

Configuring IAM Roles and Policies to Grant Appropriate Access to AWS Services

IAM roles and policies function as the core of access management inside AWS. A role represents an item with a specified collection of authorizations to create AWS service requests and policies for documents, in JSON, which detail accepted or rejected resource actions.

IAM Roles

Function: IAM roles permit delegation of access to AWS services or resources, so long-term credentials, like access keys, do not need sharing. Because of its assumption, a role is adoptable by particular items, such as IAM users from within the same or another AWS account or AWS services, for example Amazon EC2, AWS Lambda along with Amazon SageMaker plus federated users through web identity or SAML federation.

For example an Amazon SageMaker notebook instance is capable of adoption of an IAM role. That role gives it rights to read data out of an S3 bucket. The notebook's base compute resources need no enduring access credentials.

IAM Policies

An IAM policy constitutes a JSON document which details permissions (or denials) for an entity, such as a user, group or role.

Types: AWS managed policies exist, pre-created by AWS, suitable for usual situations. Examples involve AmazonS3ReadOnlyAccess or AmazonSageMakerFullAccess. Customer managed policies are custom and you make them to satisfy particular security needs. Inline policies reside directly within a user, group or role entity, generally whenever individual permissions pertain to only a single entity.

Least Privilege Principle: Concerning the least privilege principle, grant only the smallest set of permissions which a task demands. Audit plus amend policies often to eliminate stale or abundant permissions.

Example Policy: Granting an IAM role read/write access to a specific S3 bucket used for storing ML training data:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::my-ml-data-bucket/*"
    }
  ]
}
```

By attaching this policy to a role assumed by an Amazon SageMaker notebook, you ensure that the notebook can only read from and write to `my-ml-data-bucket`—nowhere else.

Creating IAM Groups to Manage Access for Multiple Users Working with ML Systems

An IAM group contains multiple IAM users. When you link a policy to it, the policy impacts every user inside. This action streamlines permission administration, particularly in collaborative ML settings.

Why should you use groups?

Efficient Management: you do not have to assign policies to each user. A group like DataScientists or ML-DevOps receives the needed policies once.

Scalability Improves: When someone new joins, you just put them into the correct group. This gives them standard permissions. In addition groups permit role-based access control.

Role-based Access Control (RBAC): For instance you can align groups with roles such as ML_Engineers, who get complete Amazon SageMaker access but only can read from S3. Further examples are ML_Admins, who

receive full administrative rights for ML tasks plus Analysts, who only can read ML logs, metrics along with data.

Stages to Build an IAM Group

- Access the IAM Console.
- Go to Groups, then choose Create New Group.
- Assign a name to the group, such as SageMaker-Users.
- Link proper policies, for example, AmazonSageMakerFullAccess, AmazonS3ReadOnlyAccess.
- Put needed users into the group.

Example Use Case:

A business owns five data experts. They want access for reading and writing to information inside an S3 bucket. They use this access for feature work and training tasks inside Amazon SageMaker. When the admin builds a group named DataScientists plus connects the wanted policies, they then add each data expert to this group. This avoids manually assigning the same policies more than once.

Summary:

- IAM Roles work well when you assign permissions to AWS services, such as SageMaker, EC2 or Lambda. They are also useful for external users, because credentials do not need to be shared.
- IAM Policies are permission documents written in JSON format. These documents state the actions permitted or restricted on certain resources.
- IAM Groups ease the task of managing several users. They accomplish this by assembling the users plus assigning them shared policies.

By following IAM best practices, your secure, scalable environment for ML workflows stays in order. For example use minimal privilege, conduct regular auditing and employ managed policies.

Chapter 4.9 SageMaker Security and Compliance Features

Security plus compliance represent essential parts of each machine learning (ML) project. Especially important, this becomes true when projects handle sensitive or regulated data. Amazon SageMaker includes several built-in features. These features help in maintaining confidentiality, integrity along with compliance all through your ML workflow. This section checks two main components of SageMaker's security model. It covers data encryption. It explores the SageMaker Role Manager to control resource access.

