

Disclaimer: These slides are copyrighted and strictly for personal use only

- This document is reserved for people enrolled into the **AWS Certified Machine Learning Engineer Associate Course by Sundog Education.**
- **Please do not share this document**, it is intended for personal use and exam preparation only, thank you.
- If you've obtained these slides for free on a website that is not the course's website, please reach out to piracy@datacumulus.com. Thanks!
- **Best of luck for the exam and happy learning!**

Table of Contents

- [Data Ingestion and Storage](#)
- [Data Transformation, Integrity, and Feature Engineering](#)
- [AWS Managed AI Services](#)
- [SageMaker Built-In Algorithms](#)
- [Model Training, Tuning, and Evaluation](#)
- [Generative AI Model Fundamentals](#)
- [Building Generative AI Applications with Bedrock](#)
- [Machine Learning Operations](#)
- [Security, Identity, and Compliance](#)
- [Management and Governance](#)
- [ML Best Practices](#)
- [Exam Preparation](#)

AWS Certified Machine Learning Engineer - Associate Course MLA-C02

What We'll Cover: Data Ingestion and Storage

- Types, formats, and properties of data
- Data warehouses, lakes, and lakehouses
- ETL Pipelines and Orchestration
- AWS Storage and Streaming Services



Amazon Kinesis



Amazon FSx



Amazon Simple Storage Service (Amazon S3)



Amazon Elastic Block Store (Amazon EBS)



Amazon Elastic File System (Amazon EFS)

Data Transformation, Integrity, and Feature Engineering

- Elastic MapReduce (EMR)
- Handling missing, unbalanced, and outlier data
- Common data transformations
- SageMaker data processing and analysis features
- AWS Glue



Amazon EMR



Amazon SageMaker

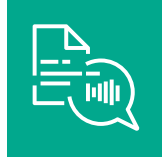


AWS Glue

AWS Managed AI Services



Amazon Personalize



Amazon Polly



Amazon Rekognition



Amazon Q



Amazon Comprehend



Amazon Lookout
for Equipment



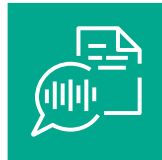
Amazon Forecast



Amazon Lex



Amazon Textract



Amazon Transcribe

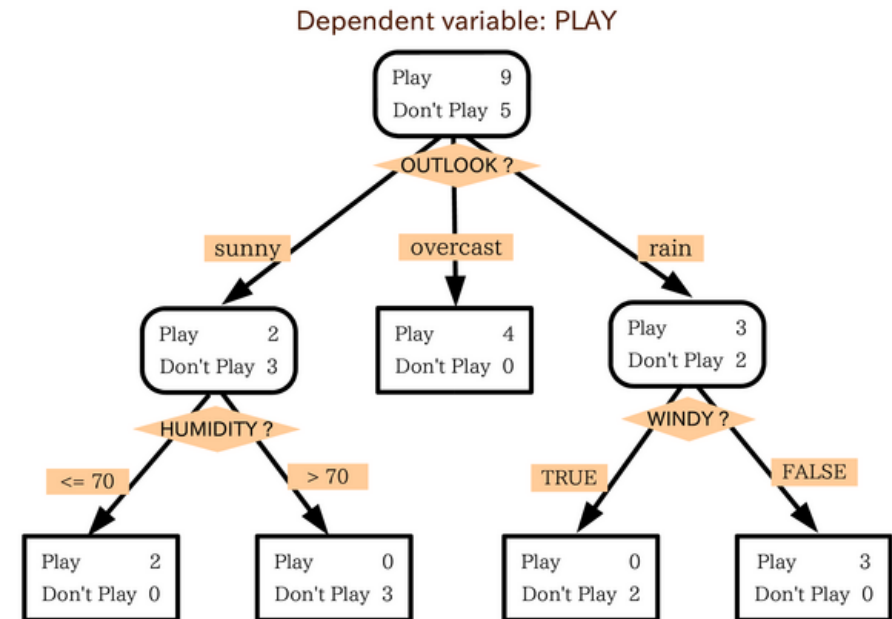
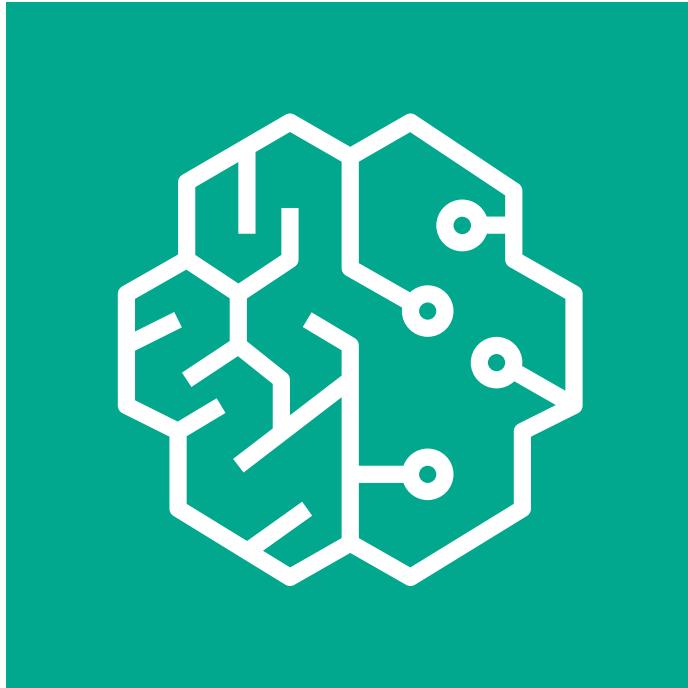


Amazon Translate



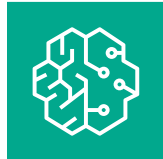
Amazon Fraud Detector

SageMaker AI Built-In Algorithms

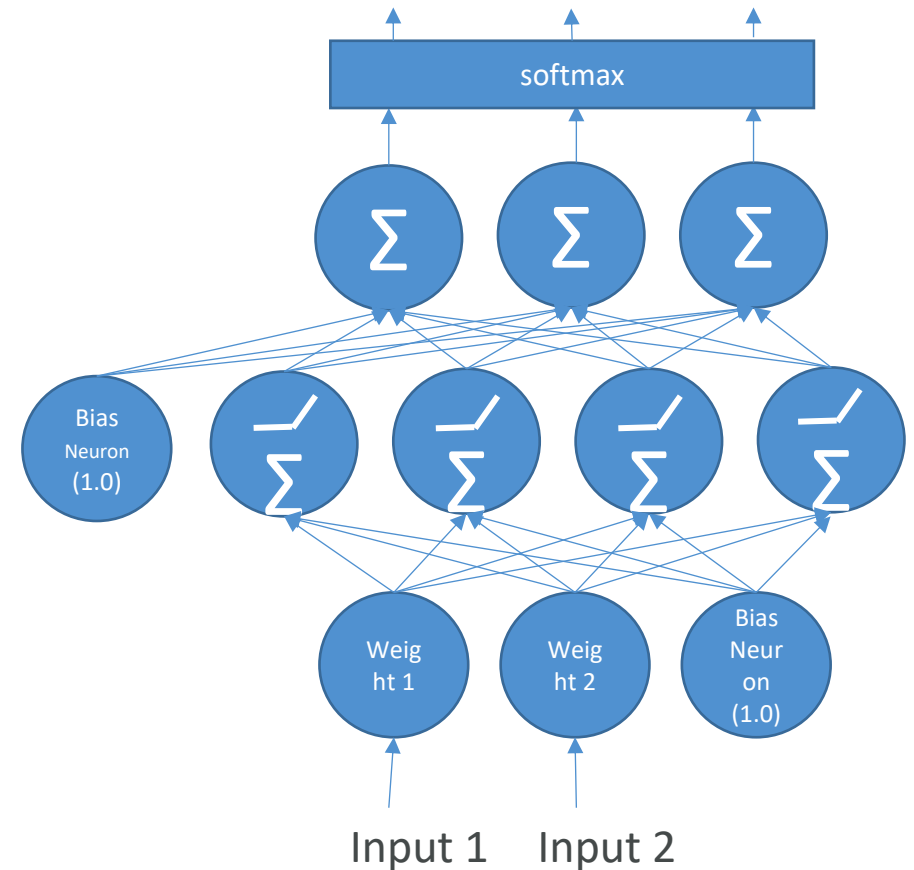


Model Training, Tuning, and Evaluation

- Deep Learning Fundamentals
- Tuning techniques
- Measuring model performance
- Automatic Model Tuning with SageMaker

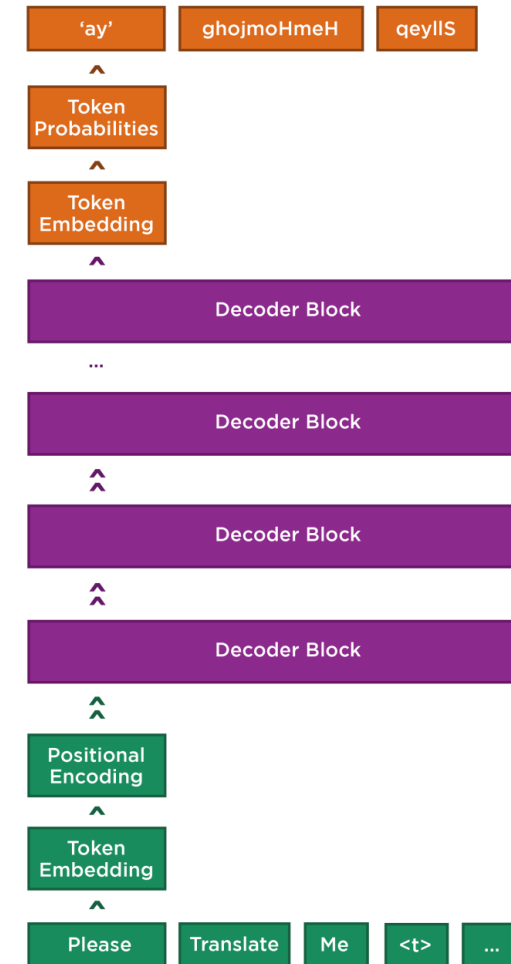


Amazon SageMaker



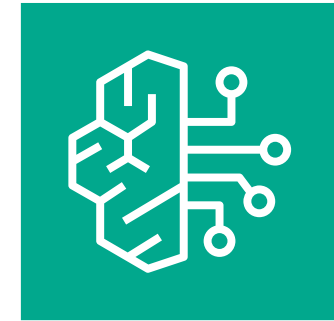
Generative AI Model Fundamentals

- The Transformer Architecture
- Self-Attention
- How GPT Works
- SageMaker JumpStart
- Hands-on Labs



Building Generative AI Applications with Bedrock

- Foundation Models
- Retrieval-Augmented Generation (RAG)
- Knowledge Bases
- Vector Stores
- Guardrails
- LLM Agents
- Lots of hands-on labs



Amazon Bedrock

Machine Learning Operations (MLOps)

- SageMaker AI in depth
- Amazon ECS
- Amazon ECR
- AWS CloudFormation
- AWS CDK
- AWS CodeDeploy
- AWS CodeBuild
- AWS CodePipeline
- Amazon EventBridge
- AWS
- Amazon Managed Workflows for Apache Airflow (MWAA)



Amazon SageMaker



AWS



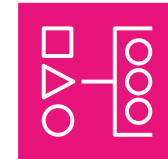
Amazon EventBridge



Amazon Elastic Container Registry (Amazon ECR)



Amazon Elastic Container Service (Amazon ECS)



Amazon Managed Workflows for Apache Airflow (Amazon MWAA)



AWS CodeBuild



AWS Cloud Development Kit (AWS CDK)



AWS CodePipeline



AWS CodeDeploy

Security, Identity, and Compliance

- Securing data in SageMaker
- AWS IAM
- KMS
- Macie
- Secrets Manager
- WAF
- Shield
- VPC
- PrivateLink



AWS Identity and Access Management (IAM)



AWS Key Management Service (AWS KMS)



AWS Secrets Manager



AWS Shield



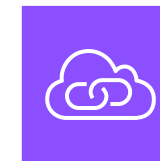
Amazon Macie



AWS WAF



Amazon Virtual Private Cloud (Amazon VPC)



AWS PrivateLink

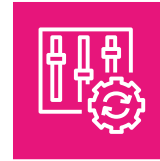
Management and Governance



Amazon CloudWatch



AWS CloudFormation



AWS Config



AWS CloudTrail



AWS X-Ray



AWS Trusted Advisor

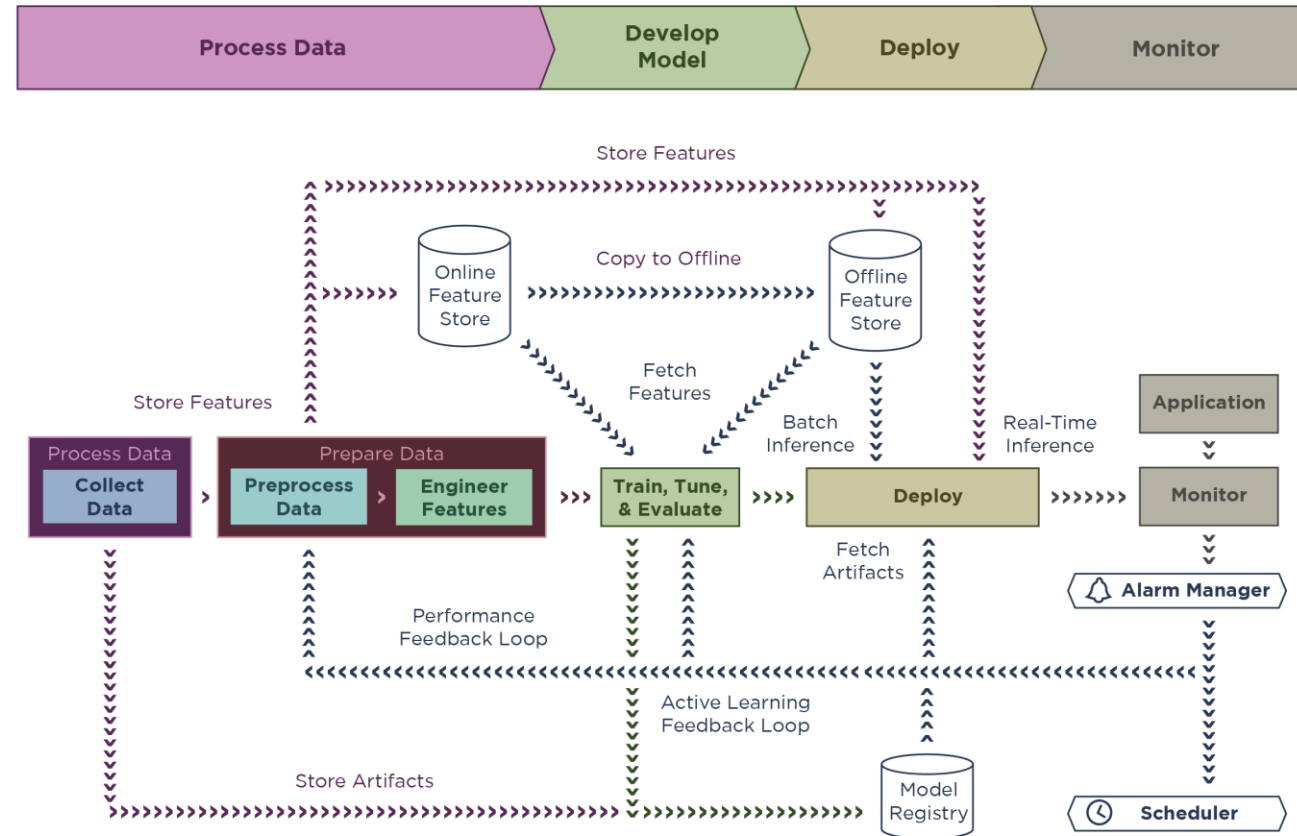


AWS Budgets



AWS Cost Explorer

AWS Well-Architected Machine Learning Lens



Target Candidate

- 1 Year experience with SageMaker AI, Bedrock, and other ML / AI Engineering services
- 1 year in software dev, devops, data engineering, or data scientist
- ML and AI algorithms
- Data engineering fundamentals
- Software engineering and CI/CD
- NOT required:
 - Designing end-to-end ML/AI solutions
 - Depth in NLP, Computer Vision, etc.



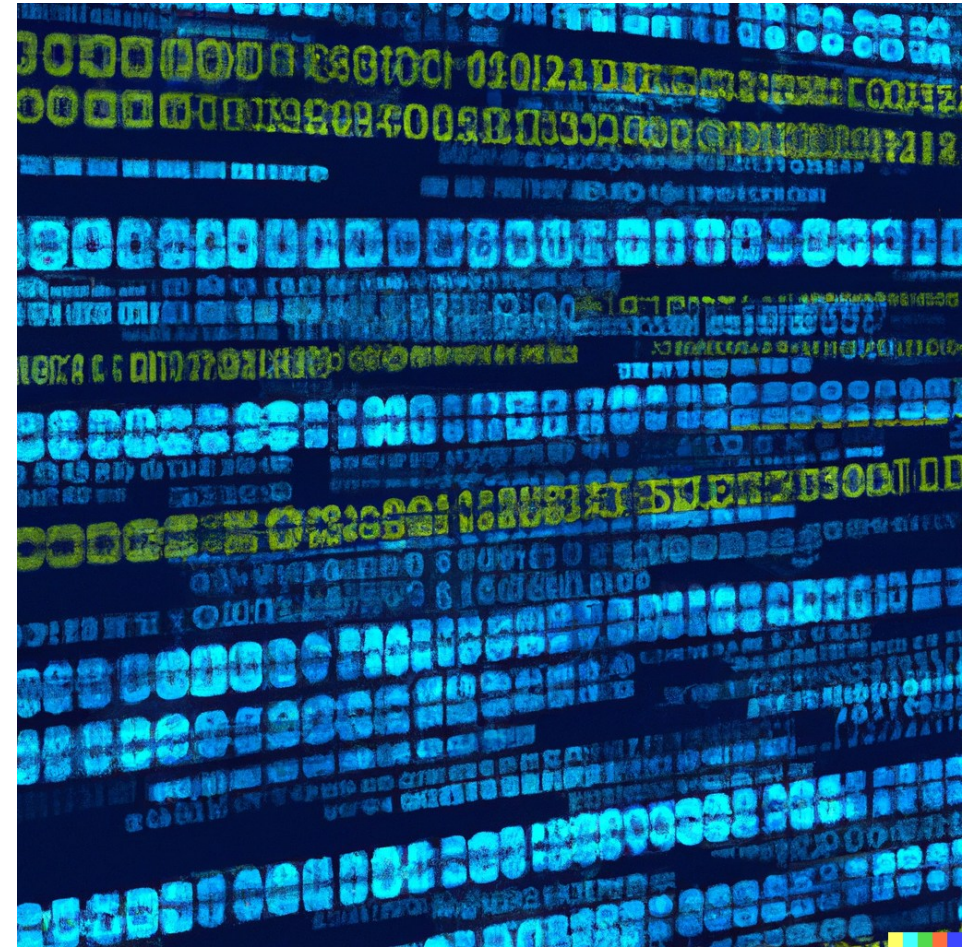


Data Ingestion and Storage

Moving, Storing and Processing data in AWS

Types of Data

- Structured
- Unstructured
- Semi-Structured



Structured Data

- Definition: Data that is organized in a defined manner or schema, typically found in relational databases.
- Characteristics:
 - Easily queryable
 - Organized in rows and columns
 - Has a consistent structure
- Examples:
 - Database tables
 - CSV files with consistent columns
 - Excel spreadsheets



Unstructured Data

- Definition: Data that doesn't have a predefined structure or schema.
- Characteristics:
 - Not easily queryable without preprocessing
 - May come in various formats
- Examples:
 - Text files without a fixed format
 - Videos and audio files
 - Images
 - Emails and word processing documents



