AWS MACHINE LEARNING CERTIFICATION

DOMAIN #4: MACHINE LEARNING IMPLEMENTATION AND OPERATIONS (20% EXAM)

AWS ML CERTIFICATION EXAM DOMAINS

Domain	% of Examination
Domain 1: Data Engineering	20%
Domain 2: Exploratory Data Analysis	24%
Domain 3: Modeling	36%
Domain 4: Machine Learning Implementation and Operations	20%
TOTAL	100%

Source: https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty_Exam%20Guide%20(1).pdf

DOMAIN #1 OVERVIEW:

SECTION #1: INTRODUCTION, DATA/ML LINGO, AWS DATA STORAGE

- What is Machine Learning and Artificial Intelligence?
- What is Amazon Web Services (AWS)?
- Artificial Intelligence and Machine learning Lingo (data types, Labeled vs. unlabeled, sagemaker groundtruth)
- structured vs. unstructured and database vs. data lake vs. data storage
- AWS Data Storage (Redshift, RDS, S3, DynamoDB)

SECTION #2: AMAZON S3

- Amazon S3 in Depth (partitions, tags)
- Amazon S3 Storage Tiers and Lifecycles
- Amazon S3 Encryption and Security
- Amazon S3 Encryption and Security – Part #2 (ACL, CloudWatch, CloudTrail, VPC)
- Additional Notes (Elastisearch, ElastiCache, and Database vs. data warehouse)

DOMAIN #1 OVERVIEW:

SECTION #3: AWS DATA MIGRATION, GLUE, PIPELINE, STEP AND BATCH

- AWS Glue (crawlers, features, built-in transformations etc)
- AWS Data pipeline
- AWS Data Migration Service (DMS)
- AWS Batch
- Step Function

SECTION #4: DATA STREAMING & KINESIS

- Kinesis Overview
- Kinesis Video Streams
- Kinesis Data Streams
- Kinesis Firehose
- Kinesis Analytics and Random Cut Forest

DOMAIN #2 OVERVIEW:

SECTION #5: JUPYTER NOTEBOOKS, SCIKIT LEARN, PYTHON PACKAGES, AND DISTRIBUTIONS

- Introduction
- Jupyter Notebooks and Scikit Learn
- Python Packages (Pandas, Numpy, Matplotlib and Seaborn
- Distributions (Normal, Standard, Poisson, Bernoulli)
- Time Series

SECTION #6: AMAZON ATHENA, QUICKSIGHT AND ELASTIC MAP REDUCE

- Amazon Athena Features
- Amazon Athena Deep Dive (Security, Cost, and glue integration)
- Amazon QuickSight Features
- Amazon QuickSight (integration with AWS services)
- Amazon QuickSight ML insights and Use Cases
- Elastic Map Reduce (EMR)
- Apache Hadoop with EMR
- Apache Spark with EMR

DOMAIN #2 OVERVIEW:

SECTION #7: FEATURE ENGINEERING

- Introduction to Feature Engineering
  - Amazon SageMaker GroundTruth
  - Feature Selection
  - Scaling
  - Imputation
  - Outliers
  - One Hot Encoding
  - Binning
  - Log Transformation
  - Shuffling, Feature Splitting, Unbalanced Datasets
  - Text Feature Engineering overview
  - Bag of words, punctuation, and dates (easy ones!)
  - Term Frequency Inverse Document Frequency (TF-IDF)
  - N-Grams (Unigram vs. Bigram vs. Trigram)
  - Orthogonal Sparse Bigram (OSB)
  - Cartesian Product Transformation

DOMAIN #3 OVERVIEW:

SECTION #8: MACHINE AND DEEP LEARNING BASICS – PART #1

- Artificial Neural Networks Basics: Single Neuron Model
- Activation Functions
- Multi-Layer Perceptron Model
- How do Artificial Neural Networks Train?
- ANN Parameters Tuning – Learning rate and batch size
- Tensorflow playground
- Gradient Descent and Backpropagation
- Overfitting and Under fitting
- How to overcome overfitting?
- Bias Variance Trade-off
- L1 Regularization
- L2 Regularization

SECTION #9: MACHINE AND DEEP LEARNING BASICS – PART #2

- Artificial Neural Networks Architectures
- Convolutional Neural Networks
- Recurrent Neural Networks
- Vanishing Gradient Problem
- LSTM Networks
- Model Performance Assessment – Confusion Matrix
- Model Performance Assessment – Precision, recall, F1-score
- Model Performance Assessment – ROC, AUC, Heatmap, and RMSE
- K-Fold Cross validation
- Transfer Learning
- Ensemble Learning – Bagging and Boosting

DOMAIN #3 OVERVIEW:

SECTION #10: MACHINE AND DEEP LEARNING IN AWS – BUILT-IN ALGORITHMS PART #1

- AWS SageMaker
- Deep Learning on AWS
- SageMaker Built-in algorithms
- Object Detection
- Image Classification
- Semantic Segmentation
- SageMaker Linear Learner
- Factorization Machines
- XG-Boost
- SageMaker Seq2Seq
- SageMaker DeepAR
- SageMaker Blazing Text

SECTION #11: MACHINE AND DEEP LEARNING IN AWS – BUILT-IN ALGORITHMS PART #2

- Random Cut Forest
- Neural Topic Model
- LDA
- K-Nearest Neighbours (KNN)
- K Means
- Principal Component Analysis (PCA)
- IP insights
- Reinforcement Learning
- Object2Vec
- Automatic Model Tuning
- SageMaker and Spark

DOMAIN #3 OVERVIEW:

SECTION #12: MACHINE AND DEEP LEARNING IN AWS – HIGH LEVEL AI/ML PART #3

- ReKognition
- Amazon Comprehend and Comprehend Medical
- Translate
- Transcribe
- Polly
- Forecast
- Lex
- Personalize
- Textract
- AWS DeepLens
- AWS DeepRacer

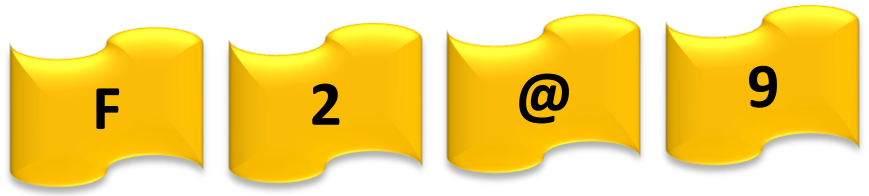
DOMAIN #4: MACHINE LEARNING IMPLEMENTATION AND OPERATIONS

SECTION #13: MACHINE LEARNING IMPLEMENTATION AND OPERATIONS

- SageMaker components overview
- AI/ML Models Deployment
- SageMaker Resources and Instances Management
- Inference Pipeline
- Amazon SageMaker Neo
- Docker Containers
- Model production variants
- Canary Deployment
- Greengrass
- Model Evaluation and validation
- SageMaker Security (IAM, VPC, Encryption)
- SageMaker Security (CloudWatch, CloudTrail)

RECALL OUR MINI CHALLENGE AND PRIZE!

- For those of you who will successfully complete the entire section and watch the videos till the end, they will receive a valuable prize!



https://pixabay.com/illustrations/trophy-win-prize-transparent-1414791/

SAGEMAKER COMPONENTS REVIEW

AMAZON SAGEMAKER COMPONENTS

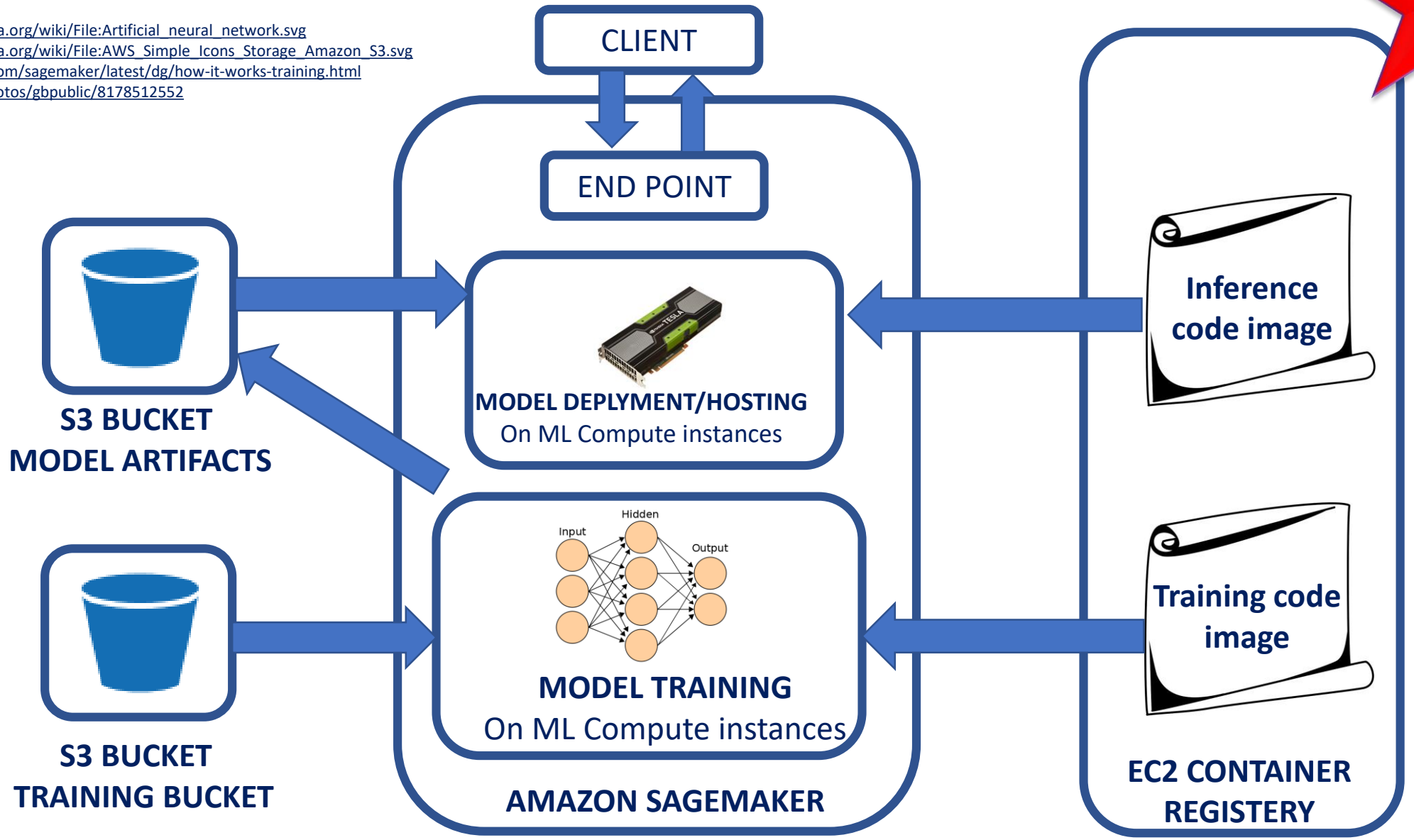
- Two components are present in Amazon SageMaker:
  - Model training
  - Model deployment
- To start training an AI/ML model using Amazon SageMaker, you will need to create a training job with the following:
  - **Amazon S3 bucket URL (training data):** where the training data is located.
  - **Compute resources:** Amazon SageMaker will train the model using instances managed by Amazon SageMaker.
  - **Amazon S3 bucket URL (Output):** this bucket will host the output from the training.
  - **Amazon Elastic Container Registry path:** where the training code is stored.
- Amazon SageMaker launches an ML compute instances once a training job is initiated.
- Amazon SageMaker uses: (1) training code and (2) training dataset to train the model.
- Amazon SageMaker saves the trained model artifacts in an S3 bucket.