Ensuring Data Encryption Within SageMaker AI

Data encryption is a fundamental aspect of securing sensitive information, especially in cloud-based environments like Amazon SageMaker AI. Encryption helps safeguard data both at rest—when it is stored on disk—and in transit—when it is being transferred between systems. Within SageMaker AI, AWS provides robust built-in mechanisms to ensure that data remains protected throughout its lifecycle.

Encryption at Rest

Amazon S3: You can turn on encryption when data is at rest, through a few methods.

- One way is Server-Side Encryption with Amazon S3-Managed Keys.
- Another is Server-Side Encryption with AWS KMS keys (SSE-KMS).
- For certain situations needing outside key control, Client-Side Encryption is an option.

EBS Volumes: For SageMaker notebook examples, training tasks or conclusion points needing Amazon EC2 and linked EBS volumes, EBS encryption is available. In this way all data saved to disk receives protection. AES-256 or keys handled in AWS KMS accomplish this.

AWS KMS (Key Management Service): KMS lets you oversee personalized keys, giving exact entry rules. This becomes useful for closely watched fields where key change and key use log happen often and are mandatory.

Best Practices:

- Use SSE-KMS because it makes audit logs stronger. It gives you power over key use via IAM policies.
- **Grant least privilege:** For KMS keys, only allow access to the minimum needed. This makes certain only required roles or users are able to use the keys to decrypt or encrypt.

Encryption in Transit

- **HTTPS Endpoints:** SageMaker talks to Amazon S3 through TLS/SSL. This confirms the secrecy of your information as it moves across the web.
- **Amazon SageMaker Endpoints:** Models located after a SageMaker endpoint utilize HTTPS for inference requests. This defends data as it goes from user software to SageMaker.
- **VPC Endpoints:** With additional safety, users locate SageMaker resources in a personal subnet. Use VPC endpoints (interface or gateway endpoints) for a reliable, private link to services like Amazon S3.

Best Practices:

- To avoid eavesdropping or man-in-the-middle attacks, you must enforce TLS/SSL on every connection.
- Utilize AWS PrivateLink with VPC endpoints if you want to keep traffic inside the AWS internal network.

Using SageMaker Role Manager to Control Access to SageMaker Resources

SageMaker Role Manager helps in simple creation and oversight of the IAM roles. SageMaker employs these roles. This process assists you to manage access to notebooks, training jobs, inference endpoints plus additional SageMaker tasks. Access management depends on user positions plus duties.

Role-Based Access Control in SageMaker

- **SageMaker Execution Roles:** A SageMaker resource commonly uses an IAM role. The role holds permissions to reach S3 buckets, for training data, logs or model artifacts. For example a notebook instance or a training job can access these roles.
- **SageMaker Role Manager:** It gives a wizard-like interface. Through this interface, you easily create roles showing tailored permissions. You define which S3 buckets someone can reach and what level of access (read, write or both) someone has. The system asks if the role requires permission to produce or oversee endpoints. The system specifies which VPC configurations someone permits for safe training or inference.

How SageMaker Role Manager Helps

- **Reduced Complexity:** This tool simplifies complexity because you choose from established configurations that follow ideal methods instead of creating unique IAM policies yourself.
- **Least Privilege Enforcement:** With the interface, enforcement of minimal privilege happens. It prompts you to provide only the permissions necessary for a specific job.
- **Easier Auditing:** For roles produced by SageMaker Role Manager, audits become simpler. These roles adhere to uniform naming standards and policies with structure.

Example Use Case:

Data scientists often access several S3 buckets. They create training jobs, hosting endpoints next to SageMaker experiments. Data analysts however, sometimes need read-only access to logs and performance metrics. With SageMaker Role Manager, you establish roles like SageMaker-DataScientist-Role or

SageMaker-Analyst-Role. Because of this the roles match needs besides give both security besides ease of use.

Summary

Encryption practices involve a couple of areas.

- To safeguard data at rest, use SSE-KMS or SSE-S3 for storage. Employ EBS encryption for notebooks or training volumes.
- You can also manage encryption keys via AWS KMS.
- As for protecting data in transit, utilize HTTPS endpoints, TLS/SSL connections and VPC endpoints to support data privacy over networks.