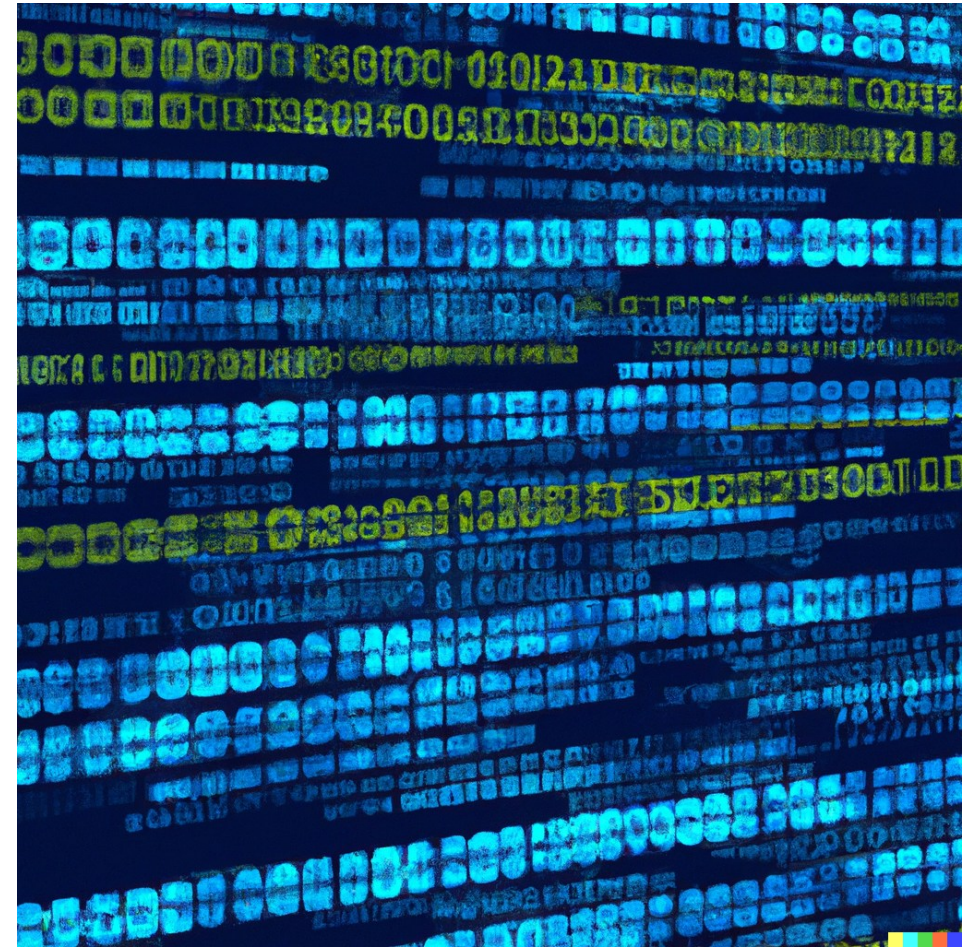
Semi-Structured Data

- Definition: Data that is not as organized as structured data but has some level of structure in the form of tags, hierarchies, or other patterns.
- Characteristics:
 - Elements might be tagged or categorized in some way
 - More flexible than structured data but not as chaotic as unstructured data
- Examples:
 - XML and JSON files
 - Email headers (which have a mix of structured fields like date, subject, etc., and unstructured data in the body)
 - Log files with varied formats



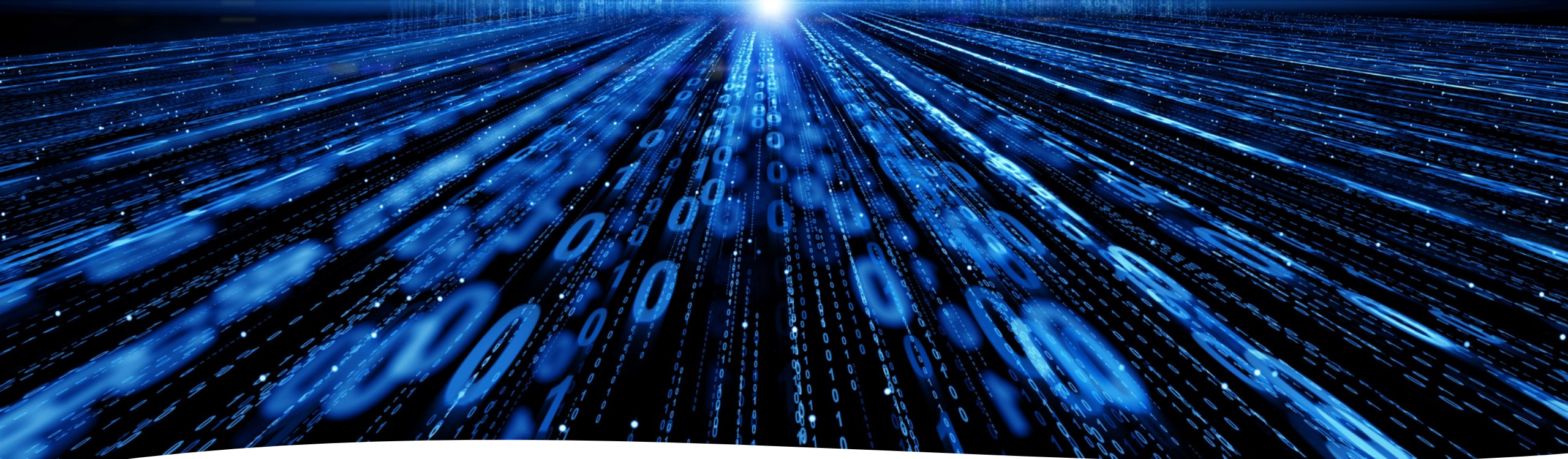
Properties of Data

- Volume
- Velocity
- Variety



Volume

- Definition: Refers to the amount or size of data that organizations are dealing with at any given time.
- Characteristics:
 - May range from gigabytes to petabytes or even more
 - Challenges in storing, processing, and analyzing high volumes of data
- Examples:
 - A popular social media platform processing terabytes of data daily from user posts, images, and videos.
 - Retailers collecting years' worth of transaction data, amounting to several petabytes.



Velocity

- Definition: Refers to the speed at which new data is generated, collected, and processed.
- Characteristics:
 - High velocity requires real-time or near-real-time processing capabilities
 - Rapid ingestion and processing can be critical for certain applications
- Examples:
 - Sensor data from IoT devices streaming readings every millisecond.
 - High-frequency trading systems where milliseconds can make a difference in decision-making.



Variety

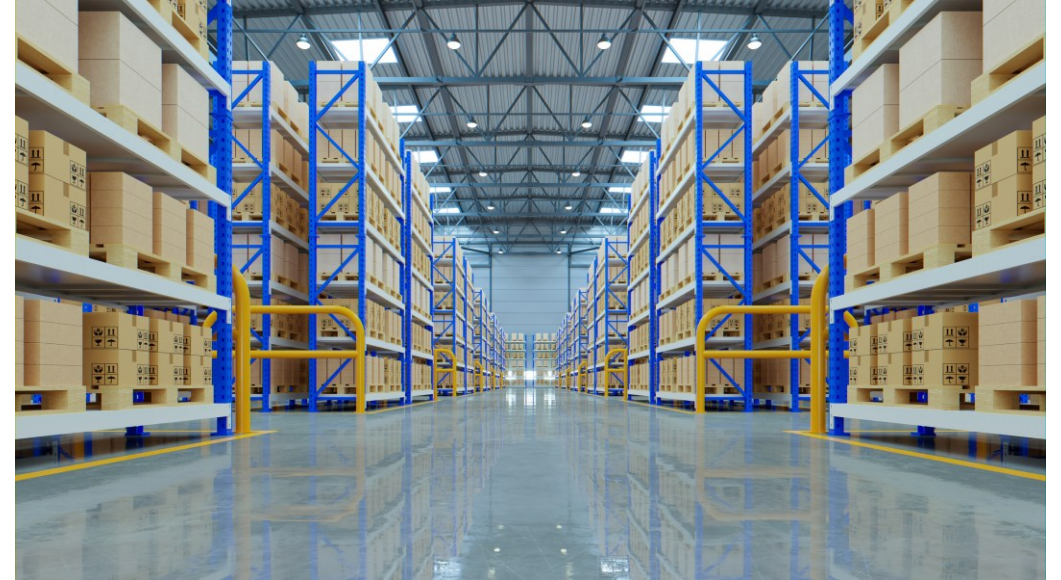
- Definition: Refers to the different types, structures, and sources of data.
- Characteristics:
 - Data can be structured, semi-structured, or unstructured
 - Data can come from multiple sources and in various formats
- Examples:
 - A business analyzing data from relational databases (structured), emails (unstructured), and JSON logs (semi-structured).
 - Healthcare systems collecting data from electronic medical records, wearable health devices, and patient feedback forms.



Data Warehouses vs. Data Lakes

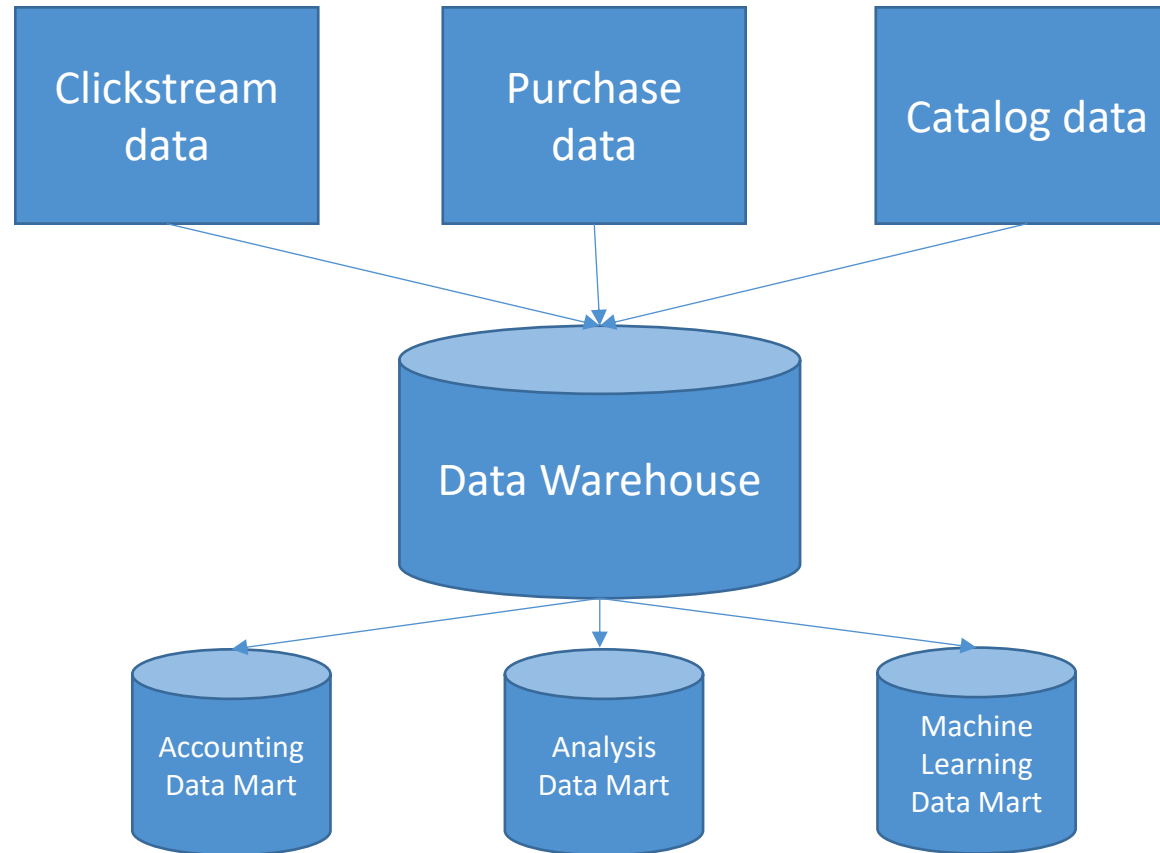
Data Warehouse

- Definition: A centralized repository optimized for analysis where data from different sources is stored in a structured format.
- Characteristics:
 - Designed for complex queries and analysis
 - Data is cleaned, transformed, and loaded (ETL process)
 - Typically uses a star or snowflake schema
 - Optimized for read-heavy operations
- Examples:
 - Amazon Redshift
 - Google BigQuery
 - Microsoft Azure SQL Data Warehouse



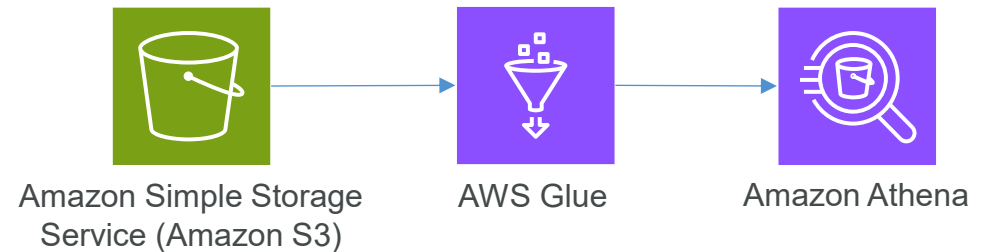
Amazon Redshift

Data Warehouse example



Data Lake

- Definition: A storage repository that holds vast amounts of raw data in its native format, including structured, semi-structured, and unstructured data.
- Characteristics:
 - Can store large volumes of raw data without predefined schema
 - Data is loaded as-is, no need for preprocessing
 - Supports batch, real-time, and stream processing
 - Can be queried for data transformation or exploration purposes
- Examples:
 - Amazon Simple Storage Service (S3) when used as a data lake
 - Azure Data Lake Storage
 - Hadoop Distributed File System (HDFS)



Comparing the two

- **Schema:**
 - Data Warehouse: Schema-on-write (predefined schema before writing data)
 - Extract – Transform – Load (**ETL**)
 - Data Lake: Schema-on-read (schema is defined at the time of reading data)
 - Extract – Load – Transform (**ELT**)
- **Data Types:**
 - Data Warehouse: Primarily structured data
 - Data Lake: Both structured and unstructured data
- **Agility:**
 - Data Warehouse: Less agile due to predefined schema
 - Data Lake: More agile as it accepts raw data without a predefined structure
- **Processing:**
 - Data Warehouse: ETL (Extract, Transform, Load)
 - Data Lake: ELT (Extract, Load, Transform) or just Load for storage purposes
- **Cost:**
 - Data Warehouse: Typically more expensive because of optimizations for complex queries
 - Data Lake: Cost-effective storage solutions, but costs can rise when processing large amounts of data

Choosing a Warehouse vs. a Lake

- **Use a Data Warehouse when:**
 - You have structured data sources and require fast and complex queries.
 - Data integration from different sources is essential.
 - Business intelligence and analytics are the primary use cases.
- **Use a Data Lake when:**
 - You have a mix of structured, semi-structured, or unstructured data.
 - You need a scalable and cost-effective solution to store massive amounts of data.
 - Future needs for data are uncertain, and you want flexibility in storage and processing.
 - Advanced analytics, machine learning, or data discovery are key goals.
- Often, organizations use a combination of both, ingesting raw data into a data lake and then processing and moving refined data into a data warehouse for analysis.



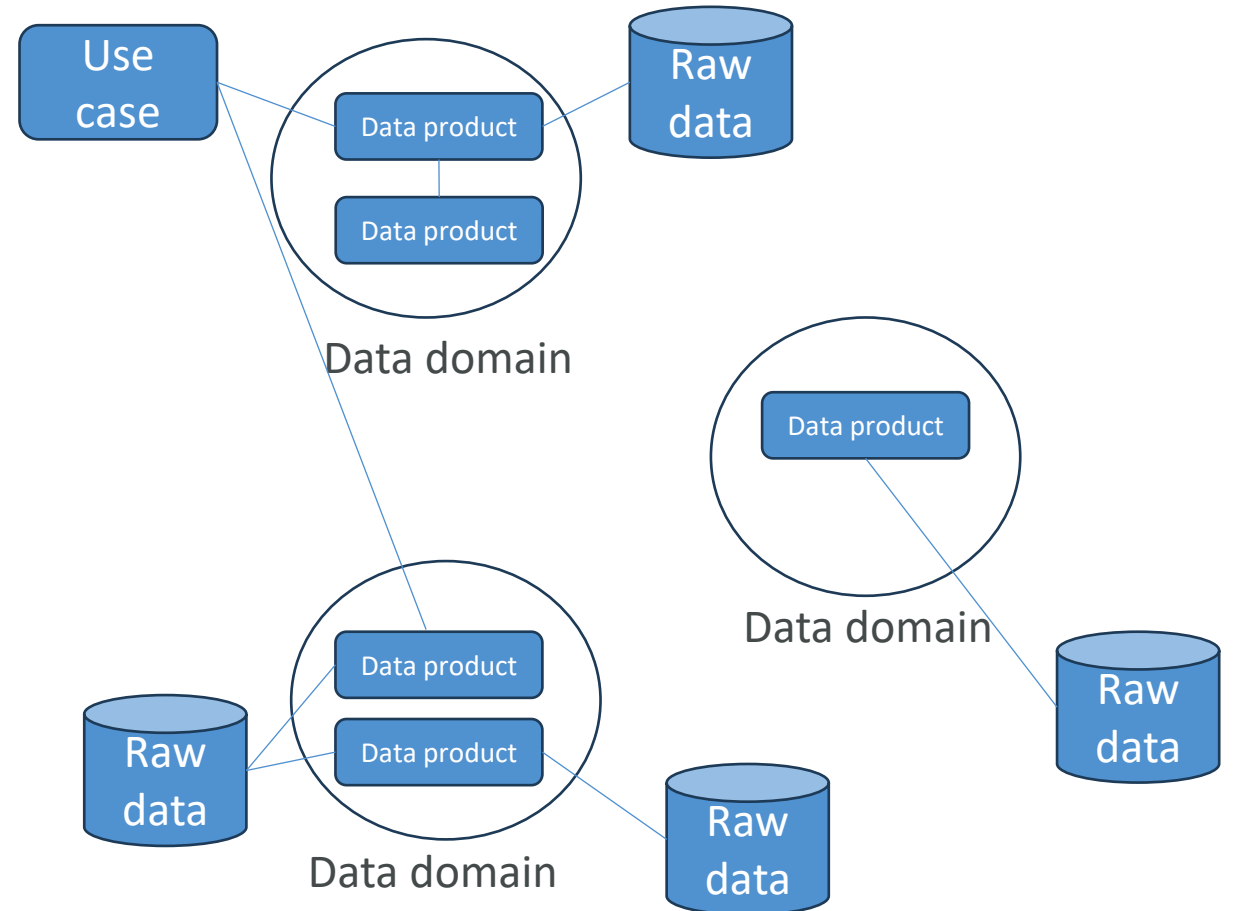
Data Lakehouse

- Definition: A hybrid data architecture that combines the best features of data lakes and data warehouses, aiming to provide the performance, reliability, and capabilities of a data warehouse while maintaining the flexibility, scale, and low-cost storage of data lakes.
- Characteristics:
 - Supports both structured and unstructured data.
 - Allows for schema-on-write and schema-on-read.
 - Provides capabilities for both detailed analytics and machine learning tasks.
 - Typically built on top of cloud or distributed architectures.
 - Benefits from technologies like Delta Lake, which bring ACID transactions to big data.
- Examples:
 - **AWS Lake Formation (with S3 and Redshift Spectrum)**
 - **Delta Lake:** An open-source storage layer that brings ACID transactions to Apache Spark and big data workloads.
 - **Databricks Lakehouse Platform:** A unified platform that combines the capabilities of data lakes and data warehouses.
 - **Azure Synapse Analytics:** Microsoft's analytics service that brings together big data and data warehousing.



Data Mesh

- Coined in 2019; it's more about governance and organization
- Individual teams own “data products” within a given domain
- These data products serve various “use cases” around the organization
- “Domain-based data management”
- Federated governance with central standards
- Self-service tooling & infrastructure
- Data lakes, warehouses, etc. may be part of it
 - But a “data mesh” is more about the “data management paradigm” and not the specific technologies or architectures.



ETL Pipelines

- **Definition:** ETL stands for Extract, Transform, Load. It's a process used to move data from source systems into a data warehouse.
- **Extract:**
 - Retrieve raw data from source systems, which can be databases, CRMs, flat files, APIs, or other data repositories.
 - Ensure data integrity during the extraction phase.
 - Can be done in real-time or in batches, depending on requirements.

ETL Pipelines: Transform

- Convert the extracted data into a format suitable for the target data warehouse.
- Can involve various operations such as:
 - Data cleansing (e.g., removing duplicates, fixing errors)
 - Data enrichment (e.g., adding additional data from other sources)
 - Format changes (e.g., date formatting, string manipulation)
 - Aggregations or computations (e.g., calculating totals or averages)
 - Encoding or decoding data
 - Handling missing values

ETL Pipelines: Load

- Move the transformed data into the target data warehouse or another data repository.
- Can be done in batches (all at once) or in a streaming manner (as data becomes available).
- Ensure that data maintains its integrity during the loading phase.



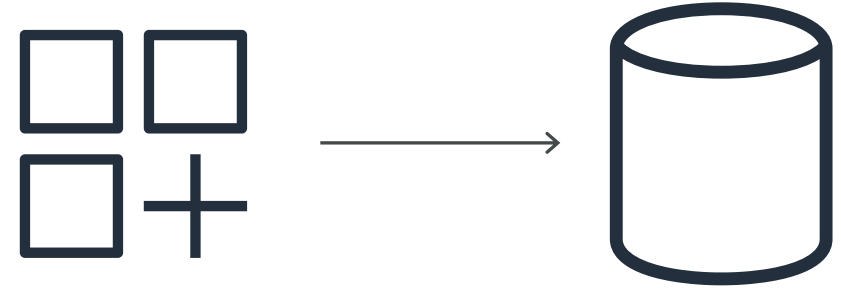
Managing ETL Pipelines

- This process must be automated in some reliable way
- AWS Glue
- Orchestration services
 - EventBridge
 - Amazon Managed Workflows for Apache Airflow [Amazon MWAA]
 - AWS
 - Lambda
 - Glue Workflows
- We'll get into specific architectures later.