Source: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

AMAZON SAGEMAKER MODEL TRAINING AND DEPLOYMENT OVERVIEW

Source: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg
https://commons.wikimedia.org/wiki/File:AWS_Simple_Icons_Storage_Amazon_S3.svg
https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html
https://www.flickr.com/photos/gbpublic/8178512552



TRAINING OPTIONS OFFERED BY SAGEMAKER

USE AN ALGORITHM PROVIDED BY AMAZON SAGEMAKER

- Amazon SageMaker provides ready, off the shelve training algorithms such as: Linear Learner Algorithm and the XGBoost Algorithm, K Means, Principal Component Analysis, image classification, LDA, Sequence to Sequence Algorithm.

USE APACHE SPARK WITH AMAZON SAGEMAKER

- Apache Spark can be used to train models with Amazon SageMaker.

CUSTOM CODE TRAINING USING POPULAR DEEP LEARNING FRAMEWORKS

- custom python code with TensorFlow or Apache MXNet for model training.

USE YOUR OWN CUSTOM ALGORITHMS:

- The code could be placed in a docker container and then a registry path of the image could be provided in an Amazon SageMaker CreateTrainingJob API call.

AWS MARKETPLACE

- Choose an algorithm from Amazon marketplace, https://aws.amazon.com/marketplace/solutions/machine-learning

Source: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

SAGEMAKER MODEL DEPLOYMENT

MODEL DEPLOYMENT BY SAGEMAKER

- After the AI/ML model is trained, it can be deployed as follows:
  - **For Single Prediction at a time:** set up a persistent endpoint using Amazon SageMaker hosting services.
  - **For Multiple Predictions:** Use Amazon SageMaker batch transform in order to obtain predictions for an entire dataset.
- **SageMaker Inference Pipeline:** SageMaker offers tools to process batch transforms in a pipeline format.
- **Batch Transform:** is used to allow preprocessing of large datasets and performing model inferencing quickly/efficiently without a need to have a persistent endpoint.
- **SageMaker Automatically Scaling:** as the workload changes throughout the day, SageMaker can dynamically adjust the number of instances provisioned so it could save money and compute resources.
- **Amazon SageMaker Elastic Inference (EI):** EI could be used to speed up inference and reduce latency by adding an accelerator without the need of having a dedicated GPU (which will cost much more)
- **Amazon SageMaker Neo:** Used to train TensorFlow, Apache MXNet, PyTorch, ONNX, and XGBoost models once and optimize them for deployment on ARM, Intel, and Nvidia processors. (*Before Neo, you will need to spend major man-hour efforts to deploy AI/ML Models on a specific hardware with specific compiler, memory, operating systems…etc).*

SAGEMAKER RESOURCES AND INSTANCE TYPES

INSTANCE TYPES SELECTION

- Amazon SageMaker offers a wide variety of instance types to fit many machine learning use cases:
  - Training vs. deployment
  - Deep learning vs. simple regression
- Instance types varies based on: CPU, GPU, memory, and networking capacity
- Resources could be scaled to meet target workload requirements.
- Instance types for deep learning will require a GPU and include the following:
  - P3: 8 Tesla V100 GPU's
  - P2: 16 K80 GPU's
  - G3: 4 M60 GPU's (all Nvidia chips)
- Inference instances are less computationally expensive therefore a compute instance would be sufficient.
  - C4
  - C5

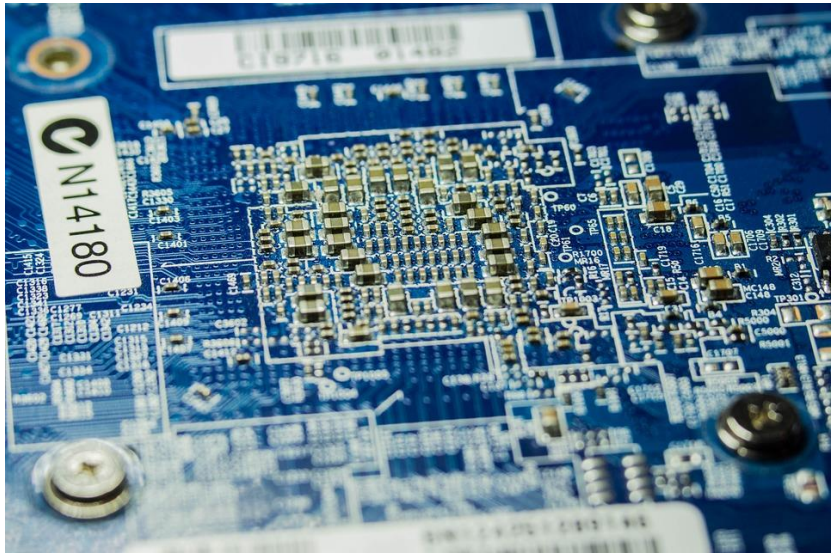


Photo Credit: https://pixabay.com/photos/transistors-gpu-processor-pc-chip-1137502/

INSTANCE TYPES SELECTION

- Check this out for a full list of ML instance Types: https://aws.amazon.com/sagemaker/pricing/instance-types/

STANDARD

Instance type	vCPU	GPU	Mem (GiB)	GPU Mem (GiB)	Network Performance
Standard – Current Generation					
ml.t2.medium	2	-	4	-	Low to Moderate
ml.t2.large	2		8		Low to Moderate

MEMORY OPTIMIZED

Memory Optimized - Current Generation					
ml.r5.large	2	-	16	-	Up to 10 Gbps
ml.r5.xlarge	4	-	32	-	Up to 10 Gbps
ml.r5.2xlarge	8	-	64	-	Up to 10 Gbps
ml.r5.4xlarge	16	-	128	-	Up to 10 Gbps

COMPUTE OPTIMIZED

Compute Optimized – Current Generation					
ml.c5.large	2		4		Up to 10 Gbps
ml.c5.xlarge	4	-	8	-	Up to 10 Gbps
ml.c5.2xlarge	8	-	16	-	Up to 10 Gbps
ml.c5.4xlarge	16	-	32	-	Up to 10 Gbps
ml.c5.9xlarge	36	-	72	-	10 Gigabit

ACCELERATED COMPUTING

Accelerated Computing – Current Generation					
ml.p3.2xlarge	8	1xV100	61	16	Up to 10 Gbps
ml.p3.8xlarge	32	4xV100	244	64	10 Gigabit
ml.p3.16xlarge	64	8xV100	488	128	25 Gigabit
ml.p3dn.24xlarge	96	8xV100	768	256	100 Gigabit

Photo Credit: https://pixabay.com/photos/transistors-gpu-processor-pc-chip-1137502/

SPOT INSTANCES

- A Spot offers a lower price compared to an on-Demand instance.
- “Spot price” is the price you pay for the spot instance and is adjusted based on availably zone and demand.
- The spot instance will run when:
  - When capacity permits.
  - When the maximum hourly price set is more than the Spot price, the instance will run.
- Choose Spot instances when:
  - You have limited resources
  - When you have some degrees of flexibility. Note that your applications can be interrupted.
  - You want to perform optional tasks, data analysis, and batch jobs.
- Spot instances could save up to 90% compared to on-demand instances