SageMaker Role Manager:

- SageMaker Role Manager makes it simple to form SageMaker-specific IAM roles. This confirms least privilege.
- It helps in preserving a uniform, safe method to give access to ML resources.

Due to the combination of strong encryption measures with role-based access control with SageMaker Role Manager, one has robust security plus compliance for their ML applications on AWS.

Chapter 4.10 Network Access Controls

Network access controls matter to secure machine learning infrastructure. You keep sensitive data and ML systems safe because you isolate resources, control data flow next to prevent unauthorized entry. In this chapter we check how to configure Virtual Private Clouds, subnets next to security groups for improved security. This chapter explains how to control internet access and private IPs for secure interactions with your ML infrastructure.

Configuring VPCs, Subnets, and Security Groups to Isolate ML Systems from Unauthorized Access

Virtual Private Cloud (VPC)

AWS Virtual Private Cloud (VPC) helps you set up a separate network inside AWS - you then locate your machine learning (ML) structure there. Right setup of the network makes sure your ML setups stay safe from unknown users or outside networks.

With VPC, you make a network just for you inside AWS - you set the IP range, subnets, route tables along with gateways. When you put ML systems in a VPC, you get control of access.

VPC Benefits:

- It gives a safe, separate space for ML systems.
- It permits private talk between spots inside the VPC, so they don't show up on the regular internet.
- Integration becomes possible with VPNs, Direct Connect or AWS Transit Gateway, which provides hybrid cloud ways.

Subnets

A subnet represents a piece of a Virtual Private Cloud Internet Protocol address range. You divide your Machine Learning resources into subnets. This division considers their security needs and traffic direction.

- **Private subnets:** Hold resources shielded from direct internet connections. They suit database instances, model training instances or backend services.

- **Public subnets:** For resources such as SageMaker endpoints or load balancers that demand internet access, public subnets work. Access remains regulated through security groups and network Access Control Lists.

Security Groups

A security group is a virtual firewall for EC2 instances and resources like Amazon SageMaker. It manages traffic entering and leaving. Because security groups are stateful, a permit for traffic to enter an instance also opens a way for related traffic to exit.

Best Practices:

- Effective methods involve setting rules. Rules give access only from reliable sources like specific IP ranges, VPCs or security groups. Examine plus adjust security group rules frequently. You maintain minimal exposure this way.

Example: For example a data scientist's Amazon SageMaker notebook instance could exist in a private subnet. The instance accepts incoming traffic only from other resources in the VPC. No traffic is accepted from the internet.

Network Access Control list (NACLs)

Network Access Control Lists (NACLs) add a security layer at the subnet. These lists work as simple filters for traffic that enters and leaves. Although security groups function at the instance, NACLs permit overall access control at the subnet.

- Because NACLs are stateless, different from security groups, meaning you must define rules for both traffic directions (inbound and outbound)
- For enhanced security, you should use NACLs to limit traffic. This confirms that networks with authorization can communicate with your ML infrastructure.

Controlling Internet Access and Private IPs for Secure Interactions with ML Infrastructure

To handle a production Machine Learning environment, a user can increase system security and privacy by managing internet access plus applying private IPs. The user restricts public access to resources and also communication between services occurs through secure channels.

Controlling Internet Access

Internet Gateway (IGW): To permit internet access, connect an Internet Gateway (IGW) to a VPC. Route tables govern which subnets gain this access.

Private Subnet: A private subnet lacks immediate internet access. Utilize a NAT Gateway/Instance when instances in a private subnet require outgoing internet access. This might be useful, for instance, when software needs updates.

Public Subnet: In contrast, instances inside a public subnet need immediate internet access and typically possess a public IP address. A SageMaker endpoint is an example.

VPC Endpoints: For safe dedicated access to AWS services such as Amazon S3, SageMaker along with DynamoDB, employ VPC endpoints. This arrangement confirms that data remains within the AWS private network and prevents exposure on the public internet.

Best Practice: As a rule of thumb, adopt VPC endpoints for services like S3 and DynamoDB. When doing so you avoid routing traffic across the internet, which improves security and also lowers data transfer costs.