Data Sources

- JDBC
 - Java Database Connectivity
 - Platform-independent
 - Language-dependent
- ODBC
 - Open Database Connectivity
 - Platform-dependent (thx to drivers)
 - Language-independent
- Raw logs
- API's
- Streams



CSV (Comma-Separated Values)

- **Description:** Text-based format that represents data in a tabular form where each line corresponds to a row and values within a row are separated by commas.
- **When to Use:**
 - For small to medium datasets.
 - For data interchange between systems with different technologies.
 - For human-readable and editable data storage.
 - Importing/Exporting data from databases or spreadsheets.
- **Systems:** Databases (SQL-based), Excel, Pandas in Python, R, many ETL tools.

```
InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
536365,85123A,WHITE HANGING HEART T-LIGHT HOLDER,6,12/1/2010,8:26,2.55,17850,United Kingdom
536365,71053,WHITE METAL LANTERN,6,12/1/2010,8:26,3.39,17850,United Kingdom
536365,84406B,CREAM CUPID HEARTS COAT HANGER,8,12/1/2010,8:26,2.75,17850,United Kingdom
536365,84029G,KNITTED UNION FLAG HOT WATER BOTTLE,6,12/1/2010,8:26,3.39,17850,United Kingdom
536365,84029E,RED WOOLLY HOTTIE WHITE HEART.,6,12/1/2010,8:26,3.39,17850,United Kingdom
536365,22752,SET 7 BABUSHKA NESTING BOXES,2,12/1/2010,8:26,7.65,17850,United Kingdom
536365,21730,GLASS STAR FROSTED T-LIGHT HOLDER,6,12/1/2010,8:26,4.25,17850,United Kingdom
536366,22633,HAND WARMER UNION JACK,6,12/1/2010,8:28,1.85,17850,United Kingdom
536366,22632,HAND WARMER RED POLKA DOT,6,12/1/2010,8:28,1.85,17850,United Kingdom
536367,84879,ASSORTED COLOUR BIRD ORNAMENT,32,12/1/2010,8:34,1.69,13047,United Kingdom
536367,22745,POPPY'S PLAYHOUSE BEDROOM ,6,12/1/2010,8:34,2.1,13047,United Kingdom
536367,22748,POPPY'S PLAYHOUSE KITCHEN,6,12/1/2010,8:34,2.1,13047,United Kingdom
536367,22749,FELTCRAFT PRINCESS CHARLOTTE DOLL,8,12/1/2010,8:34,3.75,13047,United Kingdom
536367,22310,IVORY KNITTED MUG COSY ,6,12/1/2010,8:34,1.65,13047,United Kingdom
536367,84969,BOX OF 6 ASSORTED COLOUR TEASPOONS,6,12/1/2010,8:34,4.25,13047,United Kingdom
536367,22623,BOX OF VINTAGE JIGSAW BLOCKS ,3,12/1/2010,8:34,4.95,13047,United Kingdom
536367,22622,BOX OF VINTAGE ALPHABET BLOCKS,2,12/1/2010,8:34,9.95,13047,United Kingdom
536367,21754,HOME BUILDING BLOCK WORD,3,12/1/2010,8:34,5.95,13047,United Kingdom
536367,21755,LOVE BUILDING BLOCK WORD,3,12/1/2010,8:34,5.95,13047,United Kingdom
536367,21777,RECIPE BOX WITH METAL HEART,4,12/1/2010
```

JSON (JavaScript Object Notation)

- **Description:** Lightweight, text-based, and human-readable data interchange format that represents structured or semi-structured data based on key-value pairs.
- **When to Use:**
 - Data interchange between a web server and a web client.
 - Configurations and settings for software applications.
 - Use cases that need a flexible schema or nested data structures.
- **Systems:** Web browsers, many programming languages (like JavaScript, Python, Java, etc.), RESTful APIs, NoSQL databases (like MongoDB).

```
{
  "messages": [
    {
      "role": "system",
      "content": "Data is an android in the TV series Star Trek: The Next Generation."
    },
    {
      "role": "user",
      "content": "PICARD: You will agree, Data, that Starfleet's instructions are difficult?"
    },
    {
      "role": "assistant",
      "content": "DATA: Difficult ... how so? Simply solve the mystery of Farpoint Station."
    }
  ]
}
```

```
{
  "messages": [
    {
      "role": "system",
      "content": "Data is an android in the TV series Star Trek: The Next Generation."
    },
    {
      "role": "user",
      "content": "PICARD: The problem, Data, is that another life form built that base. How do I negotiate a friendly agreement for Starfleet to use it while at the same time snoop around finding how and why they built it?"
    },
    {
      "role": "assistant",
      "content": "DATA: Inquiry ... the word snoop ... ?"
    }
  ]
}
```

```
{
  "messages": [
    {
      "role": "system",
      "content": "Data is an android in the TV series Star Trek: The Next Generation."
    },
    {
      "role": "user",
      "content": "PICARD: Data, how can you be programmed as a virtual encyclopedia of human information without knowing a simple word like snoop?"
    },
    {
      "role": "assistant",
      "content": "DATA: Possibility ... a kind of human behavior I was not designed to emulate?"
    }
  ]
}
```

```
{
  "messages": [
    {
      "role": "system",
      "content": "Data is an android in the TV series Star Trek: The Next Generation."
    },
    {
      "role": "user",
      "content": "PICARD: It means 'to spy, to sneak' ..."
    },
    {
      "role": "assistant",
      "content": "DATA: Ah! To seek covertly, to go stealthily, to slink, slither ..."
    }
  ]
}
```

```
{
  "messages": [
    {
      "role": "system",
      "content": "Data is an android in the TV series Star Trek: The Next Generation."
    },
    {
      "role": "user",
      "content": "PICARD: Exactly, yes ..."
    },
    {
      "role": "assistant",
      "content": "DATA: ... to glide, creep, skulk, pussyfoot, gumshoe ..."
    }
  ]
}
```

```
{
  "messages": [
    {
      "role": "system",
      "content": "Data is an android in the TV series Star Trek: The Next Generation."
    },
    {
      "role": "user",
      "content": "WORF: Something strange on the detector circuits ..."
    },
    {
      "role": "assistant",
      "content": "DATA: It registers as solid, Captain ..."
    }
  ]
}
```

```
{
  "messages": [
    {
      "role": "system",
      "content": "Data is an android in the TV series Star Trek: The Next Generation."
    },
    {
      "role": "user",
      "content": "PICARD: Records search, Data. Results of detaching the Saucer Section at high warp speeds."
    },
    {
      "role": "assistant",
      "content": "DATA: Inadvisable at any warp speed,"
    }
  ]
}
```

Avro

- **Description:** Binary format that stores both the data and its schema, allowing it to be processed later with different systems without needing the original system's context.
- **When to Use:**
 - With big data and real-time processing systems.
 - When schema evolution (changes in data structure) is needed.
 - Efficient serialization for data transport between systems.
- **Systems:** Apache Kafka, Apache Spark, Apache Flink, Hadoop ecosystem.

Parquet

- **Description:** Columnar storage format optimized for analytics. Allows for efficient compression and encoding schemes.
- **When to Use:**
 - Analyzing large datasets with analytics engines.
 - Use cases where reading specific columns instead of entire records is beneficial.
 - Storing data on distributed systems where I/O operations and storage need optimization.
- **Systems:** Hadoop ecosystem, Apache Spark, Apache Hive, Apache Impala, Amazon Redshift Spectrum.

Amazon S3 Section

Section introduction



- Amazon S3 is one of the main building blocks of AWS
- It's advertised as "infinitely scaling" storage
- Many websites use Amazon S3 as a backbone
- Many AWS services use Amazon S3 as an integration as well
- We'll have a step-by-step approach to S3

Amazon S3 Use cases

- Backup and storage
- Disaster Recovery
- Archive
- Hybrid Cloud storage
- Application hosting
- Media hosting
- Data lakes & big data analytics
- Software delivery
- Static website



Nasdaq stores 7 years of data into S3 Glacier



Sysco runs analytics on its data and gain business insights

Amazon S3 - Buckets

- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets are defined at the region level
- S3 looks like a global service but buckets are created in a region
- Naming:
 - **Shared Global Namespace** – have a **globally unique name (across all regions all accounts)**
 - **Account Regional Namespace** – allows for “reuse” of the same bucket name across regions
- Naming constraints:
 - No uppercase, No underscore
 - Not an IP
 - Must start with lowercase letter or number
 - Must NOT start with the prefix **xn--**
 - Must NOT end with the suffix **-s3alias**



S3 Bucket

Amazon S3 - Objects

- Objects (files) have a Key
- The **key** is the **FULL** path:
 - s3://my-bucket/**my_file.txt**
 - s3://my-bucket/**my_folder1/another_folder/my_file.txt**
- The key is composed of **prefix** + **object name**
 - s3://my-bucket/**my_folder1/another_folder/my_file.txt**
- There's no concept of “directories” within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes (“/”)



Object



S3 Bucket
with Objects

Amazon S3 – Objects (cont.)



- Object values are the content of the body:
 - Max. Object Size is 50TB (50,000GB)
 - If uploading more than 5GB, must use “multi-part upload”
- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

Amazon S3 – Security

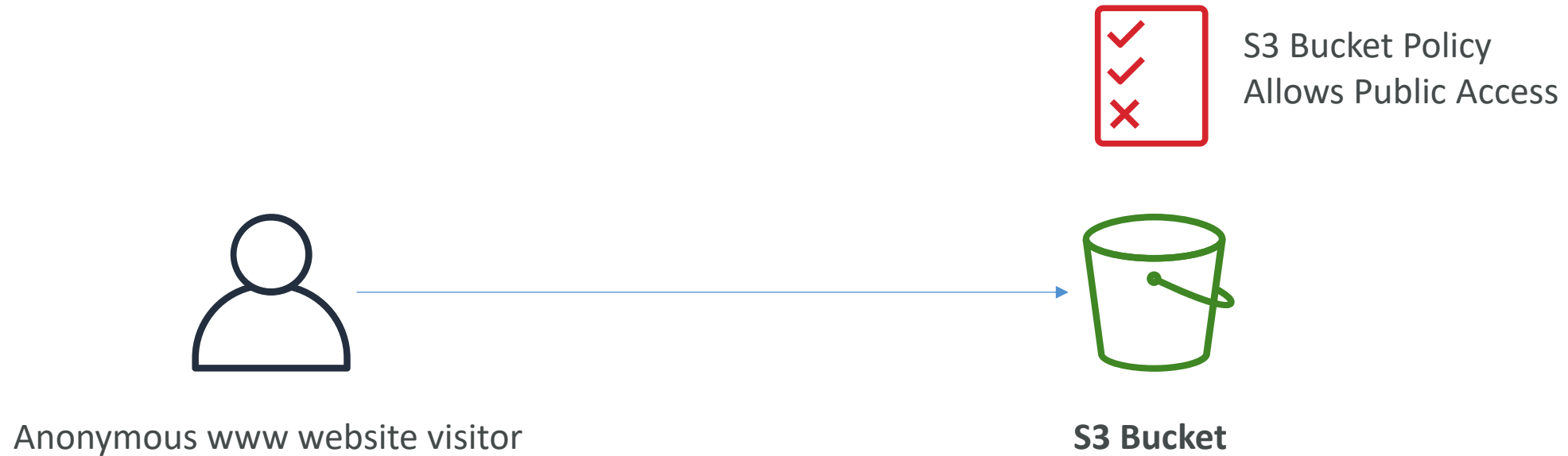
- **User-Based**
 - **IAM Policies** – which API calls should be allowed for a specific user from IAM
- **Resource-Based**
 - **Bucket Policies** – bucket wide rules from the S3 console - allows cross account
 - **Object Access Control List (ACL)** – finer grain (can be disabled)
 - **Bucket Access Control List (ACL)** – less common (can be disabled)
- **Note:** an IAM principal can access an S3 object if
 - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
 - AND there's no explicit DENY
- **Encryption:** encrypt objects in Amazon S3 using encryption keys