Photo Credit: https://pxhere.com/en/photo/747848

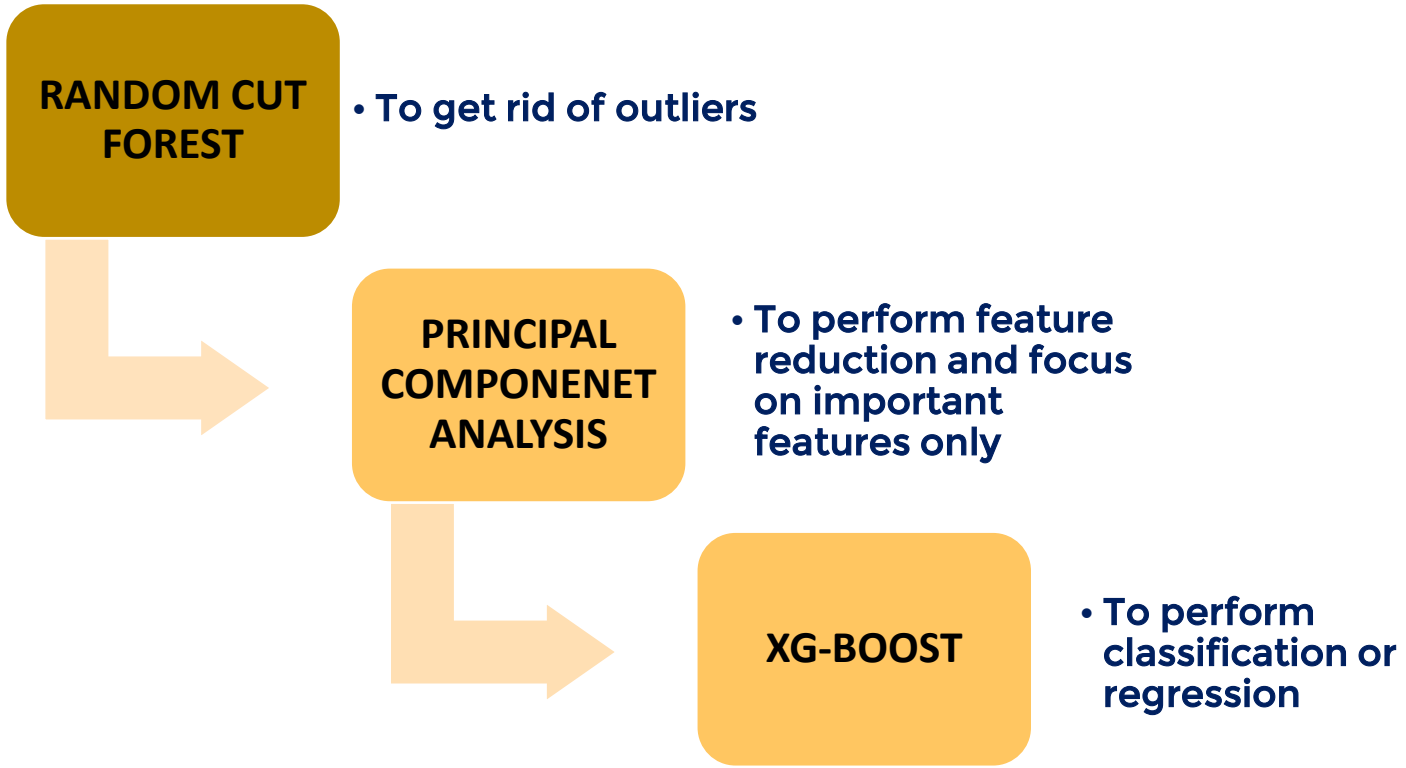
AMAZON ELASTIC INFERENCE: OVERVIEW

- After the AI/ML models are trained, they are deployed for inference to make predictions.
- Using a standalone dedicated GPU to perform inference is generally an overkill!
- Inference requires a small amount of GPU compute power
- This will result in a very high cost with an underutilized resources.
- Amazon Elastic Inference is designed to help reduce deep learning inference cost by 75%.
- Amazon Elastic inference allows you to attach low-cost GPU-powered acceleration to Amazon EC2 and SageMaker instances with no code changes.
- Amazon Elastic Inference supports TensorFlow, Apache MXNet.

https://aws.amazon.com/machine-learning/elastic-inference/

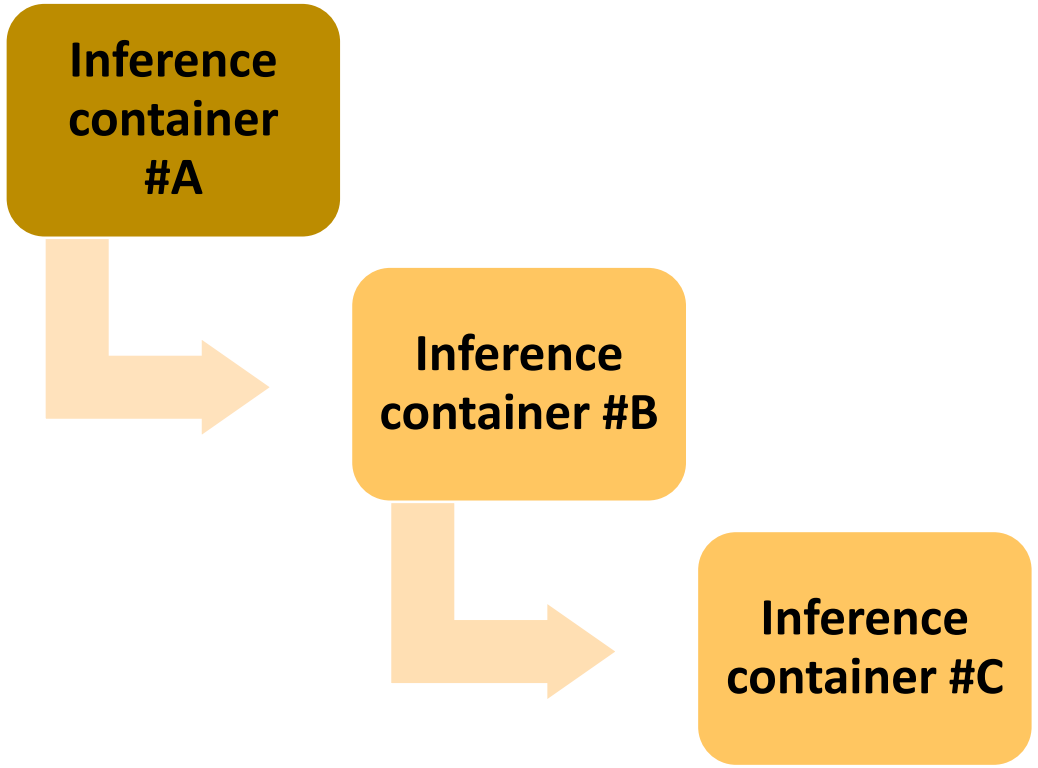
INFERENCE PIPELINE

- Sometimes we might need to apply many algorithms in a sequential fashion as follows:

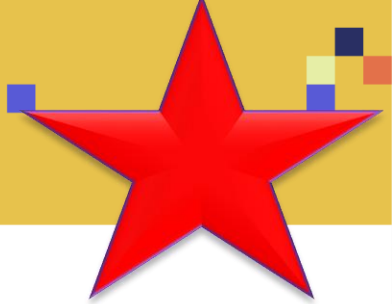


INFERENCE PIPELINE

- SageMaker Model might be composed of a “pipeline” which is a sequence of many containers that process data in a sequential fashion.
- Built-in algorithms or custom-based algorithms in containers could be used to create a Pipeline in Sagemaker



AUTOMATIC SCALING



- Amazon SageMaker offers an automatic scaling feature in which the number of instances could be adjusted in response to the workload.
  - If the workload goes up, the number of instances is increased.
  - If workload goes down, the number of instances is reduced.
- SageMaker users have to develop a scaling policy which in turn uses Amazon CloudWatch along with target values assigned by the user.
- Automatic scaling has the following components:
  - Permissions: those are required to perform automatic scaling actions.
  - AWS Identity and Access Management (IAM) role linked to an AWS service.
  - Target metric: CloudWatch metric used by Amazon SageMaker automatic scaling to decide when to scale.
  - Min/Max capacity: min/max number of instances used during scaling.
  - Cool down period: time required after a scale-in or scale-out activity completes before another scale-out activity can start.

MODEL EVALUATION (VALIDATION)

MODEL VALIDATION

- After the model is trained, its performance needs to be assessed.
- This is critical to ensure that the model meets the business goals.
- There are generally two ways of validating the model, online validation and offline validation.

ONLINE VALIDATION (LIVE DATA)

- Validation preformed using real world dataset on live traffic
- Models trained online are called production variants
- Direct 10% of the traffic to the model under test and 90% to the original model
- A/B Testing

OFFLINE VALIDATION (HISTORICAL DATA)

- Validation performed using historical data (not live traffic)
- Training data = 75% entire dataset
- Validation data = 25% entire dataset

OFFLINE VALIDATION OPTIONS

- There are generally two types of offline validation: holdout set and k-fold.

HOLDOUT SET

- Holdout set is a portion of the dataset (20%-30%) that is not used during model training.
- The holdout set is used to assess the performance of the model using data that the model has never seen before during training.
- Good trained model should be able to "generalize" and not "memorize".

K-FOLD

- Dataset is split into k parts (k ranges from 5 to 10).
- You treat each of these parts as a holdout set for k training runs, and use the other k-1 parts as the training set for that run.
- k trained models are generated
- Comparing these models performance is used to assess the final model (generalization capability).