Private IPs for Secure Interactions

Private IPs: By assigning private IP addresses to resources in your VPC, you ensure that internal communications between resources (like EC2 instances, RDS, or SageMaker models) stay within the private network and are not exposed to the internet.

Internal Load Balancers: If you use an internal load balancer in your VPC (for scaling SageMaker endpoints or other services), the communication happens over private IPs, adding an extra layer of security by restricting traffic to the internal network.

PrivateLink: For services that need to be accessed privately (from one VPC to another), AWS PrivateLink can be used to expose services like SageMaker endpoints and S3 over private IPs, bypassing the public internet.

Secure Communications

SSL/TLS Encryption: It is vital to secure machine learning endpoint communications with SSL/TLS encryption. As an example you must set up a SageMaker model endpoint with HTTPS when you make it available for predictions. This way every prediction request and any data it returns, stays encrypted.

For VPC communication, **VPC peering** establishes secure connections between different AWS accounts or regions. It is a private way to connect them.

Summary

- **Virtual Private Clouds:** subnets next to security groups are key. You set up a VPC for isolation. For security you locate machine learning resources in private subnets. You manage traffic besides from your instances with security groups.
- **Controlling Internet Access**: For controlled outbound internet access from private subnets, you use NAT Gateways. You establish VPC endpoints for safe, private access to AWS services like S3 or SageMaker.
- **Private IPs and Secure Communications**: Private IPs guarantee internal communications stay inside your VPC. SSL/TLS encryption secures data during transfer.

Through precise configuration of network access controls, you confirm your machine learning infrastructure stays safe. It also remains separate from unauthorized access. At the same time this setup permits efficient communication among internal resources.

D. Best Practices for Monitoring, Maintenance, and Security of ML Solutions

Chapter 4.11 Proactive Monitoring and Optimization

Proactive monitoring and optimization hold importance in machine learning workflows because they help sustain good health, proper performance along with efficient costs for models and infrastructure. When you establish suitable alerts and metrics for model performance and infrastructure behavior, you reduce risks such as model degradation, overfitting, underfitting along with concept drift. This section includes helpful practices to establish alerts, check infrastructure performance and refine ML models so long-term success occurs.

Setting Up Alerts for Model Degradation, Infrastructure Performance, and Cost Anomalies

Model Degradation Alerts

Model performance weakens as time passes because the basic data changes. This process has names like data drift or concept drift. An early discovery of this problem is important if you want to keep the model precise.

Amazon SageMaker Model Monitor helps in tracing how well a model works and what input data drift appears. You configure alerts to notify you when major differences from the basic performance appear or when the input data distribution changes.

Common Alerts

- **Accuracy Drop:** It warns you if the model's accuracy goes under a set limit.
- **Latency Increase:** It alerts you if the time for inference requests goes over a certain limit.
- **Precision/Recall Metrics**: Keep watch for major drops in precision or recall. These could show a shift in data or model problems.

Best Practice: As a best practice, build real-time alerts for vital performance metrics (e.g., F1 score, AUC) using CloudWatch Alarms. These alerts inform you when these metrics go under acceptable levels. And combine this setup with SNS for proper notification.

Cost Anomaly Alerts

Cost optimization matters for efficient scaling of machine learning workloads, particularly as model complexity rises and infrastructure expands.

- **AWS Cost Explorer plus AWS Budgets** let you monitor AWS utilization. They permit configuration of alerts for unforeseen cost rises.
- **Typical Alerts** For cost overruns alarms get set if machine learning infrastructure expenses surpass a defined budget.

In resource overprovisioning, alerts inform users once underused resources create needless costs, such as large EC2 instances for low-demand inference workloads.

Best Practice: A good approach involves configuration of AWS Budgets for monthly cost limits. Integrate it with SNS to gain notifications when machine learning infrastructure goes over the set budget.

Establishing Proactive Measures to Prevent Overfitting, Underfitting, or Concept Drift in Models

Monitoring matters but actions that stop overfitting, underfitting along with concept drift before they begin are vital. Below are good habits that avert these problems.

Preventing Overfitting

Overfitting happens when a model gets too involved. In this situation it learns the sound or odd jumps in the data used to train it. Because of this it does not do well with new data.