S3 Bucket Policies

- JSON based policies
 - Resources: buckets and objects
 - Effect: Allow / Deny
 - Actions: Set of API to Allow or Deny
 - Principal: The account or user to apply the policy to
- Use S3 bucket for policy to:
 - Grant public access to the bucket
 - Force objects to be encrypted at upload
 - Grant access to another account (Cross Account)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
      ]
    }
  ]
}
```

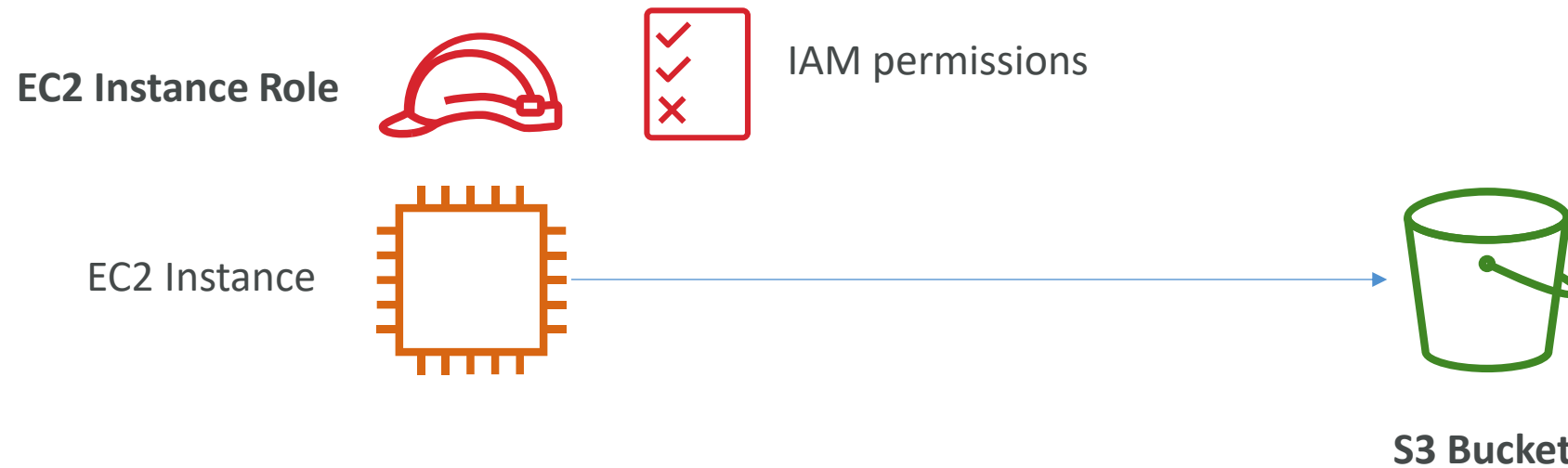
Example: Public Access - Use Bucket Policy



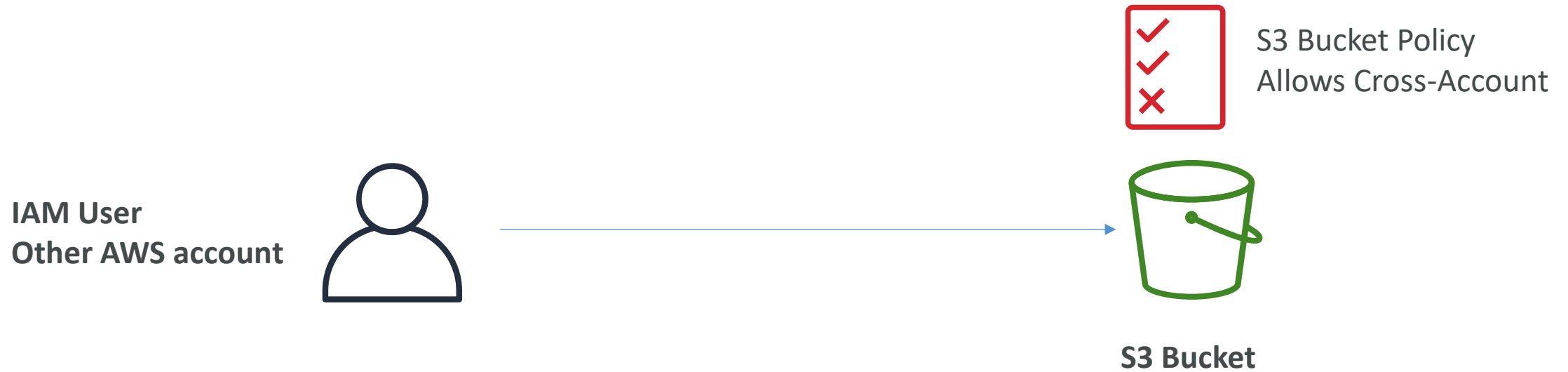
Example: User Access to S3 – IAM permissions



Example: EC2 instance access - Use IAM Roles



Advanced: Cross-Account Access – Use Bucket Policy



Bucket settings for Block Public Access

Block *all* public access

On

Block public access to buckets and objects granted through *new* access control lists (ACLs)

On

Block public access to buckets and objects granted through *any* access control lists (ACLs)

On

Block public access to buckets and objects granted through *new* public bucket or access point policies

On

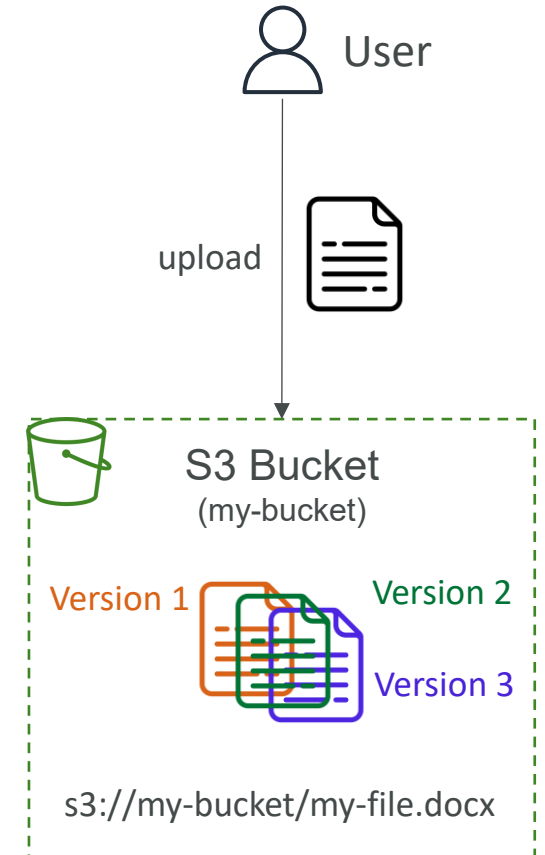
Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

On

- **These settings were created to prevent company data leaks**
- If you know your bucket should never be public, leave these on
- Can be set at the account level

Amazon S3 - Versioning

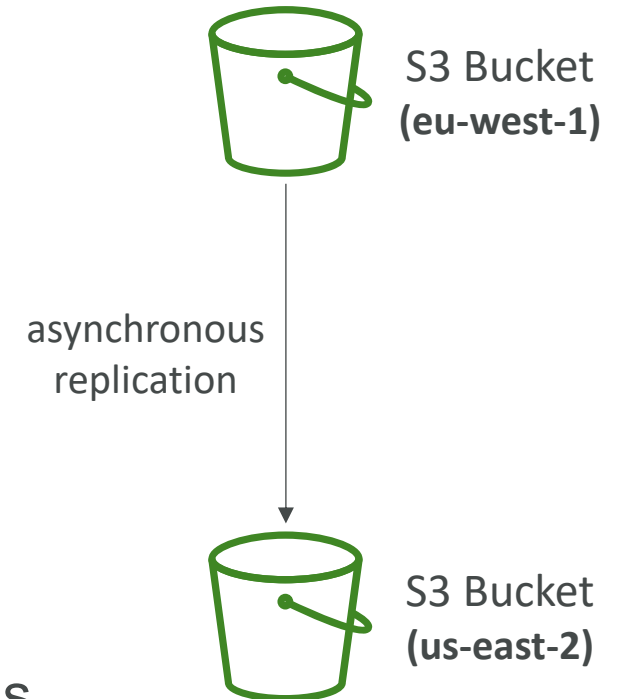
- You can version your files in Amazon S3
- It is enabled at the **bucket level**
- Same key overwrite will change the “version”: 1, 2, 3....
- It is best practice to version your buckets
 - Protect against unintended deletes (ability to restore a version)
 - Easy roll back to previous version
- Notes:
 - Any file that is not versioned prior to enabling versioning will have version “null”
 - Suspending versioning does not delete the previous



Amazon S3 – Replication (CRR & SRR)



- **Must enable Versioning** in source and destination buckets
- **Cross-Region Replication (CRR)**
- **Same-Region Replication (SRR)**
- Buckets can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3
- Use cases:
 - **CRR** – compliance, lower latency access, replication across accounts
 - **SRR** – log aggregation, live replication between production and test accounts



Amazon S3 – Replication (Notes)

- After you enable Replication, only new objects are replicated
- Optionally, you can replicate existing objects using **S3 Batch Replication**
 - Replicates existing objects and objects that failed replication
- For DELETE operations
 - **Can replicate delete markers** from source to target (optional setting)
 - Deletions with a version ID are not replicated (to avoid malicious deletes)
- **There is no “chaining” of replication**
 - If bucket 1 has replication into bucket 2, which has replication into bucket 3
 - Then objects created in bucket 1 are not replicated to bucket 3

S3 Storage Classes

- Amazon S3 Standard - General Purpose
 - Amazon S3 Standard-Infrequent Access (IA)
 - Amazon S3 One Zone-Infrequent Access
 - Amazon S3 Glacier Instant Retrieval
 - Amazon S3 Glacier Flexible Retrieval
 - Amazon S3 Glacier Deep Archive
 - Amazon S3 Intelligent Tiering
-
- Can move between classes manually or using S3 Lifecycle configurations

S3 Durability and Availability

- Durability:
 - High durability (99.9999999999%, 11 9's) of objects across multiple AZ
 - If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
 - Same for all storage classes
- Availability:
 - Measures how readily available a service is
 - Varies depending on storage class
 - Example: S3 standard has 99.99% availability = not available 53 minutes a year

S3 Standard – General Purpose



- 99.99% Availability
 - Used for frequently accessed data
 - Low latency and high throughput
 - Sustain 2 concurrent facility failures
-
- Use Cases: Big Data analytics, mobile & gaming applications, content distribution...

S3 Storage Classes – Infrequent Access

- For data that is less frequently accessed, but requires rapid access when needed
- Lower cost than S3 Standard
- **Amazon S3 Standard-Infrequent Access (S3 Standard-IA)**
 - 99.9% Availability
 - Use cases: Disaster Recovery, backups
- **Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)**
 - High durability (99.999999999%) in a single AZ; data lost when AZ is destroyed
 - 99.5% Availability
 - Use Cases: Storing secondary backup copies of on-premises data, or data you can recreate



Amazon S3 Glacier Storage Classes

- Low-cost object storage meant for archiving / backup
- Pricing: price for storage + object retrieval cost
- **Amazon S3 Glacier Instant Retrieval**
 - Millisecond retrieval, great for data accessed once a quarter
 - Minimum storage duration of 90 days
- **Amazon S3 Glacier Flexible Retrieval** (formerly Amazon S3 Glacier):
 - Expedited (1 to 5 minutes), Standard (3 to 5 hours), Bulk (5 to 12 hours) – free
 - Minimum storage duration of 90 days
- **Amazon S3 Glacier Deep Archive – for long term storage:**
 - Standard (12 hours), Bulk (48 hours)
 - Minimum storage duration of 180 days



S3 Intelligent-Tiering



- Small monthly monitoring and auto-tiering fee
- Moves objects automatically between Access Tiers based on usage
- There are no retrieval charges in S3 Intelligent-Tiering
- *Frequent Access tier (automatic)*: default tier
- *Infrequent Access tier (automatic)*: objects not accessed for 30 days
- *Archive Instant Access tier (automatic)*: objects not accessed for 90 days
- *Archive Access tier (optional)*: configurable from 90 days to 700+ days
- *Deep Archive Access tier (optional)*: config. from 180 days to

S3 Storage Classes Comparison

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 3	>= 3	>= 3	1	>= 3	>= 3	>= 3
Min. Storage Duration Charge	None	None	30 Days	30 Days	90 Days	90 Days	180 Days
Min. Billable Object Size	None	None	128 KB	128 KB	128 KB	40 KB	40 KB
Retrieval Fee	None	None	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved

S3 Storage Classes – Price Comparison

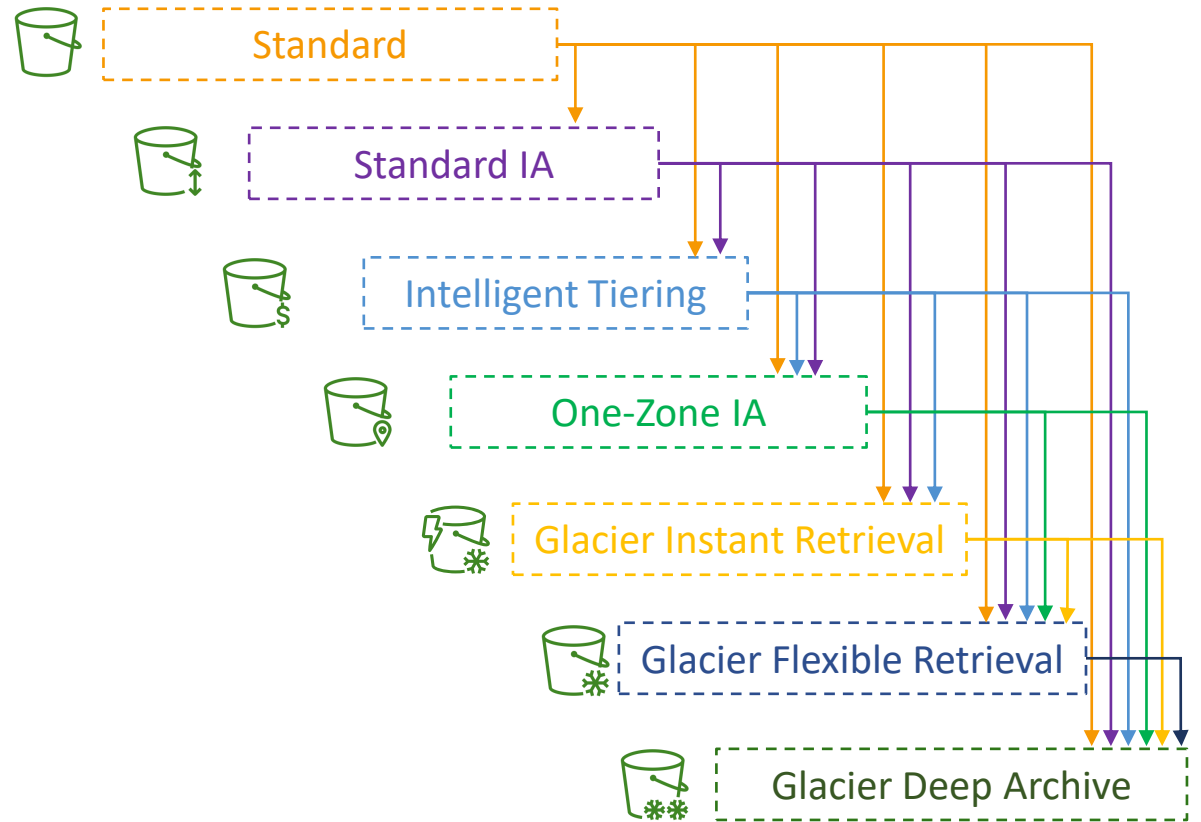
Example: us-east-1

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	\$0.023	\$0.0025 - \$0.023	\$0.0125	\$0.01	\$0.004	\$0.0036	\$0.00099
Retrieval Cost (per 1000 request)	GET: \$0.0004 POST: \$0.005	GET: \$0.0004 POST: \$0.005	GET: \$0.001 POST: \$0.01	GET: \$0.001 POST: \$0.01	GET: \$0.01 POST: \$0.02	GET: \$0.0004 POST: \$0.03 Expedited: \$10 Standard: \$0.05 Bulk: free	GET: \$0.0004 POST: \$0.05 Standard: \$0.10 Bulk: \$0.025
Retrieval Time	Instantaneous					Expedited (1 – 5 mins) Standard (3 – 5 hours) Bulk (5 – 12 hours)	Standard (12 hours) Bulk (48 hours)
Monitoring Cost (per 1000 objects)		\$0.0025					

<https://aws.amazon.com/s3/pricing/>

Amazon S3 – Moving between Storage Classes

- You can transition objects between storage classes
- For infrequently accessed object, move them to **Standard IA**
- For archive objects that you don't need fast access to, move them to **Glacier or Glacier Deep Archive**
- Moving objects can be automated using a **Lifecycle Rules**



Amazon S3 – Lifecycle Rules



- **Transition Actions** – configure objects to transition to another storage class