K-FOLD CROSS VALIDATION

- Cross-validation is used to assess the generalization capability of the model (with data that has never been seen before).
- K-fold cross validation is a technique used to assess the performance of machine learning models.
- K-Fold works by randomly dividing the data into equal sized K groups (folds).
- First fold is considered the validation dataset, and the model is trained on the remaining k − 1 folds.
- By comparing the performance of the model with these many folds, you can now know if the model is over fitted or not.
- If the error is comparable across various runs then the model is generalized.

Model1: *Trained on Fold1 + Fold2 and Tested on Fold3*
Model2: *Trained on Fold2 + Fold3 and Tested on Fold1*
Model3: *Trained on Fold1 + Fold3 and Tested on Fold2*

MODEL VALIDATION: XGBOOST METRICS

Metrics Computed by the XGBoost Algorithm

The XGBoost algorithm computes the following nine metrics during training. When tuning the model, choose one of these metrics as the objective to evaluate the model.

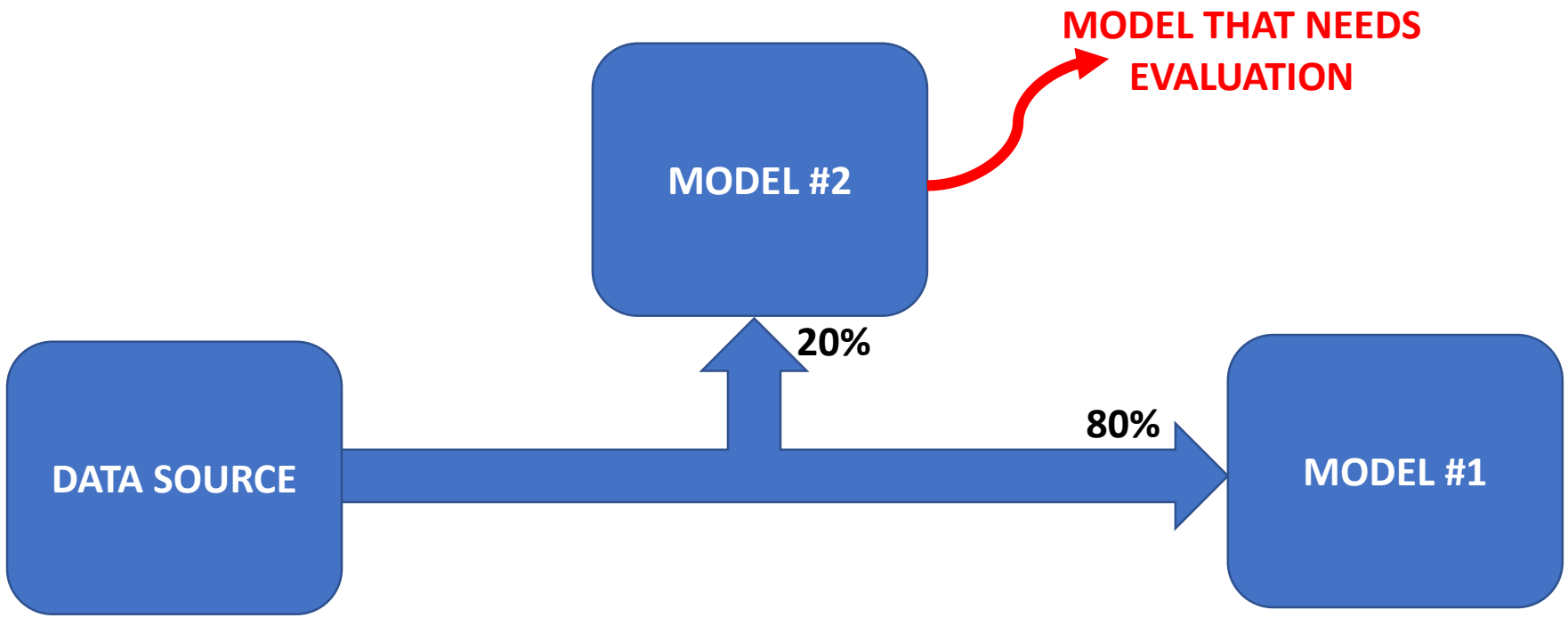
Metric Name	Description	Optimization Direction
`validation:accuracy`	Classification rate, calculated as #(right)/#(all cases).	Maximize
`validation:auc`	Area under the curve.	Maximize
`validation:error`	Binary classification error rate, calculated as #(wrong cases)/#(all cases).	Minimize
`validation:f1`	Indicator of classification accuracy, calculated as the harmonic mean of precision and recall.	Maximize
`validation:logloss`	Negative log-likelihood.	Minimize
`validation:mae`	Mean absolute error.	Minimize
`validation:map`	Mean average precision.	Maximize
`validation:merror`	Multiclass classification error rate, calculated as #(wrong cases)/#(all cases).	Minimize
`validation:mlogloss`	Negative log-likelihood for multiclass classification.	Minimize
`validation:mse`	Mean squared error.	Minimize
`validation:ndcg`	Normalized Discounted Cumulative Gain.	Maximize
`validation:rmse`	Root mean square error.	Minimize

https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-tuning.html

PRODUCTION VARIANTS

PRODUCTION VARIANTS/CANARY DEPLOYMENT

- Production variants are used to test out several models on incoming live traffic.
- If many models are deployed, variants weights can tell Amazon SageMaker how to distribute traffic among the models.
- A subset of the live traffic (let's say 20%) could be directed to the model that requires validation.
- After the model is evaluated and its performance is assessed, 100% of the traffic could be directed to it.



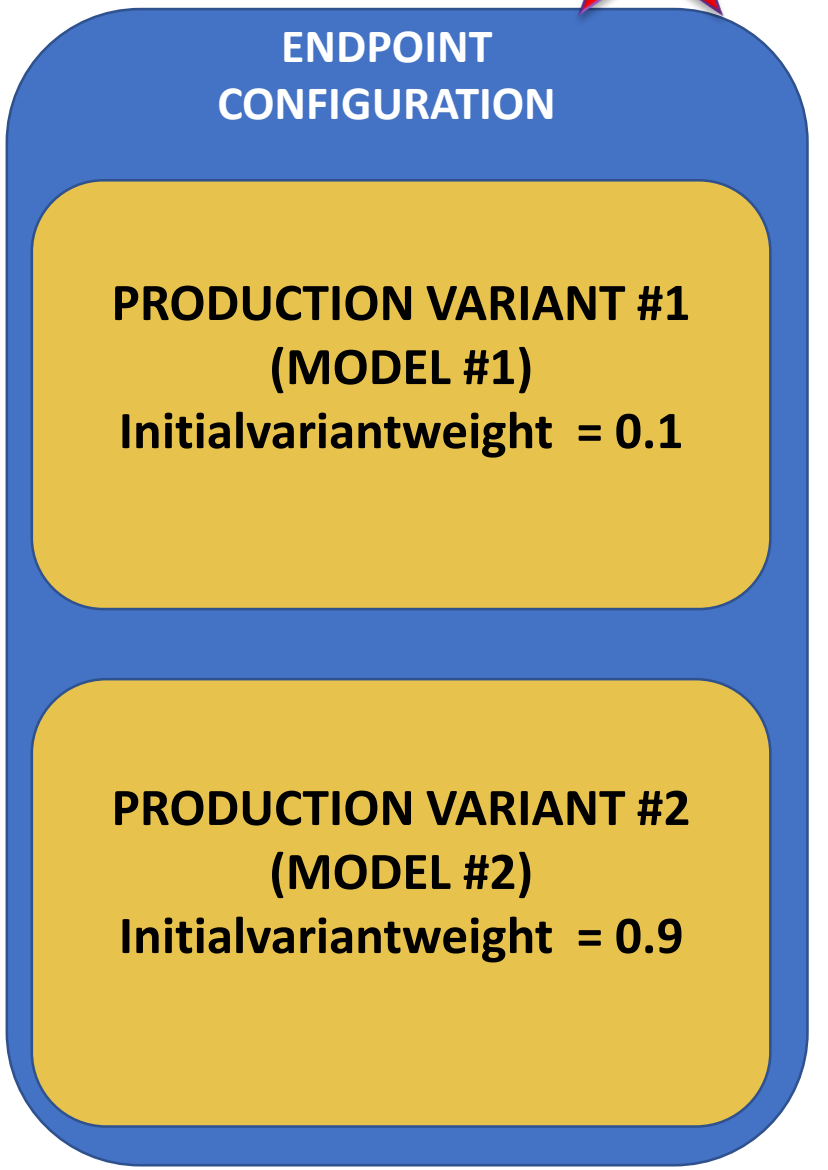
PRODUCTION VARIANTS: INITIAL VARIANT WEIGHT

InitialVariantWeight:

- It specifies the traffic distribution among the models specified in an endpoint.

- Traffic is calculated as follows:

$$Traffic\ (\%) = \frac{VariantWeight}{Sum\ of\ all\ Variant\ weight\ values\ across\ production\ variants}$$



CANARY DEPLOYMENT

CANARY DEPLOYMENT

- Canary deployment is used to minimize the risk of deploying a new AI/ML model.
- Instead of directing the entire traffic to the new model, an incremental approach is taken.
- 10% of the traffic is directed to the new model and the remaining 90% is directed to the old (trusted) model.
- As we have more confidence in the new model, we will start to replace the current (old) model completely.
- You have to include a parameter that indicate which model is currently running and reporting metrics.