- **Cross-Validation:** For example one can use k-fold cross-validation. It checks how well a model works across parts of data and also lowers the chance of overfitting.
- **Regularization:** Techniques like L2 regularization (Ridge) or L1 regularization (Lasso) punish big values and push for simpler models. During the training process, validation performance receives assessment.
- **Early Stopping:** One can stop training when the model starts to degrade. This prevents overfitting the training data.

Best Practice: Use Amazon SageMaker Hyperparameter Tuning as it adjusts hyperparameters, for example, regularization strength plus dropout rates. This prevents potential overfitting.

Preventing Concept Drift

Concept drift means the link between the properties and the predicted outcome shifts with time. This makes the model work worse. External things like market shifts or how users act, cause this.

- **Continuous Monitoring:** To handle concept drift, continuous monitoring is important. Track changes in data besides how well the model works with SageMaker Model Monitor. Create alerts that warn about substantial drift.

- **Retraining Strategy:** Apply a retraining plan that renews the model on current data at specific times. Employ Amazon SageMaker Pipelines to handle this automatically.
- **Data Augmentation:** For better resistance to shifts, apply methods like SMOTE or data augmentation. These tactics evolve the pattern recognition of the model.

Best Practice: As a good idea, configure SageMaker Model Monitor to constantly match production data with training data. Trigger a retrain when the system finds concept drift.

Summary

Model health warnings: Watch important measurements. These are accuracy, precision next to the time needed for predictions. CloudWatch and SageMaker Model Monitor help you to build current notifications.

Infrastructure performance warnings: CloudWatch tracks CPU use, memory consumption as well as network traffic. Alarms help to automatically grow or refine resources.

About cost abnormality warnings: AWS Cost Explorer plus Budgets trace your ML system costs. Adjust warnings for inflated costs or wastage.

Preventing Overfitting and Underfitting: Since you want to stop the model from memorizing the data or being too simple, use Cross-validation, regularization as well as hyperparameter changes to assist in balancing model intricacy and training duration.

To halt concept shift: SageMaker Model Monitor traces data also idea movement. A training pipeline supplies ongoing model betterment.

Effective monitoring of models, infrastructure along with expenses happens when you use established methods for model optimization. This way you confirm that your machine learning processes are strong, affordable along with adjusting to changes in time. It safeguards that it will work successfully.

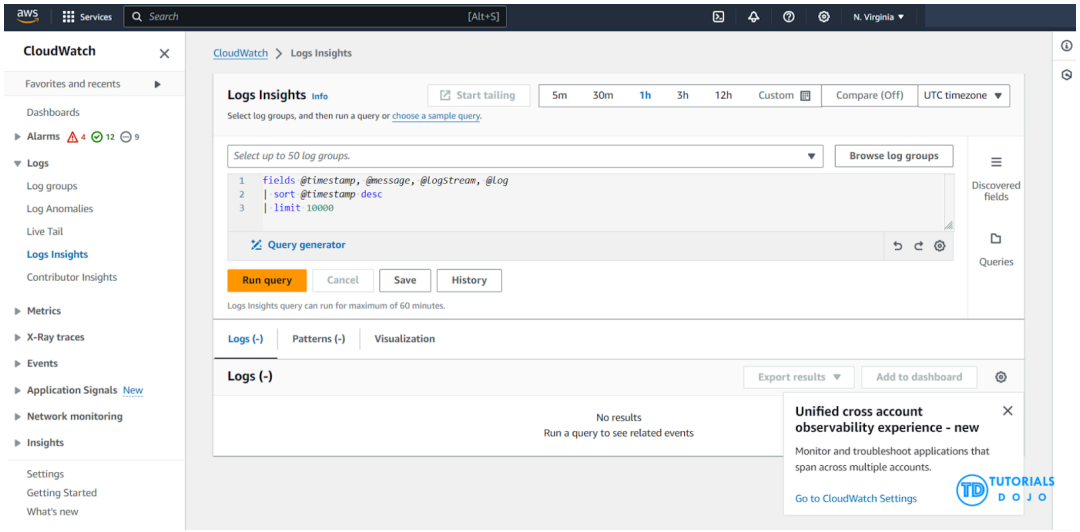
Domain 4: ML Solution Monitoring, Maintenance, and Security Sample Questions

1. A finance company recently moved its on-premises application to AWS, using Amazon EC2 for hosting, Amazon RDS for database management, and Amazon S3 for data storage. The company wants to leverage machine learning to analyze log data from various AWS resources that they are using.