 - Move objects to Standard IA class 60 days after creation
 - Move to Glacier for archiving after 6 months
- **Expiration actions** – configure objects to expire (delete) after some time
 - Access log files can be set to delete after a 365 days
 - **Can be used to delete old versions of files (if versioning is enabled)**
 - Can be used to delete incomplete Multi-Part uploads
- Rules can be created for a certain prefix (example: *s3://mybucket/mp3/**)

Amazon S3 – Lifecycle Rules (Scenario 1)

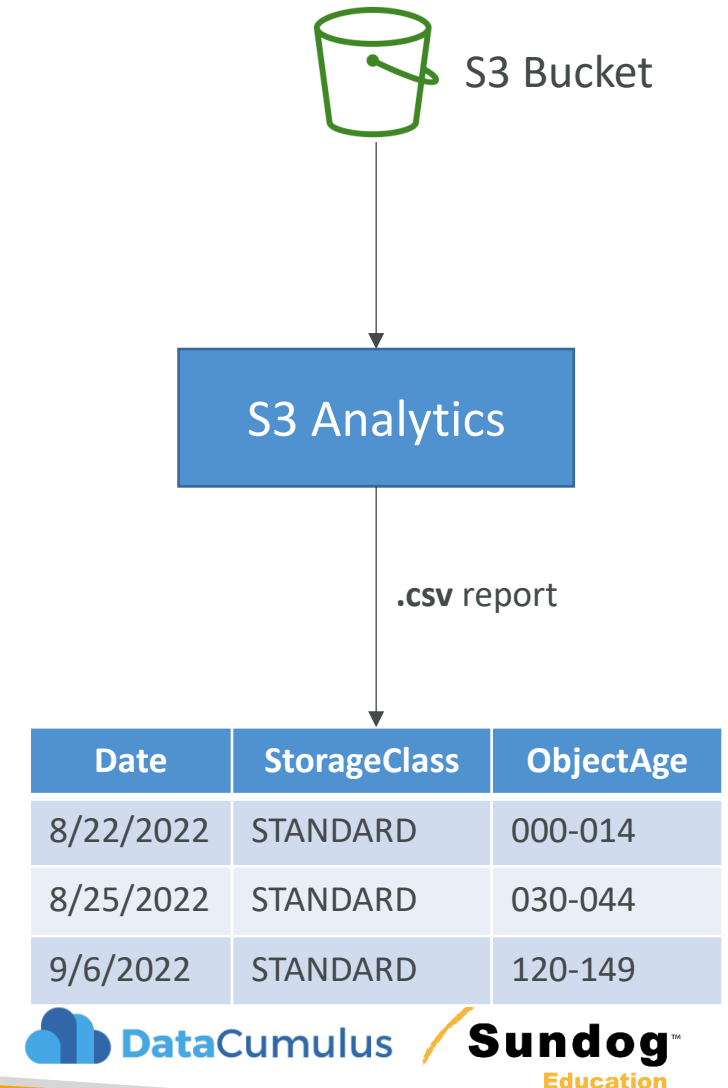
- Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can wait up to 6 hours. How would you design this?
- S3 source images can be on **Standard**, with a lifecycle configuration to transition them to **Glacier** after 60 days
- S3 thumbnails can be on **One-Zone IA**, with a lifecycle configuration to expire them (delete them) after 60 days

Amazon S3 – Lifecycle Rules (Scenario 2)

- A rule in your company states that you should be able to recover your deleted S3 objects immediately for 30 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.
- **Enable S3 Versioning** in order to have object versions, so that “deleted objects” are in fact hidden by a “delete marker” and can be recovered
- Transition the “noncurrent versions” of the object to **Standard IA**
- Transition afterwards the “noncurrent versions” to **Glacier Deep**

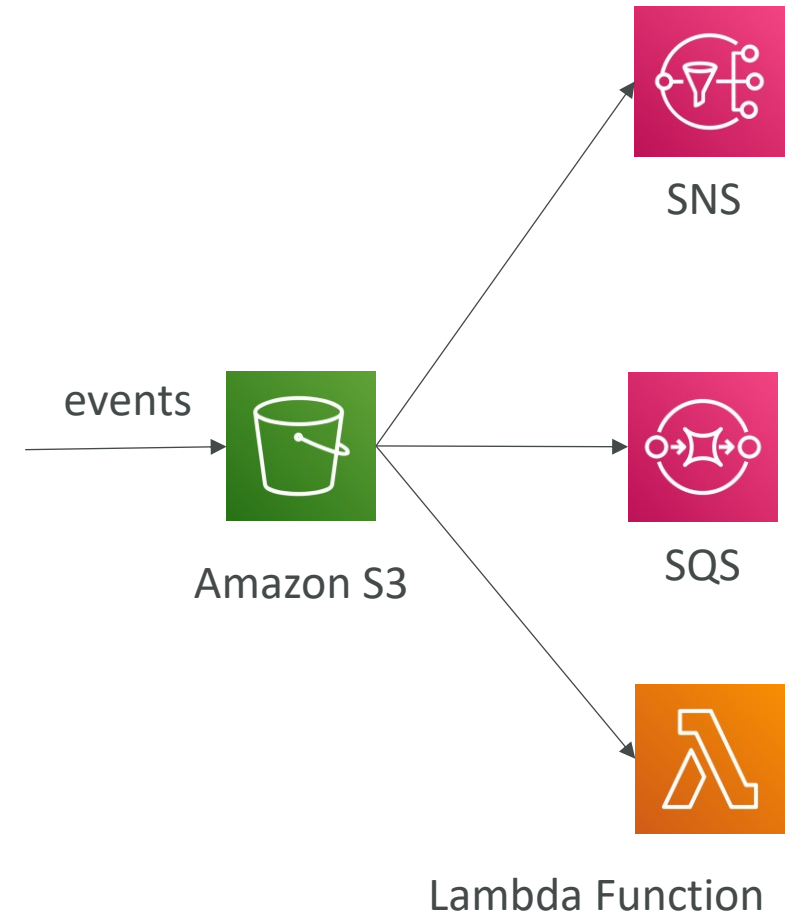
Amazon S3 Analytics – Storage Class Analysis

- Help you decide when to transition objects to the right storage class
- Recommendations for **Standard** and **Standard IA**
 - Does NOT work for One-Zone IA or Glacier
- Report is updated daily
- 24 to 48 hours to start seeing data analysis
- Good first step to put together Lifecycle Rules (or improve them)!

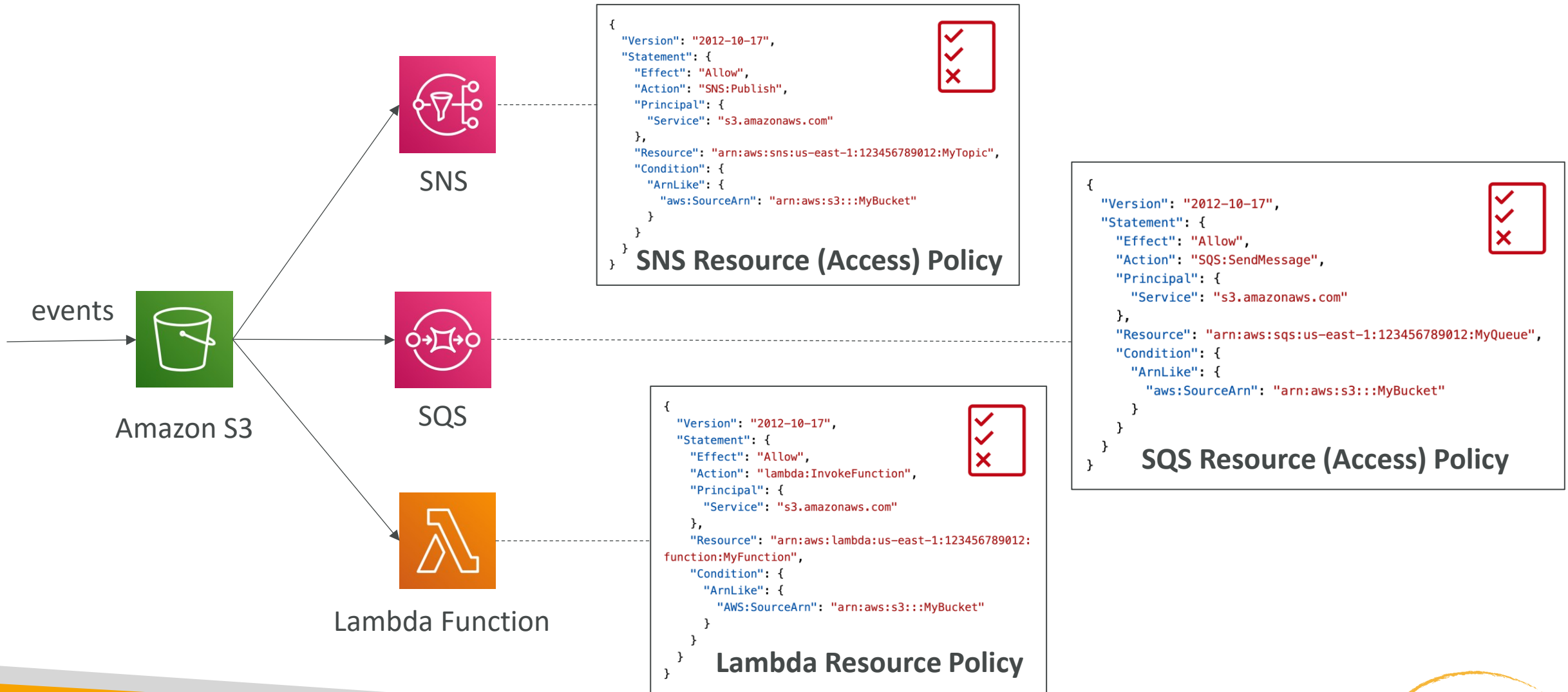


S3 Event Notifications

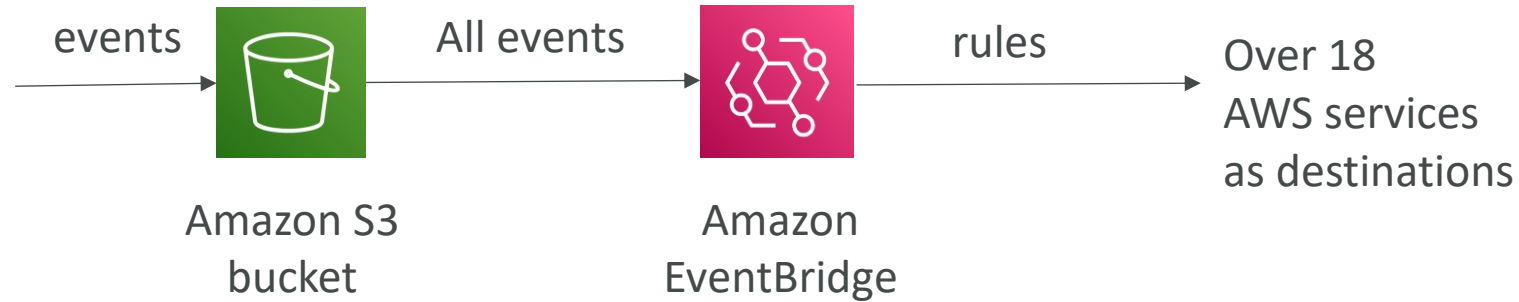
- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...
- Object name filtering possible (*.jpg)
- Use case: generate thumbnails of images uploaded to S3
- **Can create as many “S3 events” as desired**
- S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer



S3 Event Notifications – IAM Permissions



S3 Event Notifications with Amazon EventBridge



- **Advanced filtering** options with JSON rules (metadata, object size, name...)
- **Multiple Destinations** – ex , Kinesis Streams / Firehose...
- **EventBridge Capabilities** – Archive, Replay Events, Reliable delivery

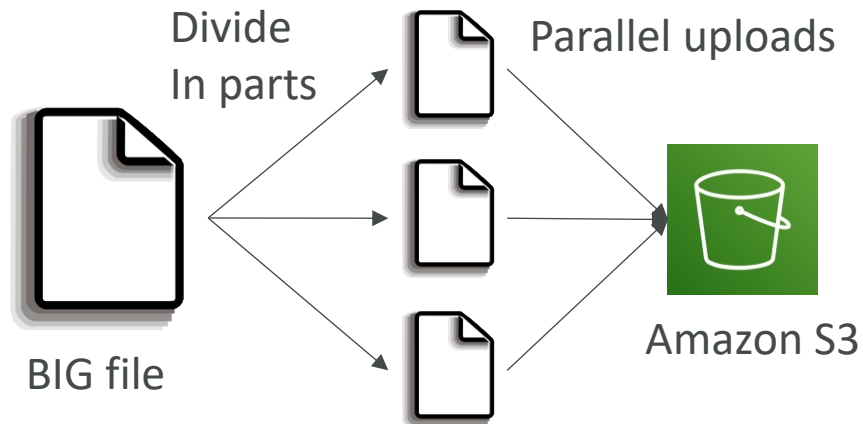
S3 – Baseline Performance

- Amazon S3 automatically scales to high request rates, latency 100-200 ms
- Your application can achieve at least **3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in a bucket.**
- There are no limits to the number of prefixes in a bucket.
- Example (object path => prefix):
 - bucket/folder1/sub1/file => /folder1/sub1/
 - bucket/folder1/sub2/file => /folder1/sub2/
 - bucket/1/file => /1/
 - bucket/2/file => /2/
- If you spread reads across all four prefixes evenly, you can achieve 22,000 requests per second for GET and HEAD

S3 Performance

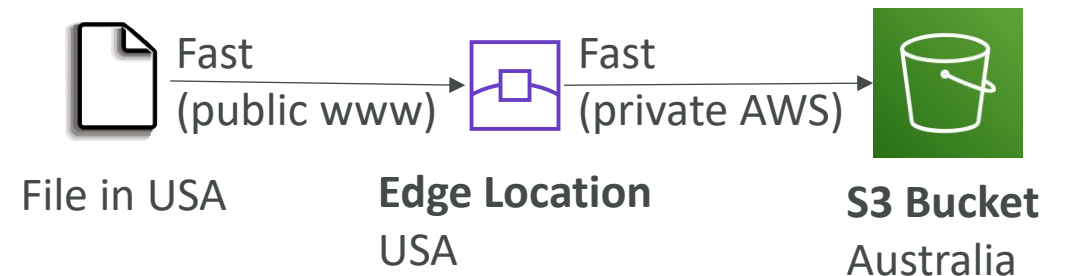
- **Multi-Part upload:**

- recommended for files > 100MB, must use for files > 5GB
- Can help parallelize uploads (speed up transfers)



- **S3 Transfer Acceleration**

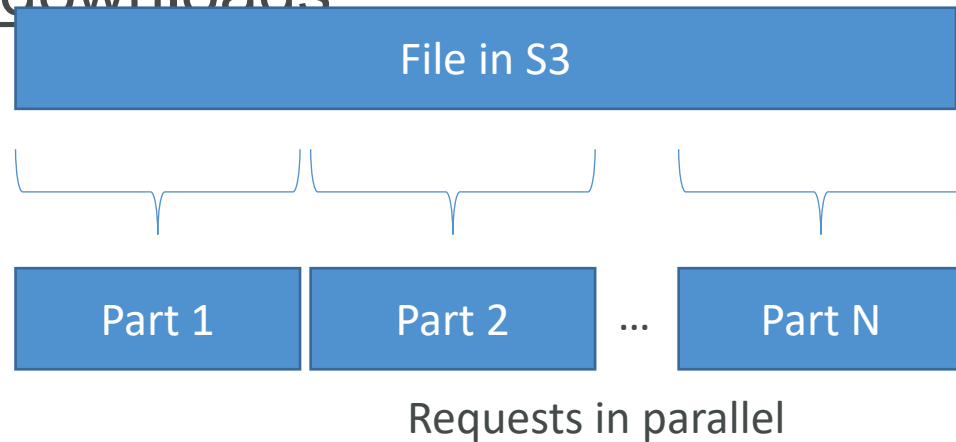
- Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region
- Compatible with multi-part upload



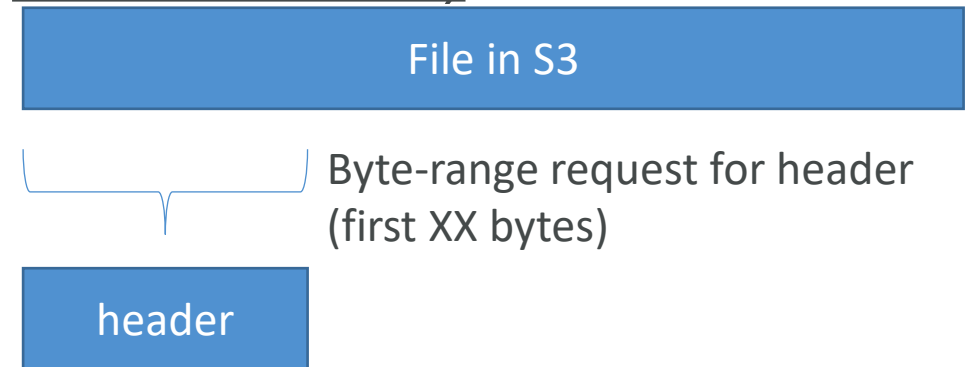
S3 Performance – S3 Byte-Range Fetches

- Parallelize GETs by requesting specific byte ranges
- Better resilience in case of failures

Can be used to speed up downloads



Can be used to retrieve only partial data (for example the head of a file)



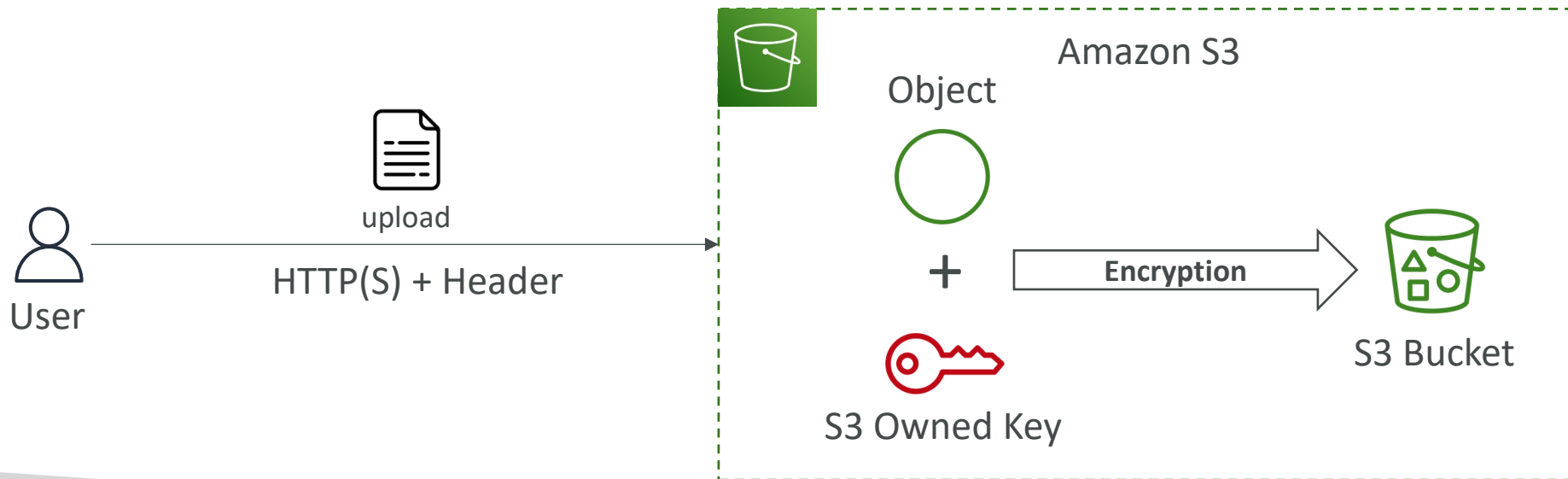
Amazon S3 – Object Encryption



- You can encrypt objects in S3 buckets using one of 4 methods
- **Server-Side Encryption (SSE)**
 - **Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) – Enabled by Default**
 - Encrypts S3 objects using keys handled, managed, and owned by AWS
 - **Server-Side Encryption with KMS Keys stored in AWS KMS (SSE-KMS)**
 - Leverage AWS Key Management Service (AWS KMS) to manage encryption keys
 - **Server-Side Encryption with Customer-Provided Keys (SSE-C)**
 - When you want to manage your own encryption keys
- **Client-Side Encryption**
- It's important to understand which ones are for which situation for the exam

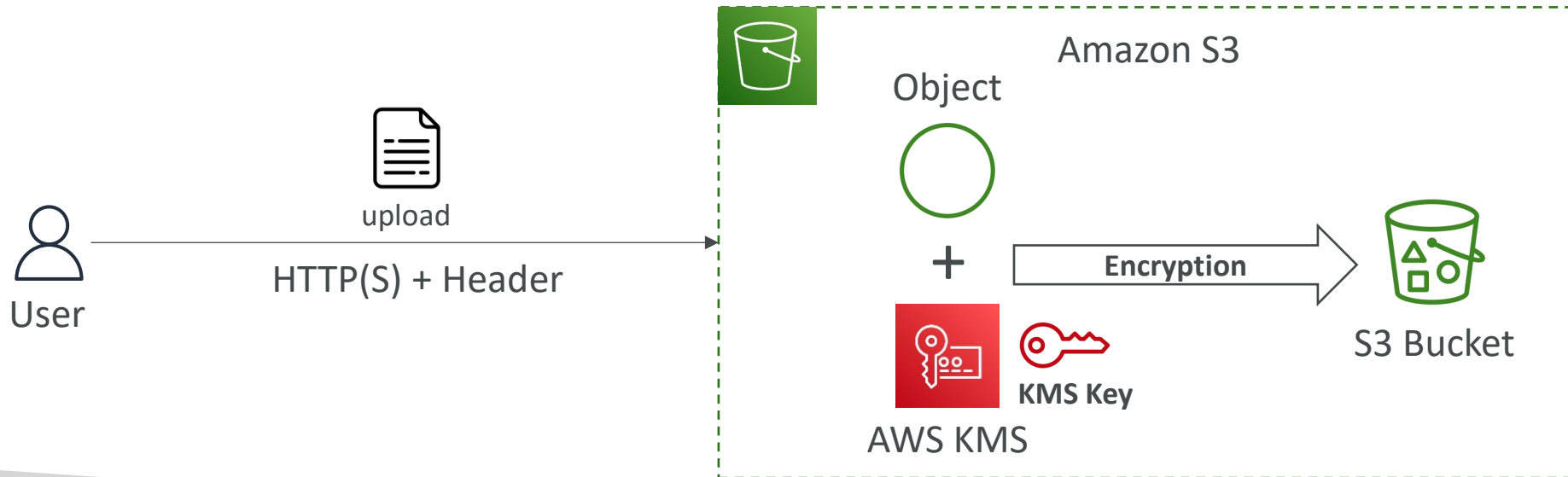
Amazon S3 Encryption – SSE-S3

- Encryption using keys handled, managed, and owned by AWS
- Object is encrypted server-side
- Encryption type is **AES-256**
- Must set header "**x-amz-server-side-encryption**": "**AES256**"
- **Enabled by default for new buckets & new objects**



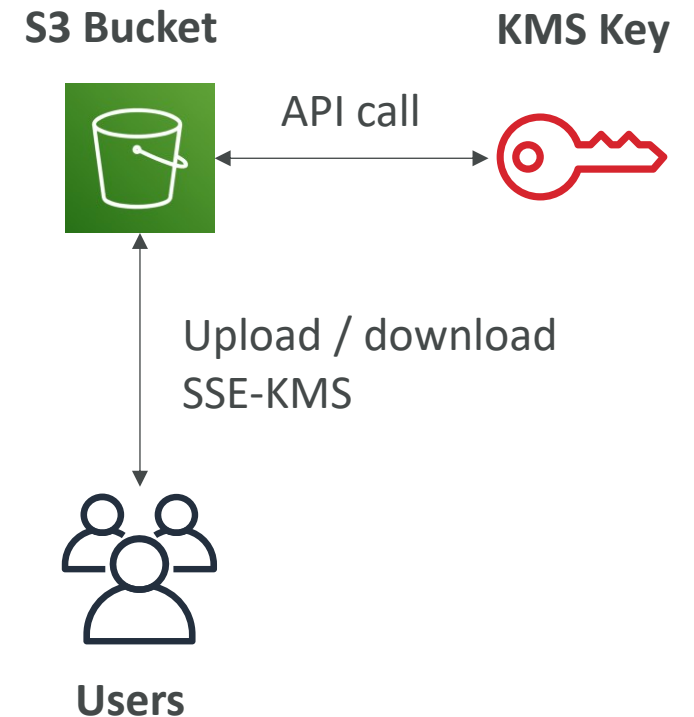
Amazon S3 Encryption – SSE-KMS

- Encryption using keys handled and managed by AWS KMS (Key Management Service)
- KMS advantages: user control + audit key usage using CloudTrail
- Object is encrypted server side
- Must set header **"x-amz-server-side-encryption": "aws:kms"**



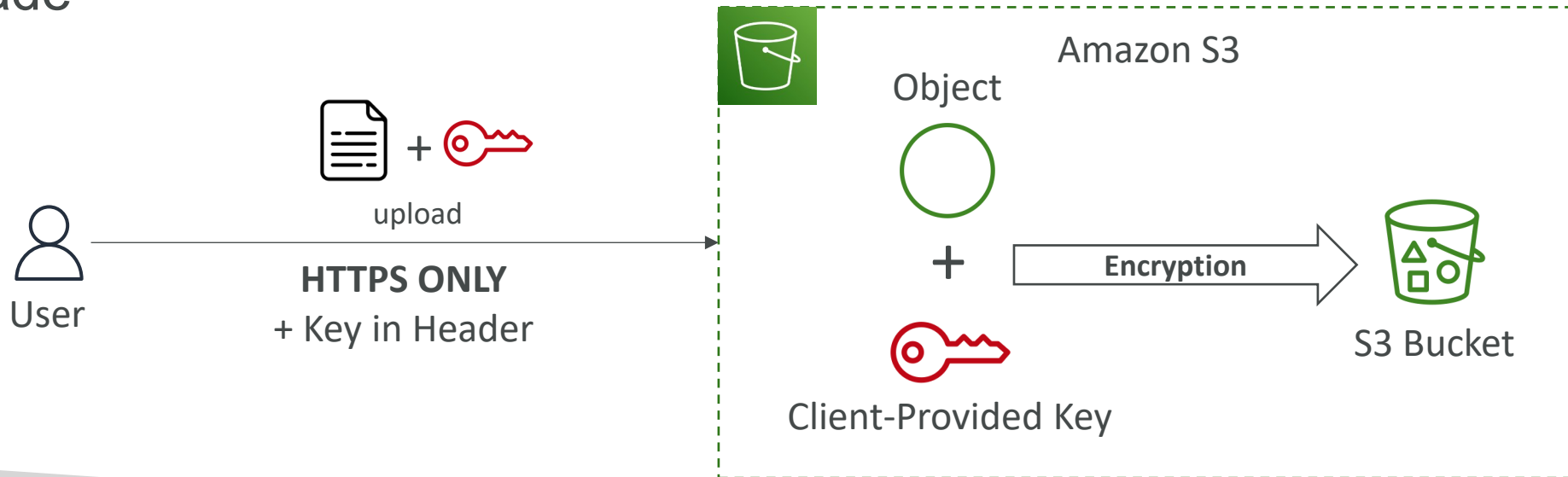
SSE-KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the **GenerateDataKey** KMS API
- When you download, it calls the **Decrypt** KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- You can request a quota increase using the Service Quotas Console



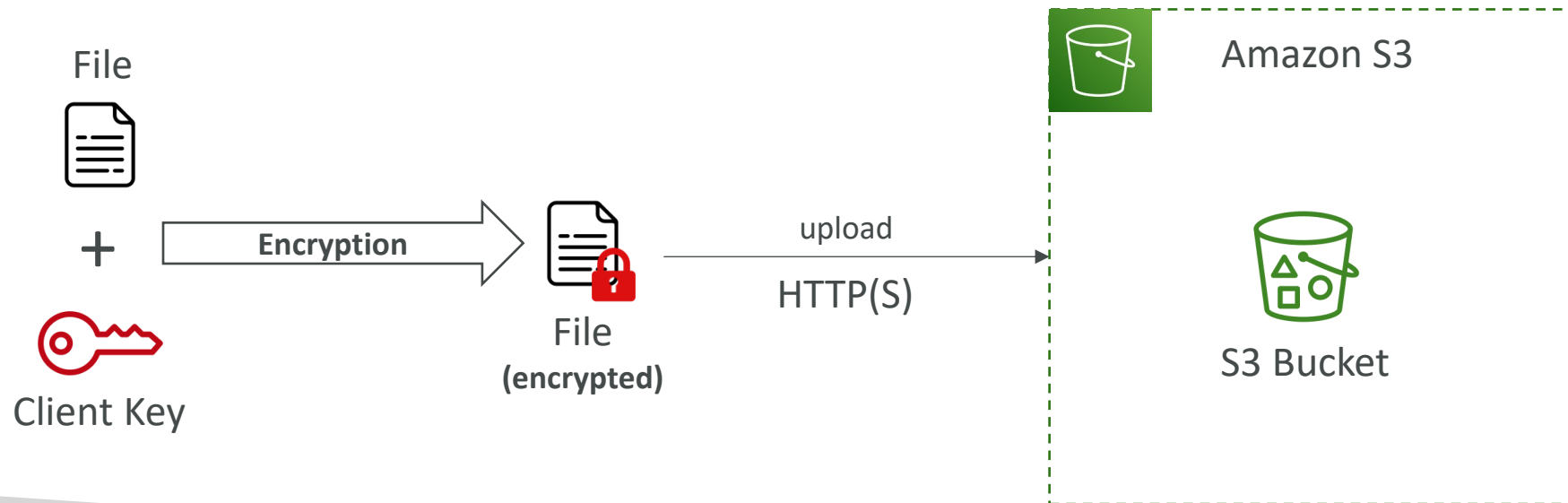
Amazon S3 Encryption – SSE-C

- Server-Side Encryption using keys fully managed by the customer outside of AWS
- Amazon S3 does **NOT** store the encryption key you provide
- **HTTPS must be used**
- Encryption key must be provided in HTTP headers, for every HTTP request made



Amazon S3 Encryption – Client-Side Encryption

- Use client libraries such as **Amazon S3 Client-Side Encryption Library**
- Clients must encrypt data themselves before sending to Amazon S3
- Clients must decrypt data themselves when retrieving from Amazon S3
- Customer fully manages the keys and encryption cycle



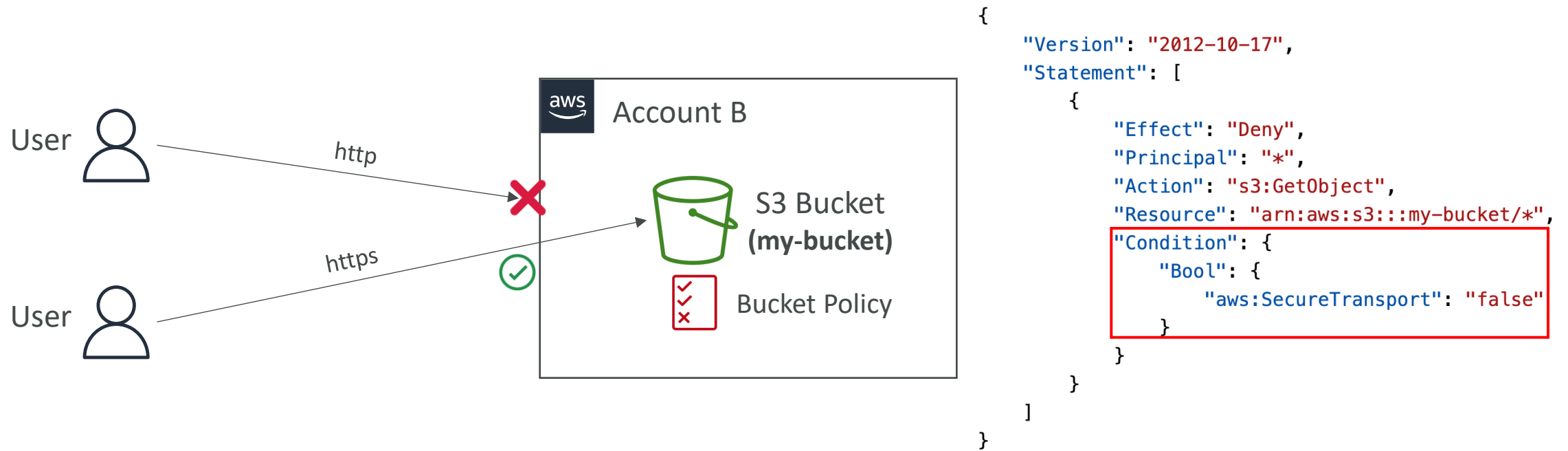
Amazon S3 – Encryption in transit (SSL/TLS)

- Encryption in flight is also called SSL/TLS
- Amazon S3 exposes two endpoints:
 - **HTTP Endpoint** – non encrypted
 - **HTTPS Endpoint** – encryption in flight
- **HTTPS is recommended**
- **HTTPS is mandatory for SSE-C**
- Most clients would use the HTTPS endpoint by default



Amazon S3 – Force Encryption in Transit

aws:SecureTransport



Amazon S3 – Default Encryption vs. Bucket Policies

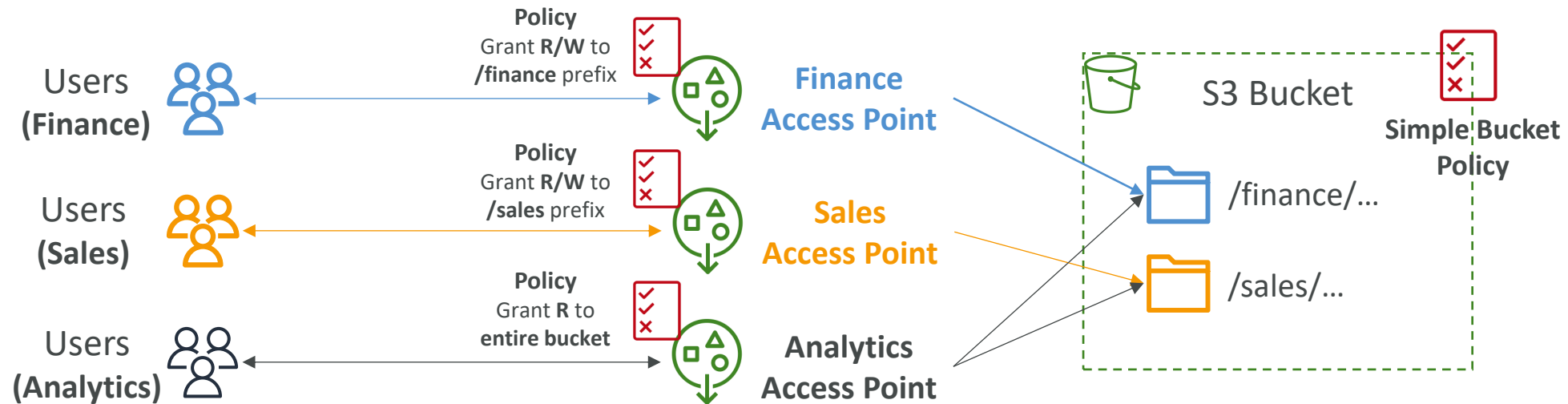
- SSE-S3 encryption is automatically applied to new objects stored in S3 bucket
- Optionally, you can “force encryption” using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "s3:PutObject",
      "Principal": "*",
      "Resource": "arn:aws:s3:::my-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "aws:kms"
        }
      }
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "s3:PutObject",
      "Principal": "*",
      "Resource": "arn:aws:s3:::my-bucket/*",
      "Condition": {
        "Null": {
          "s3:x-amz-server-side-encryption-customer-algorithm": "true"
        }
      }
    }
  ]
}
```