AMAZON SAGEMAKER NEO

AMAZON SAGEMAKER NEO: OVERVIEW

- Amazon SageMaker Neo solves one of the most difficult challenges which is how to deploy and tune machine learning models to run on a specific target hardware.
- Optimizing machine learning models to run on a specific hardware is an extremely difficult task since it requires Expert knowledge of:
  - Hardware architecture
  - Instruction set
  - Memory

SAGEMAKER NEO

NEO OPTIMIZES TRAINED MODEL TO A SPECIFIC TARGET HARDWARE

AMAZON SAGEMAKER NEO: OVERVIEW

- Amazon SageMaker Neo allows for Training the machine learning models once and running trained machine learning models anywhere on the edge or in the cloud.
- Edge devices generally have limited memory and compute requirements.
- Therefore, Neo will optimize models to run extremely fast on the selected edge device so reduce latency.
- SageMaker Neo consists of a compiler and a runtime.

SAGEMAKER NEO

NEO OPTIMIZES TRAINED MODEL TO A SPECIFIC TARGET HARDWARE

AMAZON SAGEMAKER NEO: APPLICATION EXAMPLE

- For autonomous vehicles as an example, two challenges exist:
  - Many hardware platforms exist so building and optimizing ML models to run on a specific target hardware is challenging and time consuming.
  - Latency, sensors and compute need to react quickly with minimum latency.
- Amazon Neo is here to solve these problems!
- Neo optimized ML models so it could run much faster with minimum latency.

Photo Credit: https://www.flickr.com/photos/smoothgroover22/15104006386

AMAZON SAGEMAKER NEO: STEPS

- Steps:
  - Train a machine learning model using SageMaker and with your favorite framework: MXNet, TensorFlow, PyTorch, or XGBoost.
  - Select target hardware platform from Intel, NVIDIA, or ARM.
  - Run SageMaker Neo and it will compile the model and generate an executable. Neo achieves massive gains in performance by using an artificial neural network that will implement hardware-specific performance optimizations.
  - Deploy trained model in cloud or edge.
- Neo Greengrass can allow for compute and ML inference capabilities to the edge.
- AWS Greengrass allows for direct model deployment to the edge with over the air updates.

AMAZON SAGEMAKER NEO: HOW IT WORKS?

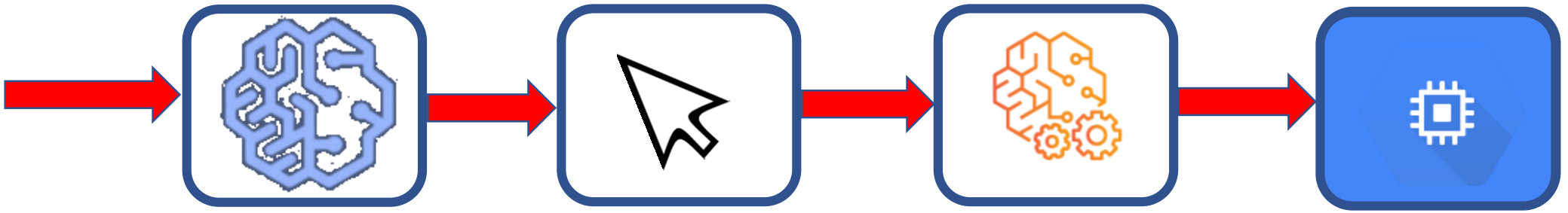
Source: https://aws.amazon.com/sagemaker/neo/

TENSORFLOW

PYTORCH

APACHE MXNET

XGBOOST



SAGEMAKER

Train model using SageMaker

USER SELECTION

User selects target hardware platform

SAGEMAKER NEO

NEO optimizes trained model to a specific target hardware

MODEL DEPLOYMENT ON CLOUD OR EDGE

AWS GREENGRASS

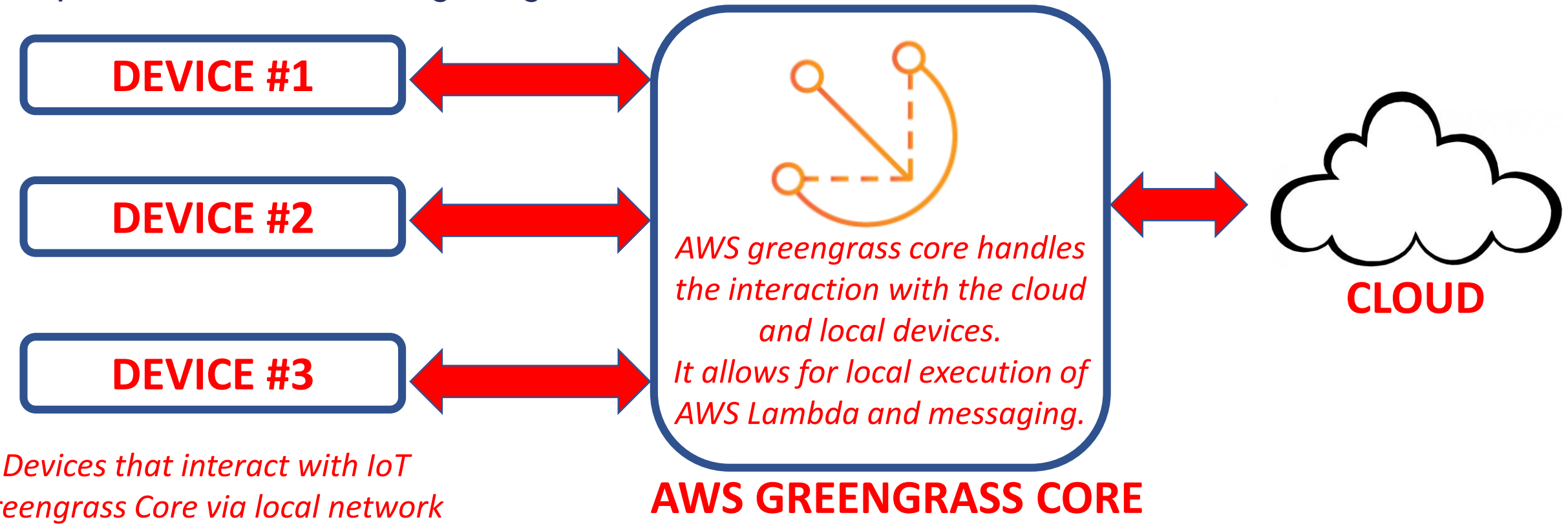
AWS GREENGRASS: OVERVIEW

- AWS Greengrass is an extremely powerful tool that enables developers to build IoT devices and application logic.
- AWS IoT Greengrass is a software that allows local devices to have cloud capabilities.
- AWS IoT Greengrass allows for data collection and secure communication with several local devices on the network.
- Local devices can use AWS IoT Greengrass to communicate securely and send data to the AWS cloud.
- AWS IoT Greengrass developers rely on AWS Lambda functions to create serverless applications.
- Watch Video: https://aws.amazon.com/greengrass/

AWS GREENGRASS

AWS GREENGRASS: HOW DOES IT WORK?

- AWS IoT Greengrass allows for easy development of IoT solutions that connect many devices to each other and to the cloud.
- Devices could be microcontroller or applicances.
- AWS IoT Greengrass Core works as a hub to communicate.
- https://aws.amazon.com/greengrass/



https://www.needpix.com/photo/1419816/image-picture-drawing-imagination-creature-expression-cloud-cloud-2-free-pictures

SAGEMAKER AND DOCKER CONTAINERS

WHAT IS A CONTAINER?

- A container packages up code along with all its dependencies so that it could reliably run across many compute environments.
- Containers are so powerful since they isolate software from its environment.
- A docker container image contains everything needed to successfully run an applications such as:
  - Code
  - System tools
  - Libraries
  - Runtime
  - Settings

SageMaker Developer Guide: https://gmoein.github.io/files/Amazon%20SageMaker.pdf

Resource: https://docs.aws.amazon.com/sagemaker/latest/dg/amazon-sagemaker-containers.html

SAGEMAKER CONTAINERS

- Models in SageMaker are hosted in Docker containers
  - Pre-built Models in scikit Learn and Spark ML
  - Pre-built Models in Tensorflow, MXNet, and PyTorch
  - Custom algorithm

- Amazon SageMaker containers is a library that enables developers to:
  - Create containers to run scripts
  - Train AI/ML models
  - Deploy models

- How to install Amazon SageMaker Containers library?
  - You can run this command from the Dockerfile