   Which AWS service would best support these requirements?

   a. Amazon Managed Grafana
   b. AWS CloudTrail
   c. AWS Config
   d. Amazon CloudWatch Logs Insights

   Explanation:
   Amazon CloudWatch Logs Insights is a powerful tool designed to facilitate in-depth analysis of log data within AWS environments. It enables users to run queries on log data collected from various AWS services and applications in real-time. By utilizing a purpose-built query language, CloudWatch Logs Insights allows for complex analysis and visualization of log data, helping to detect trends, identify performance bottlenecks, and troubleshoot issues efficiently. This capability is particularly valuable for monitoring applications and infrastructure, as it provides actionable insights into system behavior and operational health.



The service integrates seamlessly with Amazon CloudWatch Logs, offering advanced features for querying and visualizing log data. Users can create custom dashboards, set up alerts based on specific

log patterns, and generate reports to support security and compliance requirements. Users can also use the 'pattern' command, which is backed by machine learning, to automatically detect patterns in log data, assemble similar logs, and summarize vast amounts of log lines into concise, visual formats.

Hence, the correct answer is: **Amazon CloudWatch Logs Insights.**

The option that says: **AWS Config** is incorrect. This service is primarily used for assessing, auditing, and evaluating the configurations of your AWS resources. While it provides valuable insights into configuration drift and compliance status, it does not offer real-time log analysis or querying capabilities for detecting patterns and troubleshooting issues.

The option that says: **AWS CloudTrail** is incorrect because it is just a service that enables governance, compliance, and operational and risk auditing of your AWS account. It records API calls made on your account and provides a history of AWS API calls for your account, including API calls made through the AWS Management Console, AWS SDKs, command-line tools, and other AWS services. It does not provide real-time log analysis and querying capabilities.

The option that says: **Amazon Managed Grafana** is incorrect. This service is primarily a visualization tool rather than a log analysis service and does not inherently analyze log data or utilize machine learning to identify log patterns.

References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html
https://docs.aws.amazon.com/decision-guides/latest/monitoring-on-aws-how-to-choose/monitoring-on-aws-how-to-choose.html

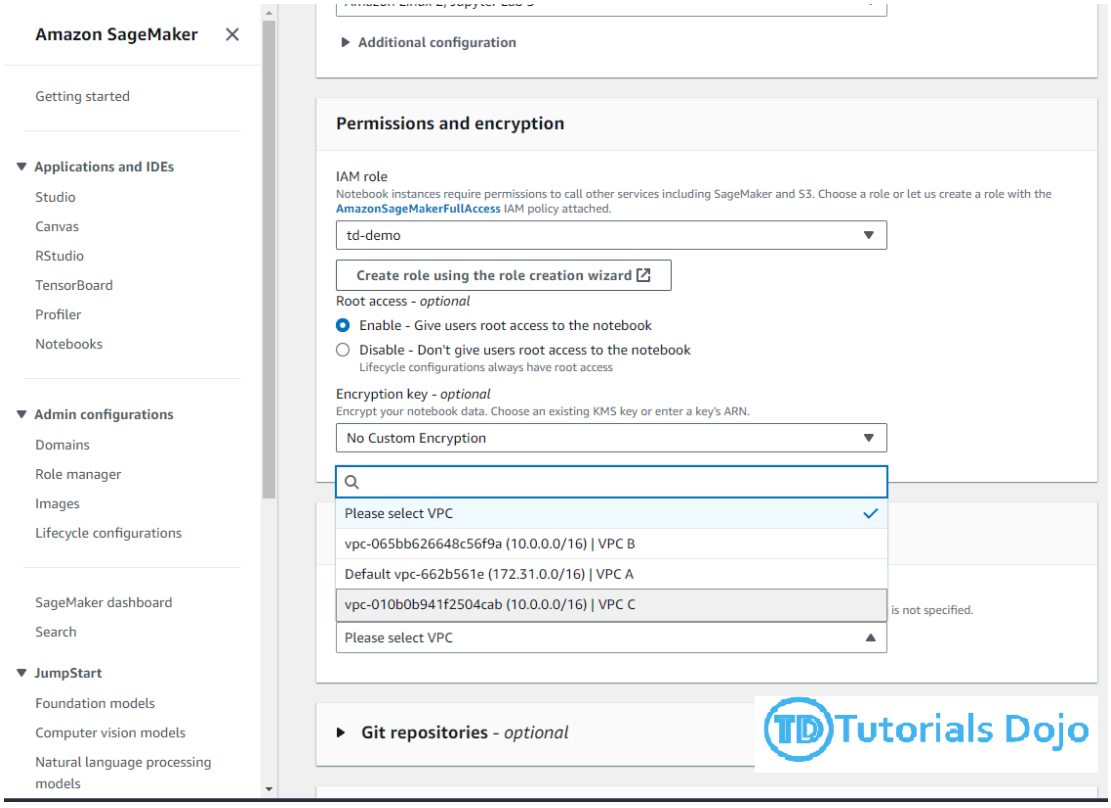
Check out this Amazon CloudWatch Cheat Sheet:
https://tutorialsdojo.com/amazon-cloudwatch/