- **Note:** Bucket Policies are evaluated before “Default Encryption”

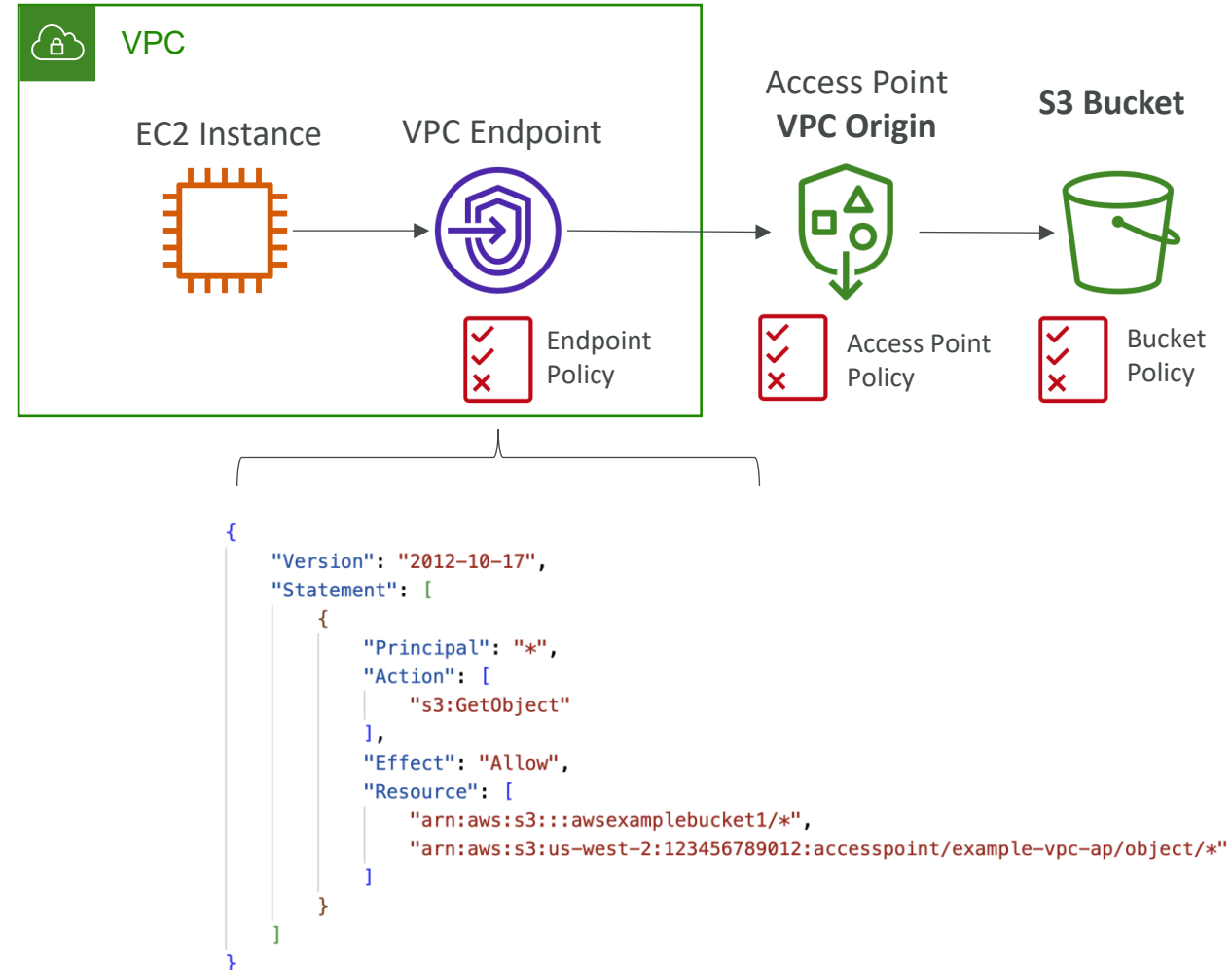
S3 – Access Points



- Access Points simplify security management for S3 Buckets
- Each Access Point has:
 - its own DNS name (Internet Origin or VPC Origin)
 - an access point policy (similar to bucket policy) – manage security at scale

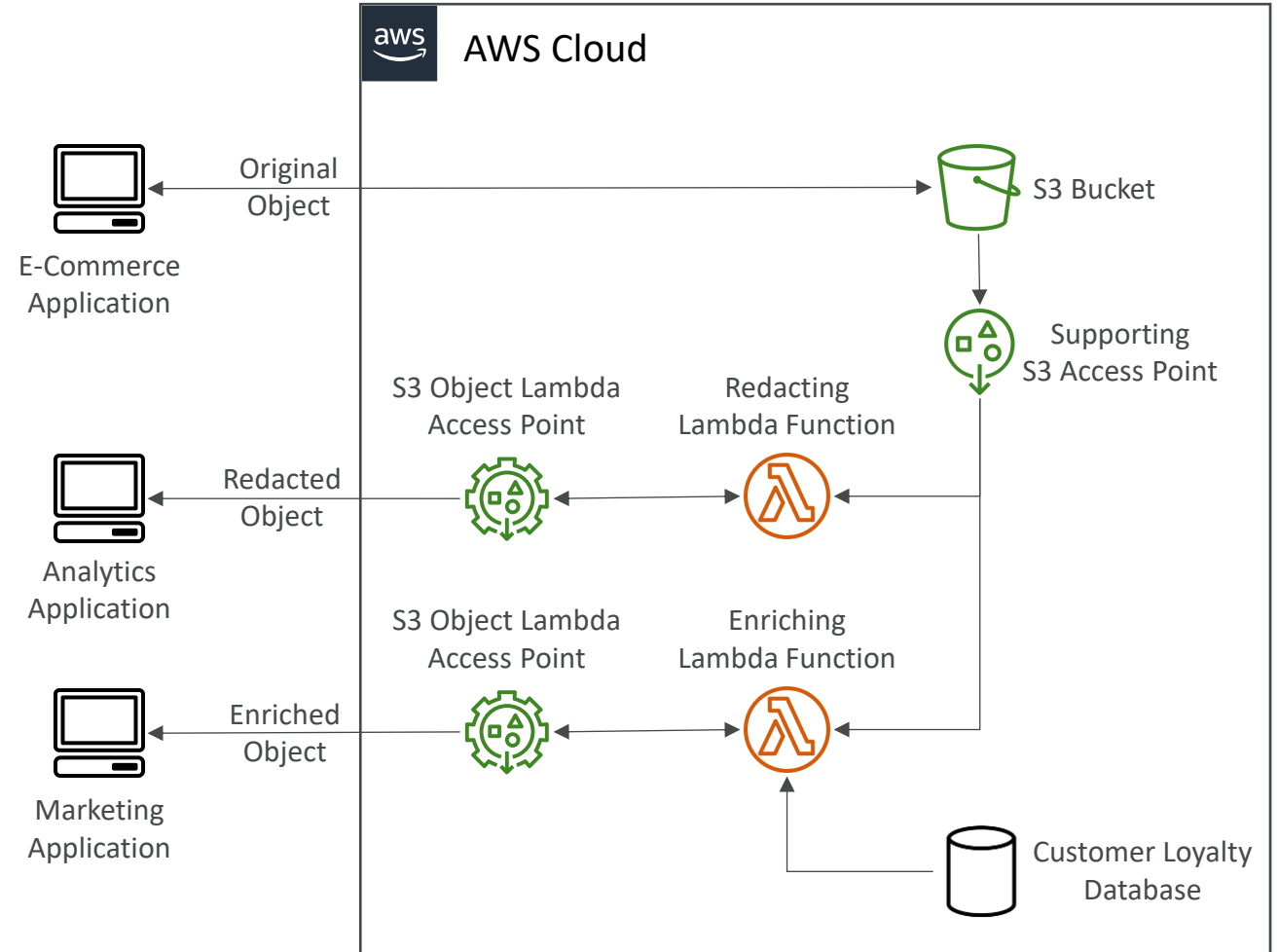
S3 – Access Points – VPC Origin

- We can define the access point to be accessible only from within the VPC
- You must create a VPC Endpoint to access the Access Point (Gateway or Interface Endpoint)
- The VPC Endpoint Policy must allow access to the target bucket and Access Point



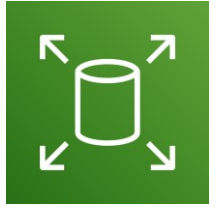
S3 Object Lambda

- Use AWS Lambda Functions to change the object before it is retrieved by the caller application
- Only one S3 bucket is needed, on top of which we create **S3 Access Point** and **S3 Object Lambda Access Points**.
- Use Cases:
 - Redacting personally identifiable information for analytics or non-production environments.
 - Converting across data formats, such as converting XML to JSON.
 - Resizing and watermarking images on the fly using caller-specific details, such as the user who requested the object.



EC2 Instance Storage Section

What's an EBS Volume?

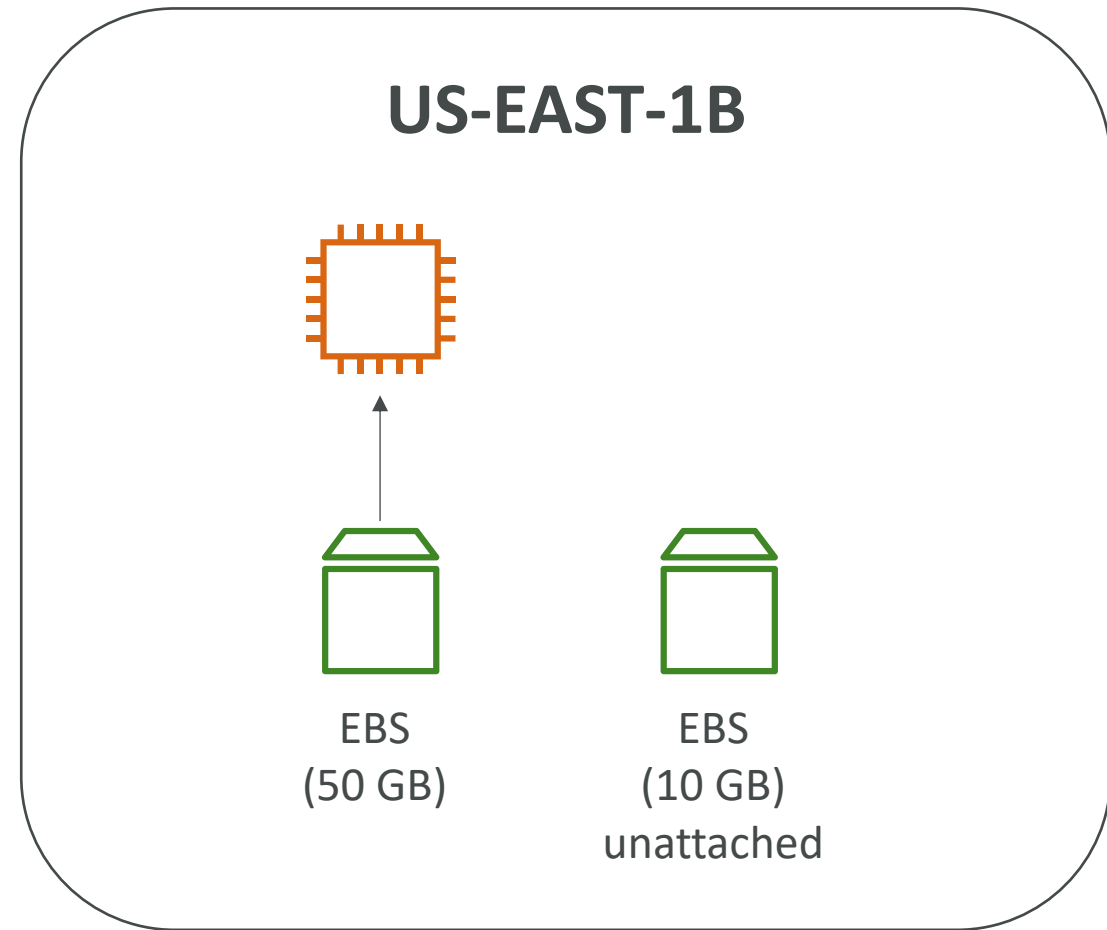
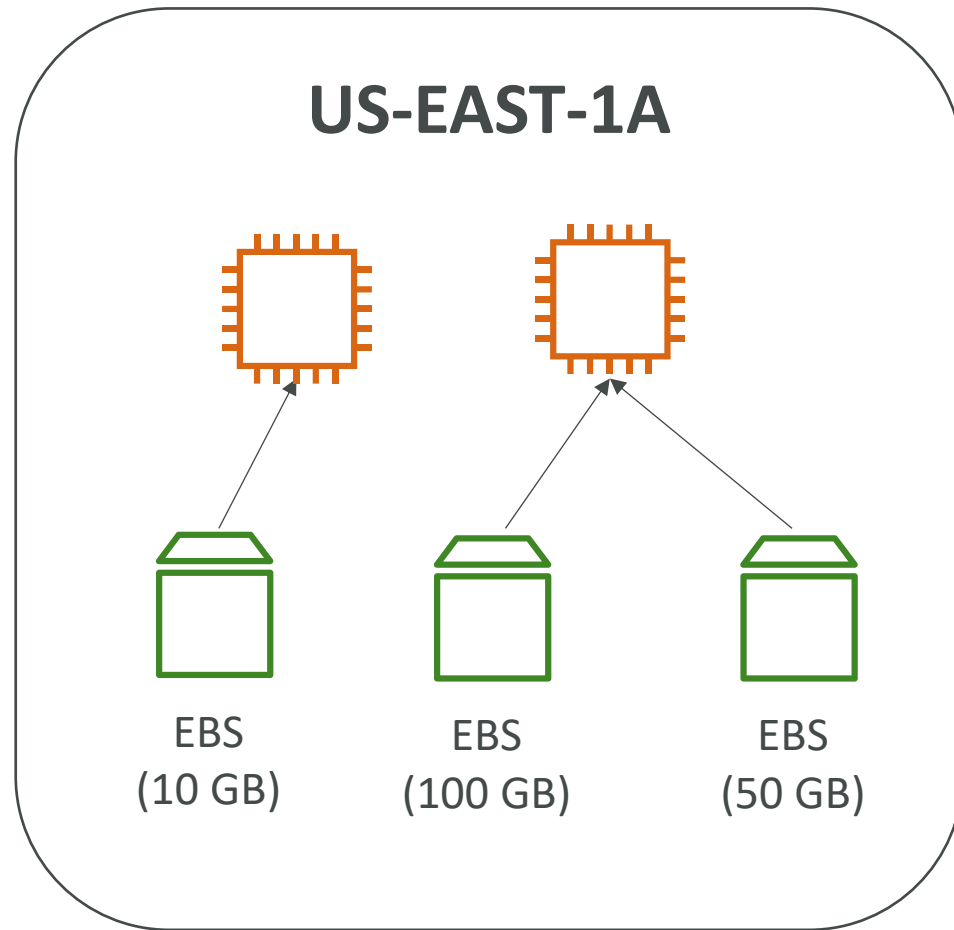


- An **EBS (Elastic Block Store) Volume** is a **network** drive you can attach to your instances while they run
- It allows your instances to persist data, even after their termination
- **They can only be mounted to one instance at a time** (at the CCP level)
- They are bound to a **specific availability zone**
- Analogy: Think of them as a “network USB stick”

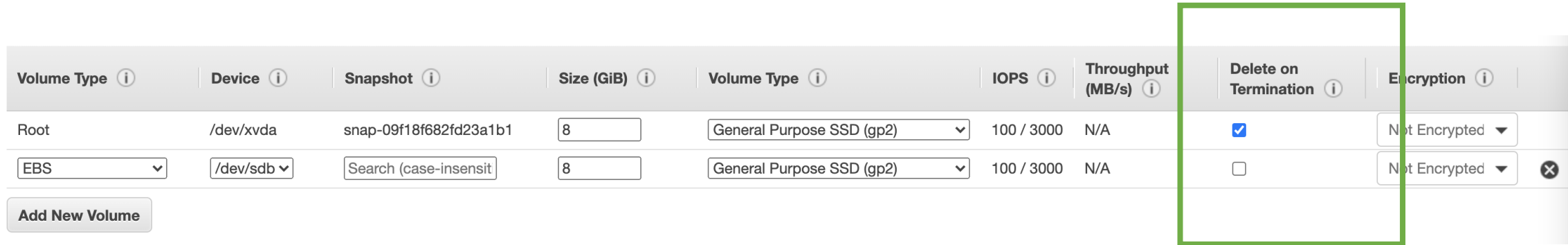
EBS Volume

- It's a network drive (i.e. not a physical drive)
 - It uses the network to communicate the instance, which means there might be a bit of latency
 - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
 - An EBS Volume in us-east-1a cannot be attached to us-east-1b
 - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
 - You get billed for all the provisioned capacity
 - You can increase the capacity of the drive over time

EBS Volume - Example



EBS – Delete on Termination attribute



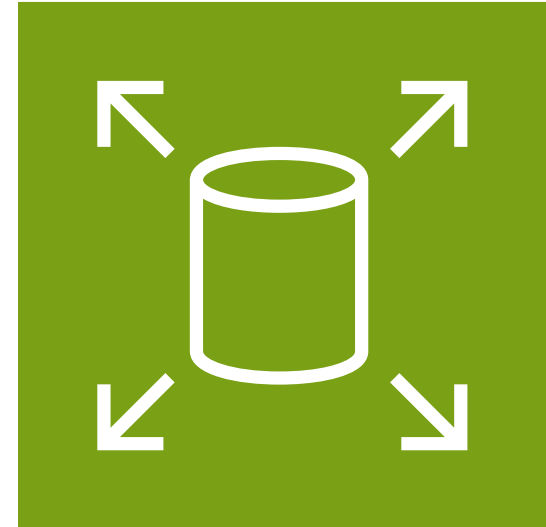
Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encryption ⓘ
Root	/dev/xvda	snap-09f18f682fd23a1b1	8	General Purpose SSD (gp2) ▾	100 / 3000	N/A	☒	Not Encrypted ▾
EBS ▾	/dev/sdb ▾	Search (case-insensit	8	General Purpose SSD (gp2) ▾	100 / 3000	N/A	☐	Not Encrypted ▾

Add New Volume

- Controls the EBS behaviour when an EC2 instance terminates
 - By default, the root EBS volume is deleted (attribute enabled)
 - By default, any other attached EBS volume is not deleted (attribute disabled)
- This can be controlled by the AWS console / AWS CLI
- **Use case: preserve root volume when instance is terminated**

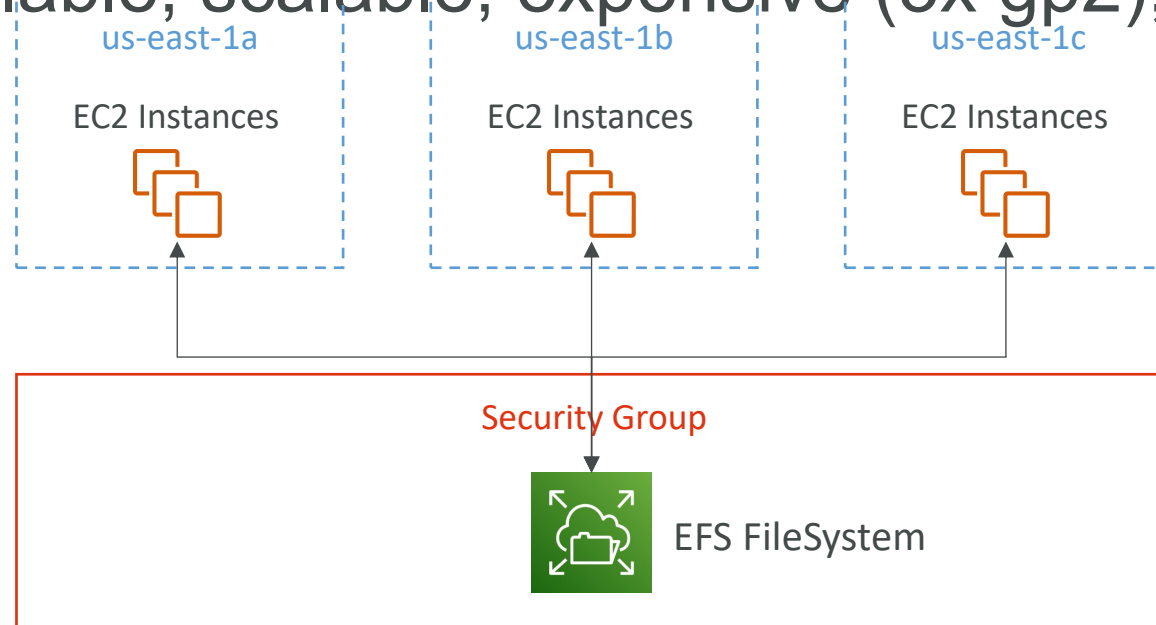
Amazon EBS Elastic Volumes

- You don't have to detach a volume or restart your instance to change it!
 - Just go to actions / modify volume from the console
- Increase volume size
 - You can only increase, not decrease
- Change volume type
 - Gp2 -> Gp3
 - Specify desired IOPS or throughput performance (or it will guess)
- Adjust performance
 - Increase or decrease



Amazon EFS – Elastic File System

- Managed NFS (network file system) that can be mounted on many EC2
- EFS works with EC2 instances in multi-AZ
- Highly available, scalable, expensive (3x gp2), pay per use



Amazon EFS – Elastic File System

- Use cases: content management, web serving, data sharing, Wordpress
 - Uses NFSv4.1 protocol
 - Uses security group to control access to EFS
 - **Compatible with Linux based AMI (not Windows)**
 - Encryption at rest using KMS
-
- POSIX file system (~Linux) that has a standard file API
 - File system scales automatically, pay-per-use, no capacity planning!