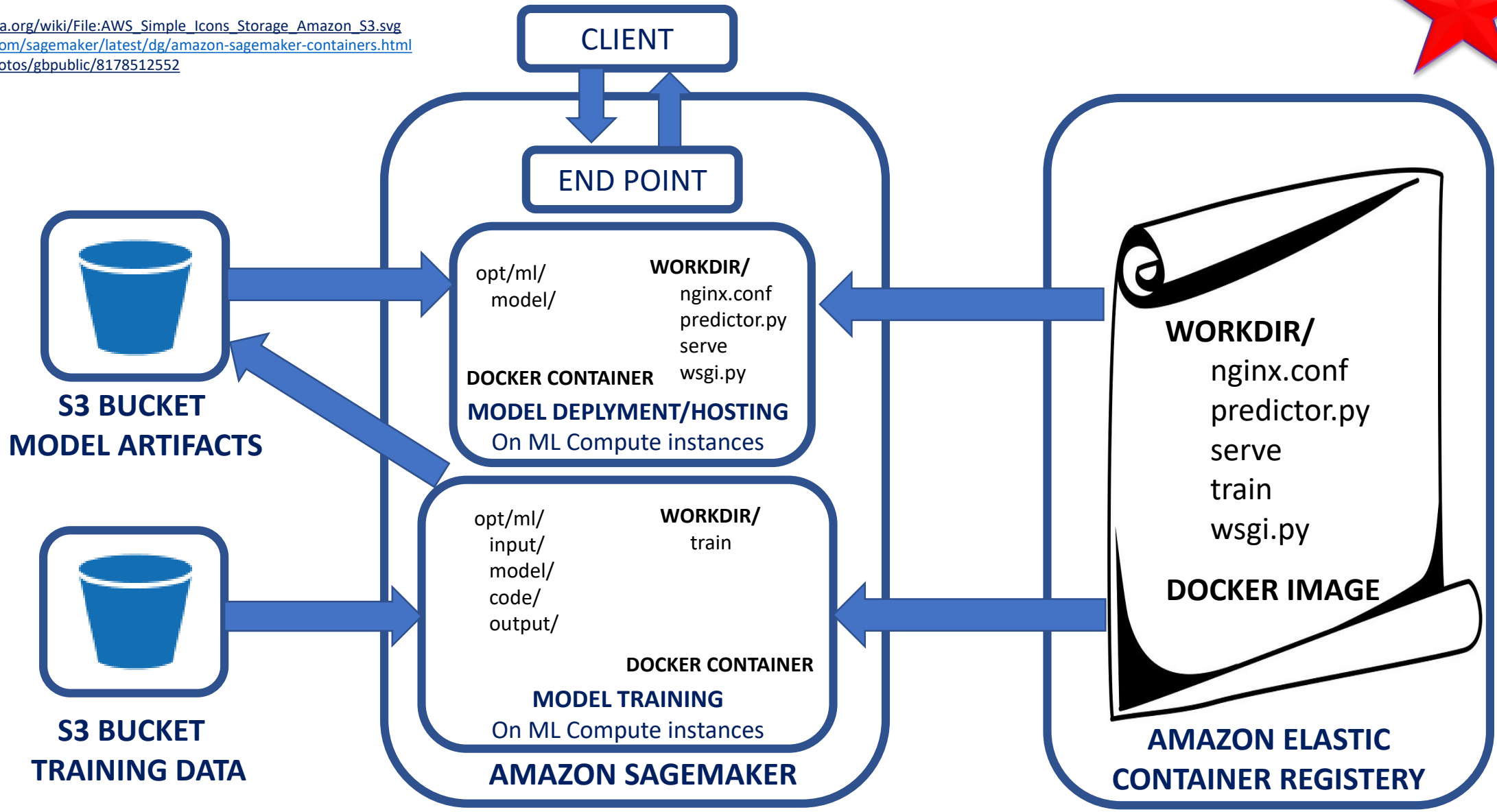
RUN pip install sagemaker-containers

SAGEMAKER CONTAINERS

- The library specifies where to store the code and other resources for both training and inference.
- The Dockerfile copies the code to be run into the location expected by an Amazon SageMaker-compatible container.
- When the container is initiated, the Dockerfile indicates the entry point containing the code to run.
- After a Docker image is built, it can be pushed to the Amazon Elastic Container Registry (ECR).

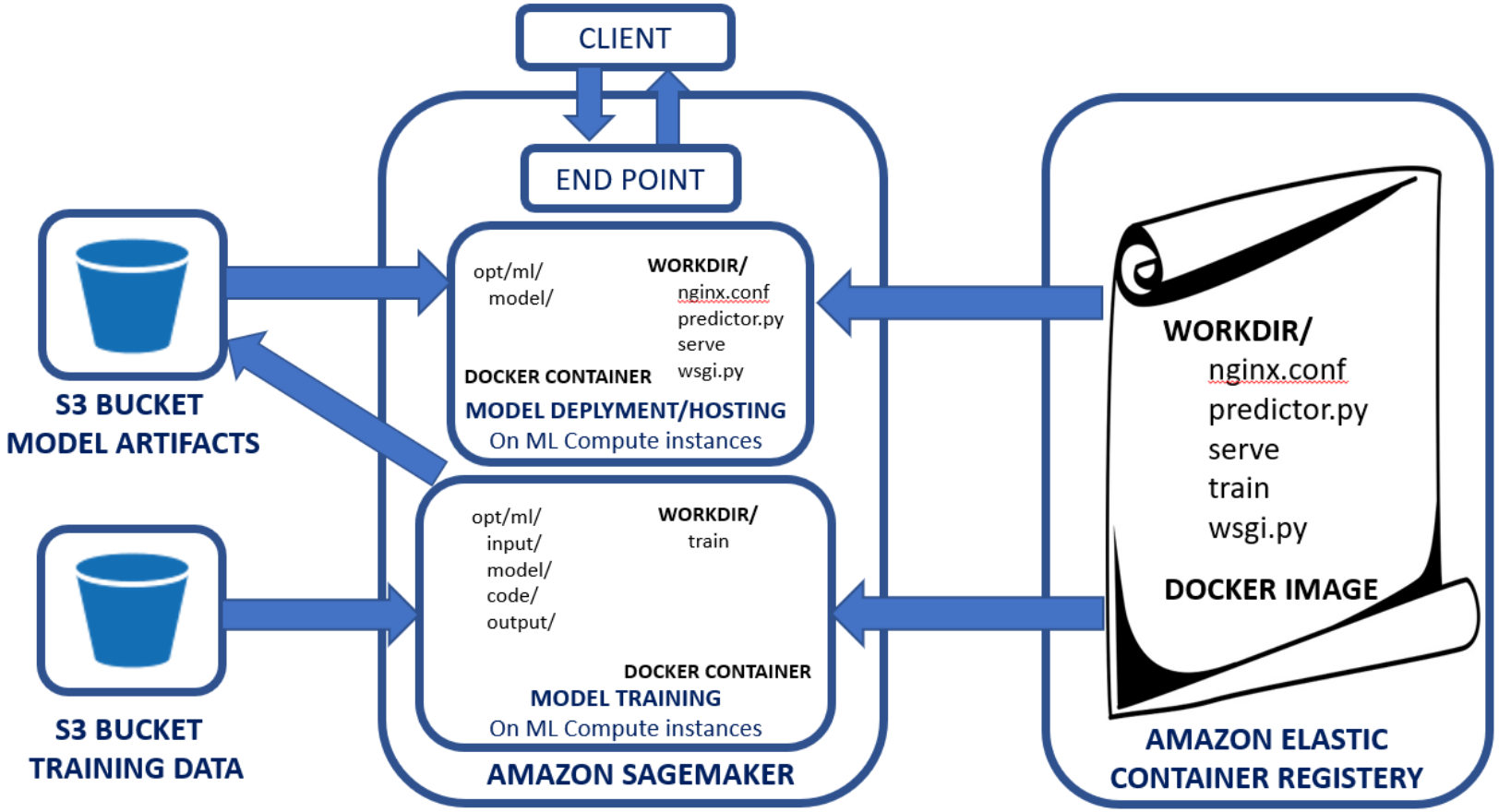
AMAZON SAGEMAKER CONTAINERS SCHEMATIC

https://commons.wikimedia.org/wiki/File:AWS_Simple_Icons_Storage_Amazon_S3.svg
https://docs.aws.amazon.com/sagemaker/latest/dg/amazon-sagemaker-containers.html
https://www.flickr.com/photos/gbpublic/8178512552



AMAZON SAGEMAKER CONTAINERS SCHEMATIC

- Docker containers are created from docker images
- Images are built from a Dockerfile
- Images are saved Amazon Elastic Container Registry



SAGEMAKER CONTAINER STRUCTURE DURING TRAINING AND INFERENCE

- The following files are created in the container's /opt/ml directory when Amazon SageMaker trains a model.

DURING TRAINING

```
/opt/ml
├── input
│   ├── config
│   │   ├── hyperparameters.json
│   │   └── resourceConfig.json
│   └── data
│       └── <channel_name>
│           └── <input data>
├── model
│
├── code
│   └── <script files>
│
└── output
    └── failure
```

DURING INFERENCE

```
/opt/ml
└── model
    └── <model files>
```

Source: https://docs.aws.amazon.com/sagemaker/latest/dg/amazon-sagemaker-containers.html

SAGEMAKER CONTAINER STRUCTURE DURING TRAINING

- **/opt/ml/input/**
  - includes JSON files that configure the algorithm's hyperparameters and network architecture.
  - The directory also indicates the channels that SageMaker will use to access the data in S3.
- **/opt/ml/code/**
  - Contains scripts to run.
- **The /opt/ml/model/**
  - Contains model generated by the algorithm.
- **/opt/ml/output/**
  - Includes data such as why a training job has failed.

DURING TRAINING

```
/opt/ml
├── input
│   ├── config
│   │   ├── hyperparameters.json
│   │   └── resourceConfig.json
│   └── data
│       └── <channel_name>
│           └── <input data>
├── model
│
├── code
│   └── <script files>
│
└── output
    └── failure
```

Source: https://docs.aws.amazon.com/sagemaker/latest/dg/amazon-sagemaker-containers.html

SAGEMAKER CONTAINER STRUCTURE DURING HOSTING/DEPLOYMENT/INFERENCE

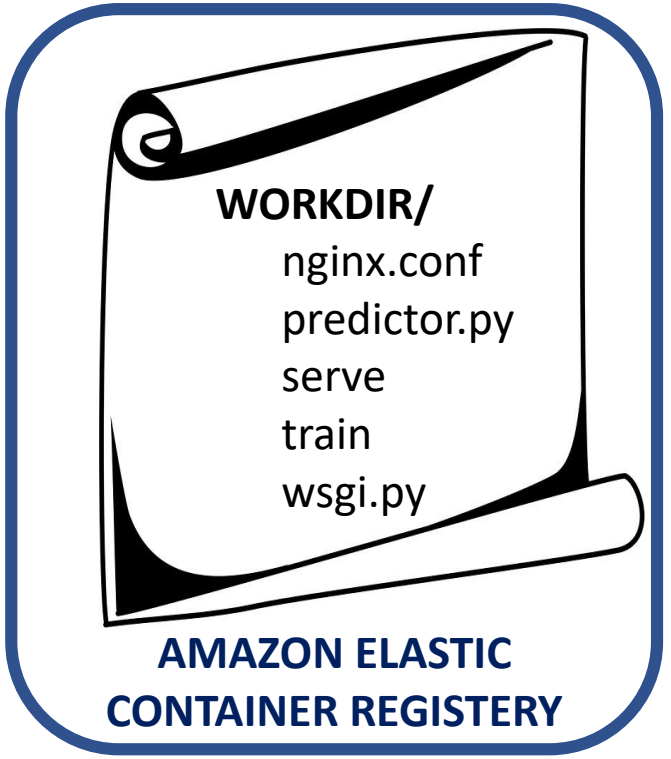
- In the container, the model files are in the same place that they were written to during training.