2. A healthcare company is using Amazon SageMaker to build a machine learning model for medical text analysis, leveraging Amazon Comprehend for natural language processing. Initially, the ML engineer successfully accessed Amazon Comprehend from the SageMaker notebook instance through the public internet. However, the security team has enforced a new policy requiring all AWS services to be accessed through VPC endpoints. The ML engineer updates the SageMaker notebook to use the DNS entry for the VPC endpoint, but the service calls to Amazon Comprehend continue to fail.

What must the ML engineer do to resolve the issue?

a. Update the SageMaker notebook role's permissions to include a policy that allows access to the VPC endpoint.
b. Attach a NAT Gateway to the SageMaker notebook's VPC to allow internet access.
c. **Ensure the SageMaker notebook instance is associated with a private subnet and has access to the VPC endpoint.**
d. Update the SageMaker notebook security group to allow traffic to the VPC endpoint.

Explanation:
Amazon SageMaker can access AWS services like Amazon Comprehend via VPC endpoints using AWS PrivateLink. To achieve this, the SageMaker notebook instance must be associated with a subnet that has access to the VPC endpoint. Simply updating the DNS entry alone is insufficient if the instance resides in a subnet that does not have proper access to the VPC endpoint.



Ensuring the notebook instance is associated with a private subnet that has access to the endpoint and is properly configured to route traffic to the VPC endpoint is important for successful service communication without relying on public internet access.

Hence, the correct answer is: **Ensure the SageMaker notebook instance is associated with a private subnet and has access to the VPC endpoint.**

The option that says: **Update the SageMaker notebook security group to allow traffic to the VPC endpoint** is incorrect. While security groups are important for controlling inbound and outbound traffic, the instance must be on a private subnet to connect to the VPC endpoint

The option that says: **Attach a NAT Gateway to the SageMaker notebook's VPC to allow internet access** is incorrect because the requirement is to just prevent access to AWS services through the public internet.A NAT Gateway is used for internet access from a private subnet. It wouldn't help access Amazon Comprehend through a VPC endpoint, as it is still dependent on the public internet, which the new policy prohibits.

The option that says: **Update the SageMaker notebook role's permissions to include a policy that allows access to the VPC endpoint** is incorrect because IAM permissions alone do not resolve the networking issue. The problem lies in ensuring that the notebook instance is correctly associated with the VPC and subnet that can access the VPC endpoint.

References:
https://docs.aws.amazon.com/sagemaker/latest/dg/host-vpc.html
https://docs.aws.amazon.com/sagemaker/latest/dg/interface-vpc-endpoint.html

Check out this Amazon SageMaker Cheat Sheet:
https://tutorialsdojo.com/amazon-sagemaker/

References for Domain 4

Data quality

What is Amazon SageMaker AI?

Data and model quality monitoring with Amazon SageMaker Model Monitor

How Amazon CloudWatch works

What is Amazon EC2 Auto Scaling?