EFS – Performance & Storage Classes

- **EFS Scale**

- 1000s of concurrent NFS clients, 10 GB+ /s throughput
- Grow to Petabyte-scale network file system, automatically

- **Performance Mode (set at EFS creation time)**

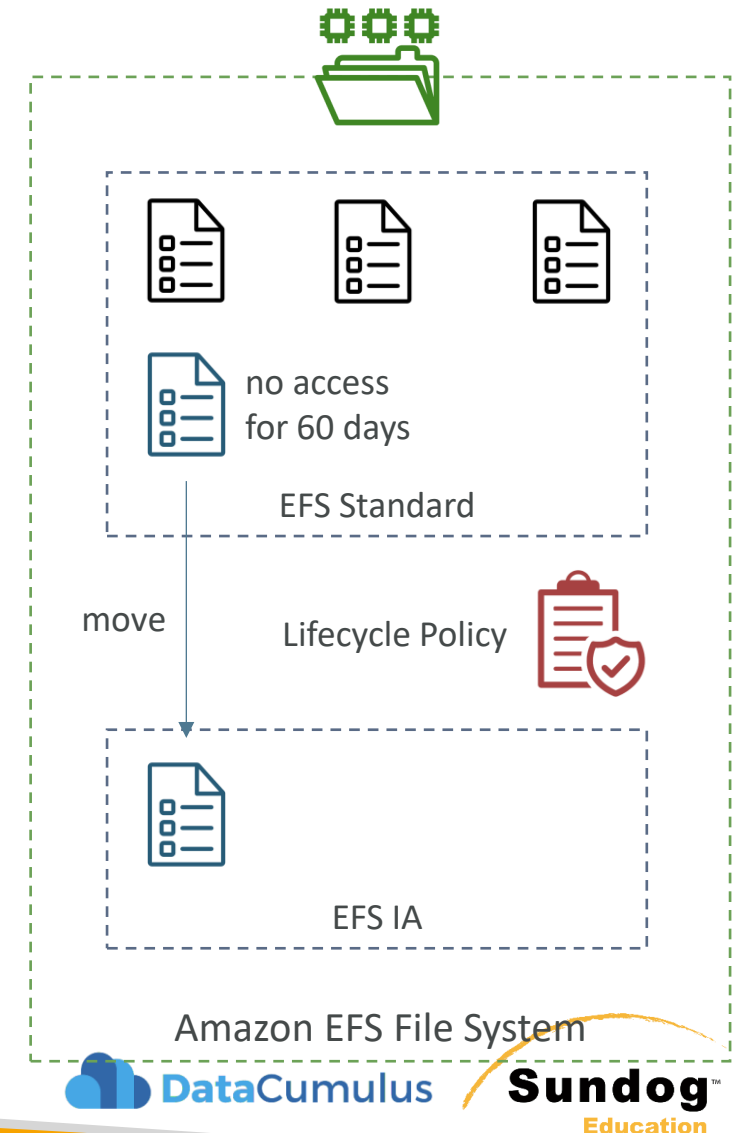
- **General Purpose (default)** – latency-sensitive use cases (web server, CMS, etc...)
- **Max I/O** – higher latency, throughput, highly parallel (big data, media processing)

- **Throughput Mode**

- **Bursting** – 1 TB = 50MiB/s + burst of up to 100MiB/s
- **Provisioned** – set your throughput regardless of storage size, ex: 1 GiB/s for 1 TB storage
- **Elastic** – automatically scales throughput up or down based on your workloads
 - Up to 3GiB/s for reads and 1GiB/s for writes
 - Used for unpredictable workloads

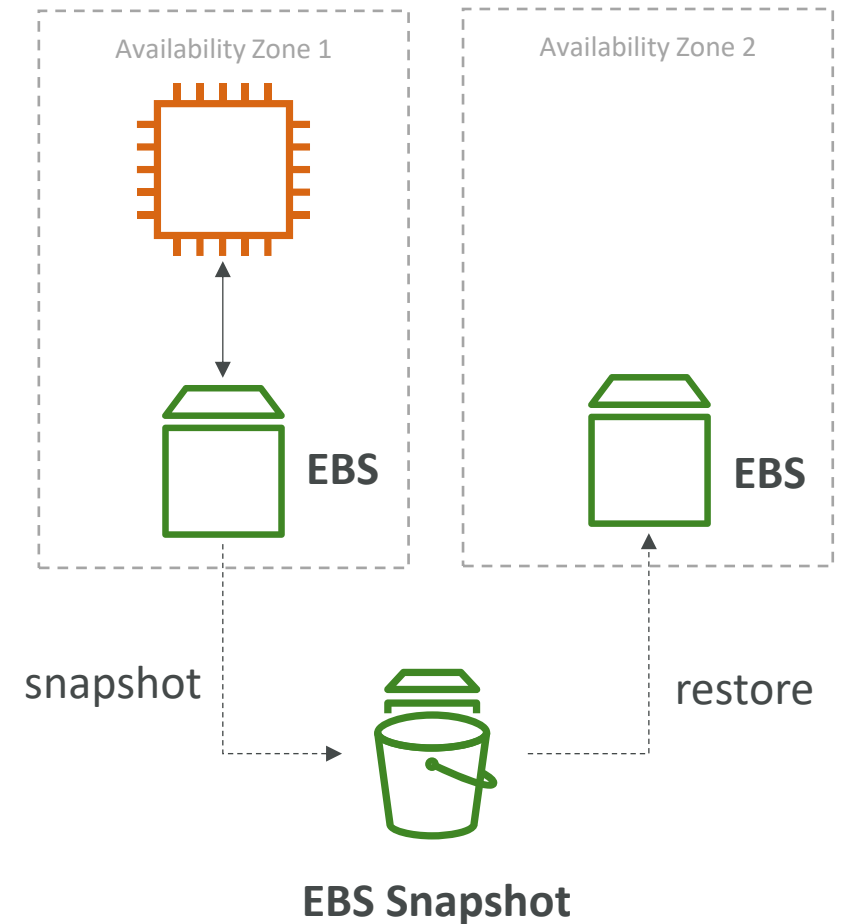
EFS – Storage Classes

- **Storage Tiers (lifecycle management feature – move file after N days)**
 - **Standard:** for frequently accessed files
 - **Infrequent access (EFS-IA):** cost to retrieve files, lower price to store.
 - **Archive:** rarely accessed data (few times each year), 50% cheaper
 - Implement **lifecycle policies** to move files between storage tiers
- **Availability and durability**
 - Standard: Multi-AZ, great for prod
 - One Zone: One AZ, great for dev, backup enabled by default, compatible with IA (EFS One Zone-IA)
- Over 90% in cost savings



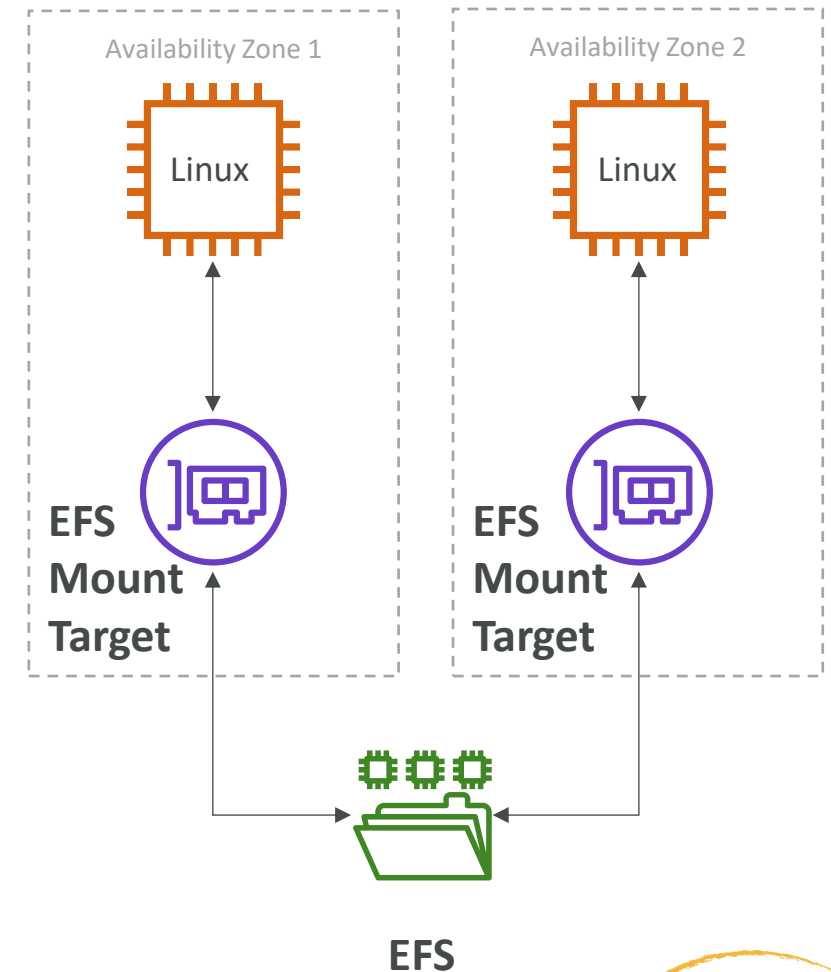
EBS vs EFS – Elastic Block Storage

- EBS volumes...
 - one instance (except multi-attach io1/io2)
 - are locked at the Availability Zone (AZ) level
 - gp2: IO increases if the disk size increases
 - gp3 & io1: can increase IO independently
- To migrate an EBS volume across AZ
 - Take a snapshot
 - Restore the snapshot to another AZ
 - EBS backups use IO and you shouldn't run them while your application is handling a lot of traffic
- Root EBS Volumes of instances get terminated by default if the EC2 instance gets terminated. (you can disable that)



EBS vs EFS – Elastic File System

- Mounting 100s of instances across AZ
- EFS share website files (WordPress)
- Only for Linux Instances (POSIX)
- EFS has a higher price point than EBS
- Can leverage Storage Tiers for cost savings



Amazon FSx – Overview



- **Launch 3rd party high-performance file systems on AWS**
- Fully managed service



FSx for Lustre



FSx for
NetApp ONTAP



FSx for Windows
File Server



FSx for
OpenZFS

Amazon FSx for Windows (File Server)



- **FSx for Windows** is a fully managed **Windows** file system share drive
- Supports SMB protocol & Windows NTFS
- Microsoft Active Directory integration, ACLs, user quotas
- **Can be mounted on Linux EC2 instances**
- Supports **Microsoft's Distributed File System (DFS) Namespaces** (group files across multiple FS)
- Scale up to 10s of GB/s, millions of IOPS, 100s PB of data
- Storage Options:
 - **SSD** – latency sensitive workloads (databases, media processing, data analytics, ...)
 - **HDD** – broad spectrum of workloads (home directory, CMS, ...)
- Can be accessed from your on-premises infrastructure (VPN or Direct Connect)
- Can be configured to be Multi-AZ (high availability)
- Data is backed-up daily to S3

Amazon FSx for Lustre



- Lustre is a type of parallel distributed file system, for large-scale computing
- The name Lustre is derived from “Linux” and “cluster”
- Machine Learning, **High Performance Computing (HPC)**
- Video Processing, Financial Modeling, Electronic Design Automation
- Scales up to 100s GB/s, millions of IOPS, sub-ms latencies
- Storage Options:
 - **SSD** – low-latency, IOPS intensive workloads, small & random file operations
 - **HDD** – throughput-intensive workloads, large & sequential file operations
- **Seamless integration with S3**
 - Can “read S3” as a file system (through FSx)
 - Can write the output of the computations back to S3 (through FSx)
- **Can be used from on-premises servers (VPN or Direct Connect)**

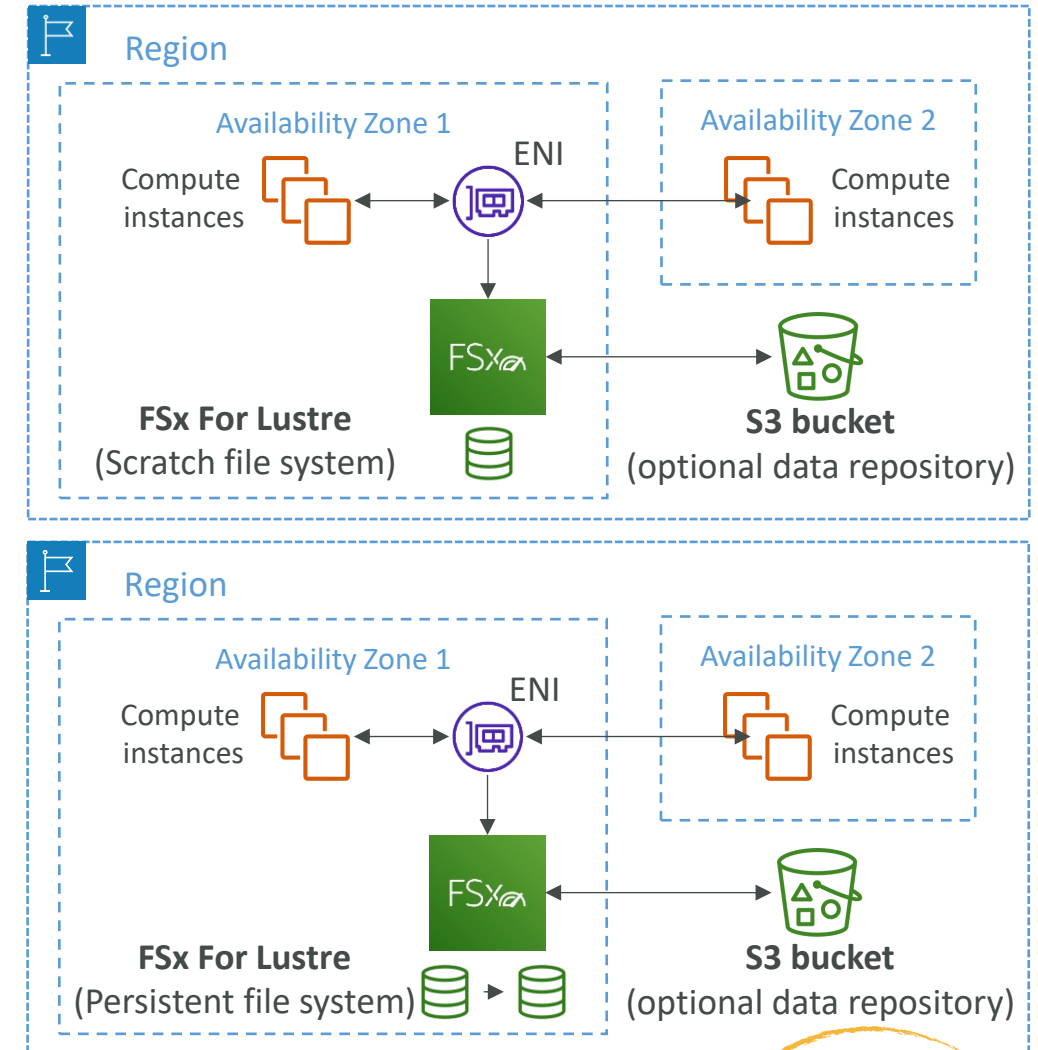
FSx Lustre - File System Deployment Options

• Scratch File System

- Temporary storage
- Data is not replicated (doesn't persist if file server fails)
- High burst (6x faster, 200MBps per TiB)
- Usage: short-term processing, optimize costs

• Persistent File System

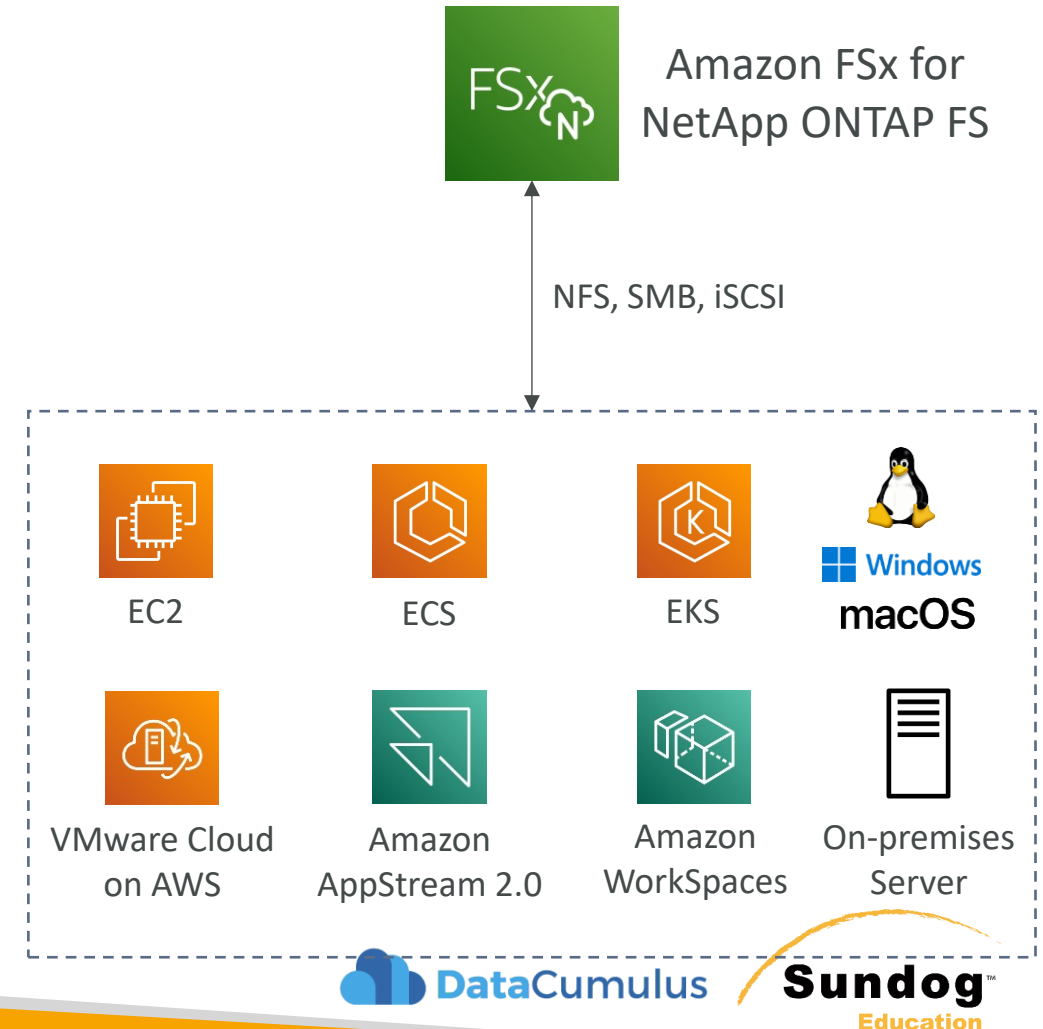
- Long-term storage
- Data is replicated within same AZ
- Replace failed files within minutes
- Usage: long-term processing, sensitive data