DURING HOSTING/DEPLOYMENT/ INFERENCE

```
/opt/ml
└── model
    └── <model files>
```

DOCKER IMAGE STRUCTURE

- When amazon sagemaker models are trained, they are deployed to an HTTP endpoint.
- This allows to perform real-time model inference.
- The container contains a serving stack to process requests.
- In SageMaker, the serving stack files are installed in the container's WORKDIR which contains the following files.
  - *nginx.conf: configuration file for nginx front end.*
  - *predictor.py: program that implements Flask web server*
  - *Serve: program that run when container is started for hosting.*
  - *Train: program that is invoked when the container is run during training.*
  - *wsgi.py – A wrapper used to invoke Flask application.*



https://gmoein.github.io/files/Amazon%20SageMaker.pdf

AMAZON SAGEMAKER SECURITY

AMAZON SAGEMAKER SECURITY OVERVIEW

AWS SAGEMAKER SECURITY: OVERVIEW

- Amazon SageMaker ensures the highest level of security to its customers.
- Amazon SageMaker follows a ***shared security model*** as follows:

  - **Security of the cloud:**
    - ❖ AWS ensures the protection of the infrastructure.
    - ❖ All services offered by AWS are very secure.
    - ❖ Security is being regularly audited by third party to ensure compliance.

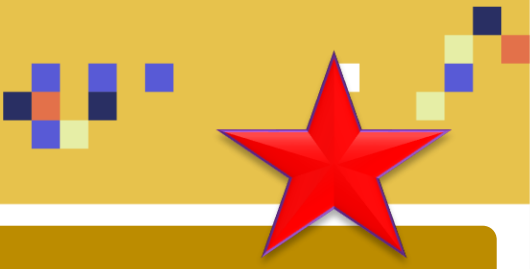
  - **Security in the cloud:**
    - ❖ Users of AWS are responsible for their own data sensitivity and organization requirements.

https://pixabay.com/illustrations/security-technology-risk-management-4210502/

AWS S3 SECURITY: RESOURCE Vs. USER BASED POLICES

- S3 buckets are private by default, only the owner can access the bucket.
- Owner can allows access to others by creating an access policy.
- There are generally two types of polices in AWS to grant permission to resources in AWS S3:
  - **Resource-based:** these are access polices that you attach to your buckets.
    - ❖ Example: bucket polices and Access Control List (ACL)
  - **User-based:** polices assigned to specific users in the account
    - ❖ Example: Identity and Access Management (IAM) polices
- Both polices use JSON-based access policy language.

SAGEMAKER SECURITY



IDENTITY AND ACCESS MANAGEMENT (IAM)

- IAM allows for managing access to AWS services in a secure fashion. This is achieved by creating AWS users/groups and then giving/denying access to AWS resources.

MULTIFACTOR AUTHENTICATION (MFA)

- MFA adds additional level of security by asking users to enter a unique authentication using an MFA mechanism besides the normal sign-in credentials.

DATA ENCRYPTION

- Encryption at rest or in transit can enhance data security.

CLOUDTRAIL

- CloudTrail is used to track activity made by users, roles on AWS services. CloudTrail provides a record of past requests, IP addresses, timing of the requests.

SSL/TLS

- Transport Layer Security and Secure Sockets Layer are cryptographic protocols used to ensure communications security over a network.

VPC

- Amazon Virtual Private Cloud (VPC) allows users to create an AWS resources inside a virtual Network. This means that the traffic will never leave or go through the public internet and will stay inside the VPC for maximum security.

MACIE

- Allows for finding and securing personal data in S3

AMAZON SAGEMAKER ENCRYPTION

AWS SAGEMAKER SECURITY: PROTECTION AT REST USING ENCRYPTION

- **Amazon S3:**
  - S3 buckets for model artifacts and training data can are encrypted

- **AWS Key Management Service (KMS):**
  - Amazon SageMaker notebooks, training jobs, hyperparameter tuning jobs, batch transform jobs, and endpoints are encrypted with KMS Key.

https://www.flickr.com/photos/140988606@N08/27891579948

AWS SAGEMAKER SECURITY: PROTECTION IN TRANSIT USING ENCRYPTION

- SageMaker encrypts machine learning model artifacts in transit and at rest.
- Inter-network data in transit are TLS encrypted.
- Requests to SageMaker API and console are made over a secure (SSL) connection.
- AWS IAM roles are assigned to Amazon SageMaker to provide permissions to access resources for both training and deployment.

https://pixabay.com/illustrations/castle-privacy-policy-security-4512299/

AWS SAGEMAKER SECURITY: INTER-CONTAINER TRAFFIC ENCRYPTION

- When distributed training is performed, ML algorithms transmit data such as model weights.
- Data could be protected by using inter-container traffic encryption.
- Inter-container traffic encryption is enabled via console or API when a training job is set up
- Inter-container traffic encryption will:
  - Increase cost
  - Increase training time (when deep learning is implemented)

https://pixabay.com/illustrations/castle-privacy-policy-security-4512299/

AWS SAGEMAKER SECURITY: PROTECTION USING VPC

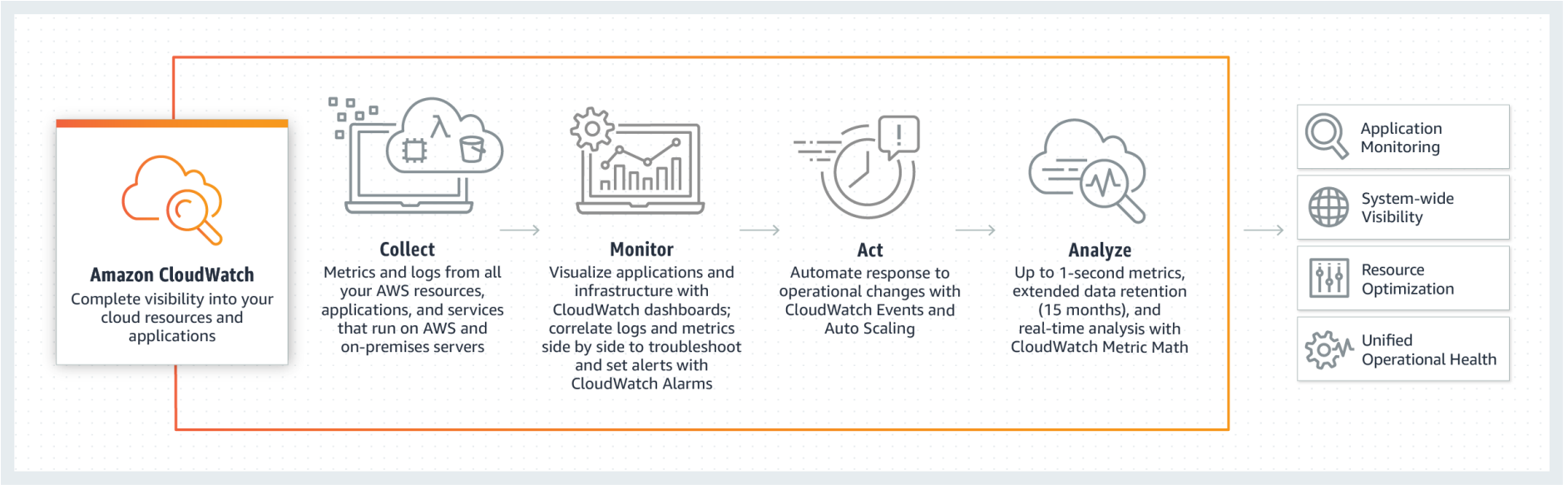
- VPC endpoints increase security by allowing users to access AWS services using AWS network without having to send traffic over the public internet.
- Amazon SageMaker runs training jobs in an Amazon Virtual Private Cloud (VPC) by default to ensure data security.
- Additional level of security could be added (optionally) to protect training containers and data by configuring a private VPC.
- By default, all SageMaker Notebooks are Internet-enabled which might pose a security risk. If disabled, the VPC needs an interface endpoint (PrivateLink) or NAT Gateway, and allow outbound connections, for training and hosting to work.

https://pixabay.com/illustrations/castle-privacy-policy-security-4512299/

AMAZON CLOUDWATCH

AWS SAGEMAKER SECURITY: CLOUDWATCH

- Amazon CloudWatch collects data such as logs, metrics, and events so users can track their applications and take actions accordingly such as resources optimization.
- CloudWatch could be used to detect anomalies, trigger an alarm, visualize metrics/logs, and automatically take actions.