What Is AWS CloudTrail?

Policies and permissions in AWS Identity and Access Management

AWS Key Management Service

Amazon EC2 Instance types

Deploy models for inference

Amazon EC2 C5 Instances

Using Amazon CloudWatch dashboards

What Is Amazon EventBridge?

Amazon SageMaker Inference Recommender

What is AWS Compute Optimizer?

What is IAM?

IAM roles

IAM user groups

How to use SageMaker AI execution roles

Configure security in Amazon SageMaker AI

Amazon SageMaker Role Manager

What is Amazon VPC?

Security group rules

Access an AWS service using an interface VPC endpoint

AWS PrivateLink concepts

Cloud Financial Management with AWS

Automatic model tuning with SageMaker AI

What is Amazon CloudWatch?

Amazon SageMaker Pipelines

FINAL REMARKS

Whether you are a student wanting to learn more about the cloud, or a fresh graduate trying to enter the industry, or even an experienced professional exploring a new field, the cloud is absolutely a fun and exciting space to be in. There are so many things you can do today that were not feasible before with a local infrastructure setup. All you need is a browser and Internet connectivity and you'll have your whole environment right at your fingertips. And as the days go by, more and more people aspire to be AWS Certified. More and more people want to learn cloud computing and bring their careers to newer heights. And with these certifications, they're like investments on yourself and on your skills. These achievements are acknowledged by everyone in the community.

We at Tutorials Dojo are dedicated to help you achieve these results. We do our best to constantly produce practical and valuable content for everyone who is preparing for his/her AWS certification exams. We have written blogs, guides, cheat sheets, and practice exams that are also constantly being updated based on our experiences and on the feedback of our students. We listen and we deliver.

So if you are currently reading our final remarks, we want to say thank you for choosing Tutorials Dojo and we hope you'll continue supporting us. We also wish you the very best on your future AWS certification exams! Our forums are always open for feedback and we would love to hear from you. It is you, our students, who are the front-runners that help improve the content that we produce.

Once you feel that you have learned the basics, we recommend testing your knowledge through our AWS Certified Machine Learning Engineer Associate Practice Exams. You can also try the free sampler version of our full practice test course here. And if you have any issues, concerns, or constructive feedback on our eBook, feel free to contact us at support@tutorialsdojo.com.

ABOUT THE AUTHORS

Jon Bonso (10x AWS Certified)

Born and raised in the Philippines, Jon is the Co-Founder of **Tutorials Dojo**. Now based in Sydney, Australia, he has over a decade of diversified experience in Banking, Financial Services, and Telecommunications. He's 10x AWS Certified and has worked with various cloud services such as Google Cloud, and Microsoft Azure. Jon is passionate about what he does and dedicates a lot of time creating educational courses. He has given IT seminars to different universities in the Philippines for free and has launched educational websites using his own money and without any external funding.

Kayne Uriel Rodrigo (2x AWS Certified)

An I.T. intern at Tutorials Dojo and a 4th-year Computer Science student who took his Bachelor's degree at **Pamantasan ng Lungsod ng Maynila** *(University of the City of Manila)*. AWS CCP Certified, ISC2 CC Certified, and a **DAP Project SPARTA** Data Science Track graduate, he has two years of project-based experience in AI, cybersecurity, and software engineering. As a co-author of the AIF-C01 Reviewer for Tutorials Dojo, Kayne leverages his expertise to provide comprehensive insights for professionals. Active as a student tech-lead in **AWS Cloud Club Haribon** and won the **2024 AWS Innovation Cup** Hackathon event.

Samantha Vivien L. Servo

A 4th year Computer Science student who took her Bachelor's degree at **Pamantasan ng Lungsod ng Maynila** *(University of the City of Manila)* and an IT intern in **Tutorials Dojo.** With her team, she has won the **2024 AWS Innovation Cup Hackathon**. She is actively involved in campus organizations such as GDSC PLM and AWS Cloud Clubs Haribon, and she's passionate about the convergence of medicine and technology, particularly data science. Through her brand, ***"Beyond the Vinculum,"*** Samantha is committed to pushing the boundaries and shaping the future of these ever-evolving fields.