Source: https://aws.amazon.com/cloudwatch/

AWS SAGEMAKER SECURITY: CLOUDWATCH

FULL VISIBILTY AN ALL AWS APPLICATIONS/RESOURCES

- CloudWatch allows for metric and logs collection from all AWS resources which allows for gaining a system-wide visibility.

EASE OF USE

- AWS CloudWatch is super easy to use
- Works with over 70 AWS services such as EC2, S3, Lambda, and DynamoDB.
- Publishes 1-minute metrics and logs (could be enhanced with up to 1-second).

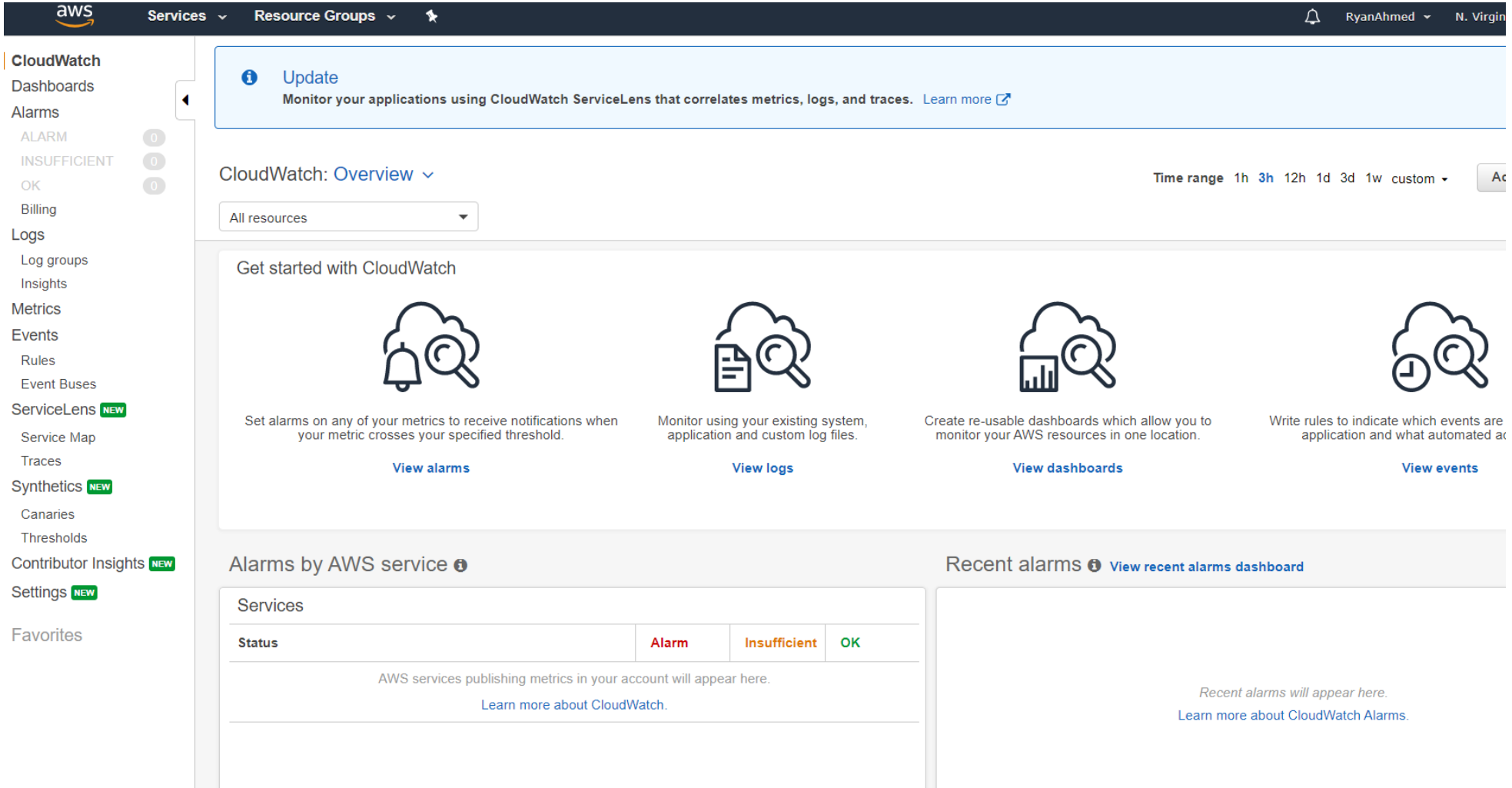
IMPROVED/OPTIMIZED PERFORMANCE

- Amazon CloudWatch allows for alarms to be triggered when a certain threshold is reached.
- Automatic scaling could use CloudWatch to increase or decrease the number of instances and therefore optimize cost.

OBTAIN VALUABLE INSIGHTS

- CloudWatch offers detailed dashboards that show valuable insights.
- Data is retained for 15 months.
- 1-sec granularity.

AWS SAGEMAKER SECURITY: CLOUDWATCH



AMAZON CLOUDTRAIL

AWS SAGEMAKER SECURITY: CLOUDTRAIL FEATURES

ALWAYS ON

- AWS CloudTrail automatically record the activity on every AWS account.
- Data is available for download from the previous 90 days.

HISTORY

- AWS users are able to search account activity history and improve their security process.

WORKS IN MANY REGIONS

- AWS CloudTrail could be configured to track account history from many regions into one Amazon S3 bucket.

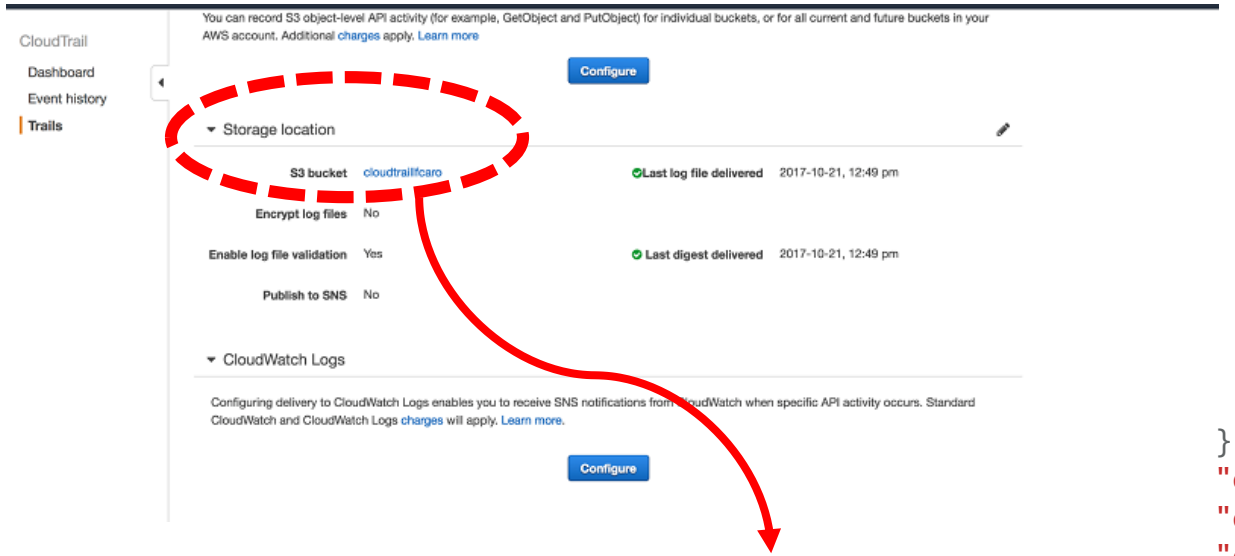
LOG FILE ENCRYPTION

- AWS CloudTrail logs are encrypted by default when they arrive at the Amazon S3 using server-side encryption (SSE).
- AWS Key Management Service (AWS KMS) key could be used for additional security.

QUERY WITH ATHENA

- After the data is available in S3, Athena could be used to query the data and analyze the data

AWS SAGEMAKER SECURITY: CLOUDTRAIL EXAMPLE



SPECIFY S3 BUCKET LOCATION

[Great article: https://aws.amazon.com/blogs/big-data/streamline-aws-cloudtrail-log-visualization-using-aws-glue-and-amazon-quicksight/](https://aws.amazon.com/blogs/big-data/streamline-aws-cloudtrail-log-visualization-using-aws-glue-and-amazon-quicksight/)


```
{ "Records": [{
        "eventVersion": "1.01",
        "userIdentity": {
                "type": "IAMUser",
                "principalId": "AIDAJDPLRKLG7UEXAMPLE",
                "arn": "arn:aws:iam::123456789012:user/Alice",
                "accountId": "123456789012",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "Alice",
                "sessionContext": {
                        "attributes": {
                                "mfaAuthenticated": "false",
                                "creationDate": "2014-03-18T14:29:23Z"
                }
        }
    },
    "eventTime": "2014-03-18T14:30:07Z",
    "eventSource": "cloudtrail.amazonaws.com",
    "eventName": "StartLogging",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "72.21.198.64",
    "userAgent": "signin.amazonaws.com",
    "requestParameters": {
                "name": "Default"
                },
    "responseElements": null,
    "requestID": "cdc73f9d-aea9-11e3-9d5a-835b769c0d9c",
    "eventID": "3074414d-c626-42aa-984b-68ff152d6ab7"
}
```

AWS SAGEMAKER SECURITY: LOGGING AND MONITORING

AMAZON CLOUDWATCH ALARMS

- Used to send an alarm once a certain threshold is exceeded for multiple number of cycles.
- Could be used to ensure the health of training instances

AWS CLOUDTRAIL LOGS

- CloudTrail is used to track activity made by users, roles, or on AWS service.
- CloudTrail provides a record of past requests, IP addresses, timing of the request…etc.