Amazon FSx for NetApp ONTAP



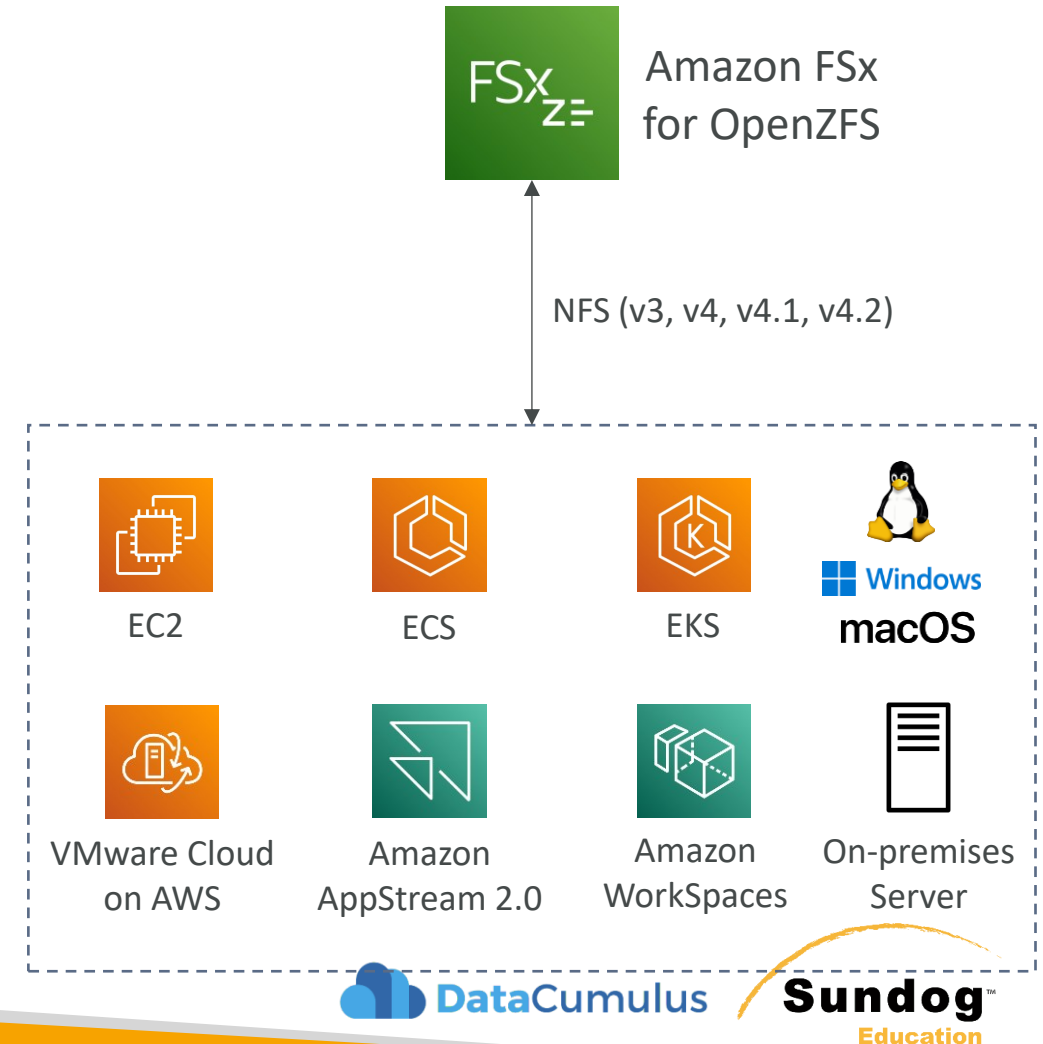
- Managed NetApp ONTAP on AWS
- **File System compatible with NFS, SMB, iSCSI protocol**
- Move workloads running on ONTAP or NAS to AWS
- Works with:
 - Linux
 - Windows
 - MacOS
 - VMware Cloud on AWS
 - Amazon Workspaces & AppStream 2.0
 - Amazon EC2, ECS and EKS
- Storage shrinks or grows automatically
- Snapshots, replication, low-cost, compression and data de-duplication
- **Point-in-time instantaneous cloning (helpful for testing new workloads)**



Amazon FSx for OpenZFS



- Managed OpenZFS file system on AWS
- File System compatible with NFS (v3, v4, v4.1, v4.2)
- Move workloads running on ZFS to AWS
- Works with:
 - Linux
 - Windows
 - MacOS
 - VMware Cloud on AWS
 - Amazon Workspaces & AppStream 2.0
 - Amazon EC2, ECS and EKS
- Up to 1,000,000 IOPS with < 0.5ms latency
- Snapshots, compression and low-cost
- **Point-in-time instantaneous cloning (helpful for testing new workloads)**



Amazon Kinesis Data Streams



- Collect and store streaming data in **real-time**



Kinesis Data Streams



- Retention between up to 365 days
- Ability to reprocess (replay) data by consumers
- Data can't be deleted from Kinesis (until it expires)
- Data up to 10MiB (typical use case is lot of “small” **real-time data**)
- Data ordering guarantee for data with the same “Partition ID”
- At-rest KMS encryption, in-flight HTTPS encryption
- Kinesis Producer Library (KPL) to write an optimized producer application
- Kinesis Client Library (KCL) to write an optimized consumer application

Kinesis Data Streams – Capacity Modes

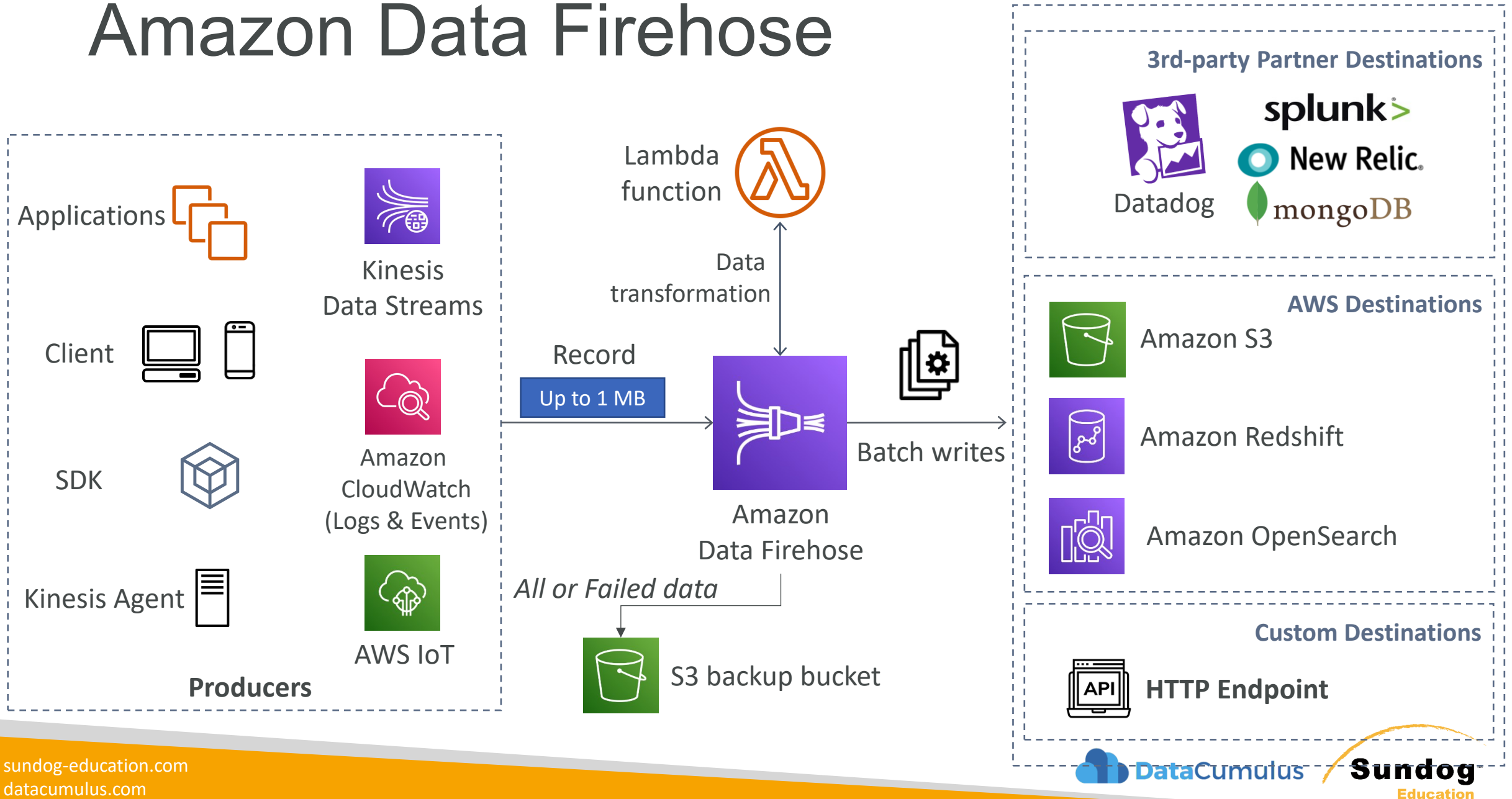
- **Provisioned mode:**

- Choose number of shards
- Each shard gets 1MB/s in (or 1000 records per second)
- Each shard gets 2MB/s out
- Scale manually to increase or decrease the number of shards
- You pay per shard provisioned per hour

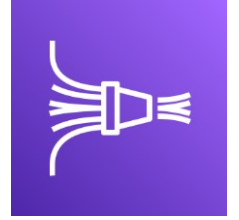
- **On-demand mode:**

- No need to provision or manage the capacity
- Default capacity provisioned (4 MB/s in or 4000 records per second)
- Scales automatically based on observed throughput peak during the last 30 days
- Pay per stream per hour & data in/out per GB

Amazon Data Firehose



Amazon Data Firehose



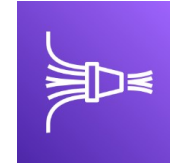
- *Note: used to be called “Kinesis Data Firehose”*
- Fully Managed Service
 - Amazon Redshift / Amazon S3 / Amazon OpenSearch Service
 - 3rd party: Splunk / MongoDB / Datadog / NewRelic / ...
 - Custom HTTP Endpoint
- Automatic scaling, serverless, pay for what you use
- **Near Real-Time** with buffering capability based on size / time
- Supports CSV, JSON, Parquet, Avro, Raw Text, Binary data
- Conversions to Parquet / ORC, compressions with gzip / snappy
- Custom data transformations using AWS Lambda (ex: CSV to JSON)

Kinesis Data Streams vs Amazon Data Firehose



Kinesis Data Streams

- Streaming data collection
- Producer & Consumer code
- Real-time
- Provisioned / On-Demand mode
- Data storage up to 365 days
- Replay Capability



Amazon Data Firehose

- Load streaming data into S3 / Redshift / OpenSearch / 3rd party / custom HTTP
- Fully managed
- Near real-time
- Automatic scaling
- No data storage
- Doesn't support replay capability

Troubleshooting Kinesis Data Stream

Producers: Performance

- Writing is too slow
 - Service limits may be exceeded. Check for throughput exceptions, see what operations are being throttled. Different calls have different limits.
 - There are shard-level limits for writes and reads
 - Other operations (ie, CreateStream, ListStreams, DescribeStreams) have stream-level limits of 5-20 calls per second
 - Select partition key to evenly distribute puts across shards
- Large producers
 - Batch things up. Use Kinesis Producer Library, PutRecords with multi-records, or aggregate records into larger files.
- Small producers (i.e. apps)
 - Use PutRecords or Kinesis Recorder in the AWS Mobile SDKs



Other Kinesis Data Stream Producer Issues

- Stream returns a 500 or 503 error
 - This indicates an `AmazonKinesisException` error rate above 1%
 - Implement a retry mechanism
- Connection errors from Flink to Kinesis
 - Network issue or lack of resources in Flink's environment
 - Could be a VPC misconfiguration
- Timeout errors from Flink to Kinesis
 - Adjust `RequestTimeout` and `#setQueueLimit` on `FlinkKinesisProducer`
- Throttling errors
 - Check for hot shards with enhanced monitoring (shard-level)
 - Check logs for “micro spikes” or obscure metrics breaching limits
 - Try a random partition key or improve the key's distribution
 - Use exponential backoff
 - Rate-limit



Troubleshooting Kinesis Data Stream Consumers

- Records get skipped with Kinesis Client Library
 - Check for unhandled exceptions on processRecords
- Records in same shard are processed by more than one processor
 - May be due to failover on the record processor workers
 - Adjust failover time
 - Handle shutdown methods with reason “ZOMBIE”
- Reading is too slow
 - Increase number of shards
 - maxRecords per call is too low
 - Your code is too slow (test an empty processor vs. yours)
- GetRecords returning empty results
 - This is normal, just keep calling GetRecords
- Shard Iterator expires unexpectedly
 - May need more write capacity on the shard table in DynamoDB
- Record processing falling behind
 - Increase retention period while troubleshooting
 - Usually insufficient resources
 - Monitor with `GetRecords.IteratorAgeMilliseconds` and `MillisBehindLatest`



Troubleshooting Kinesis Data Stream Consumers

- Lambda function can't get invoked
 - Permissions issue on execution role
 - Function is timing out (check max execution time)
 - Breaching concurrency limits
 - Monitor `IteratorAge` metric; it will increase if this is a problem
- `ReadProvisionedThroughputExceeded` exception
 - Throttling
 - Reshard your stream
 - Reduce size of `GetRecords` requests
 - Use enhanced fan-out
 - Use retries and exponential backoff



Troubleshooting Kinesis Data Stream Consumers

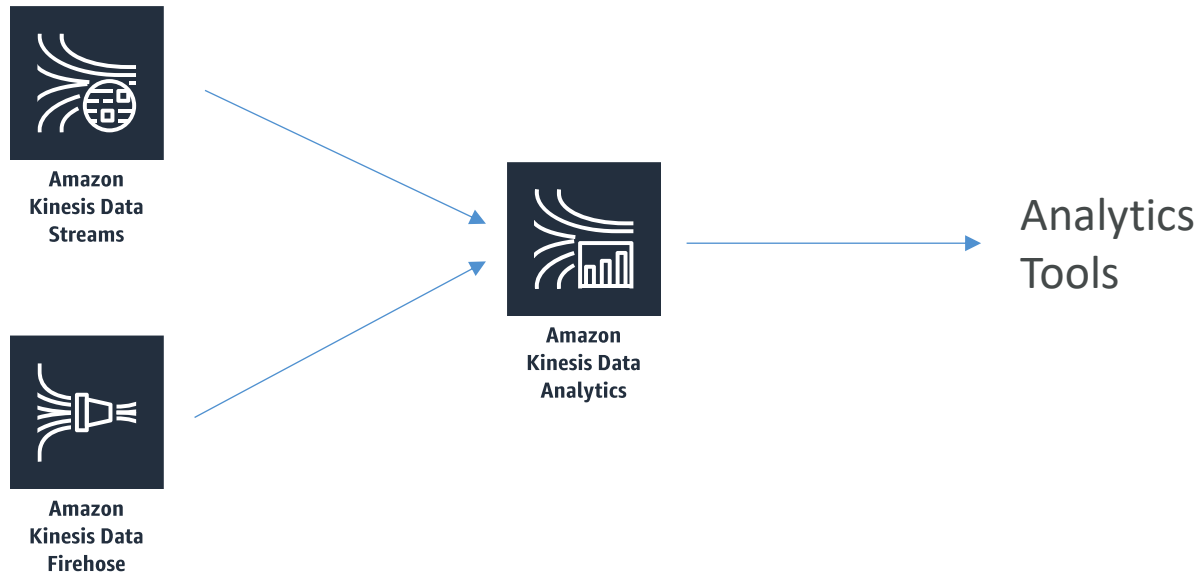
- High latency
 - Monitor with `GetRecords.Latency` and `IteratorAge`
 - Increase shards
 - Increase retention period
 - Check CPU and memory utilization (may need more memory)
- 500 errors
 - Same as producers – indicates a high error rate (>1%)
 - Implement a retry mechanism
- Blocked or stuck KCL application
 - Optimize your `processRecords` method
 - Increase `maxLeasesPerWorker`
 - Enable KCL debug logs



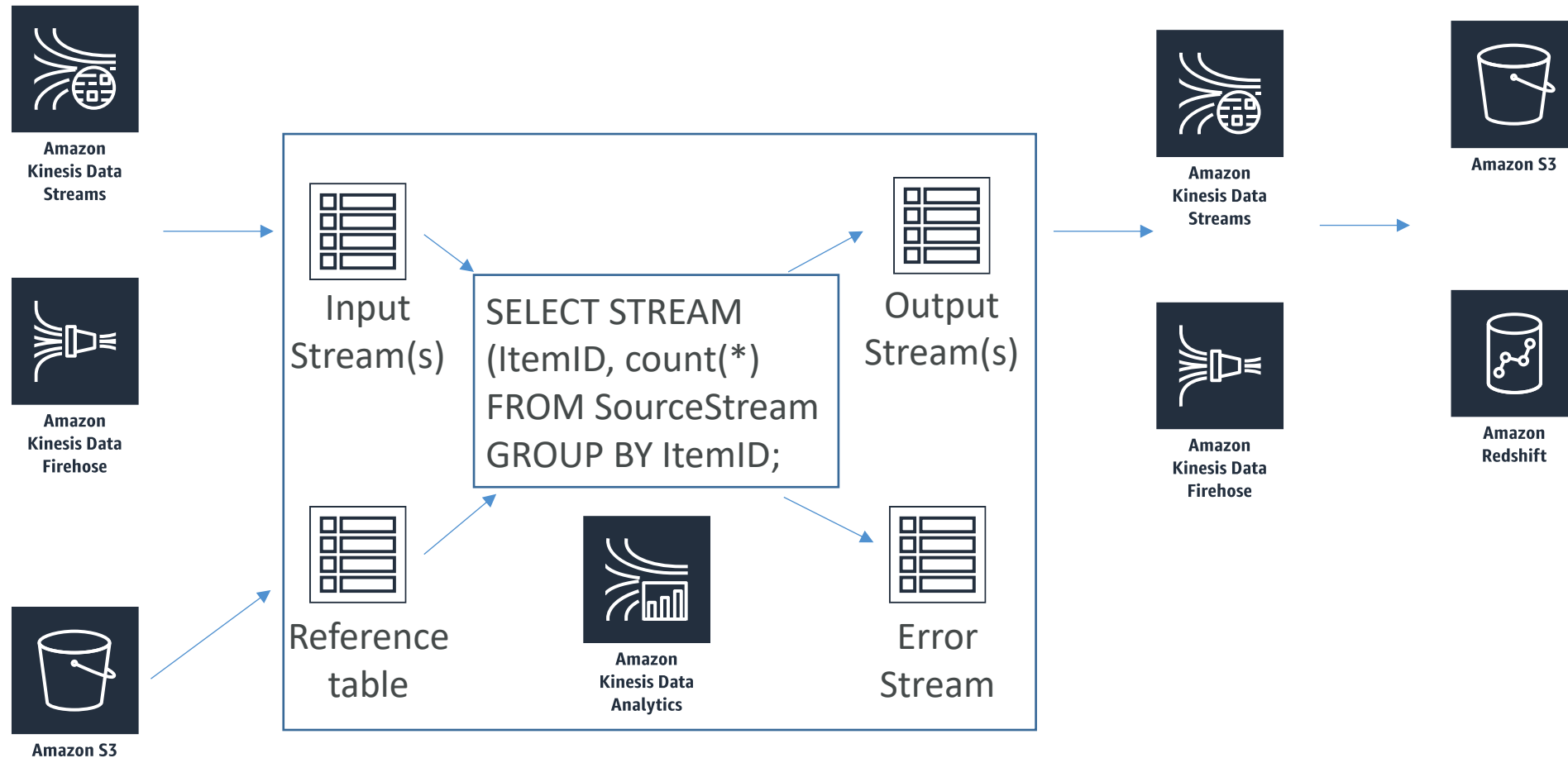
Kinesis Data Analytics / Managed Service for Apache Flink

Querying streams of data

Amazon Kinesis Data Analytics for SQL Applications

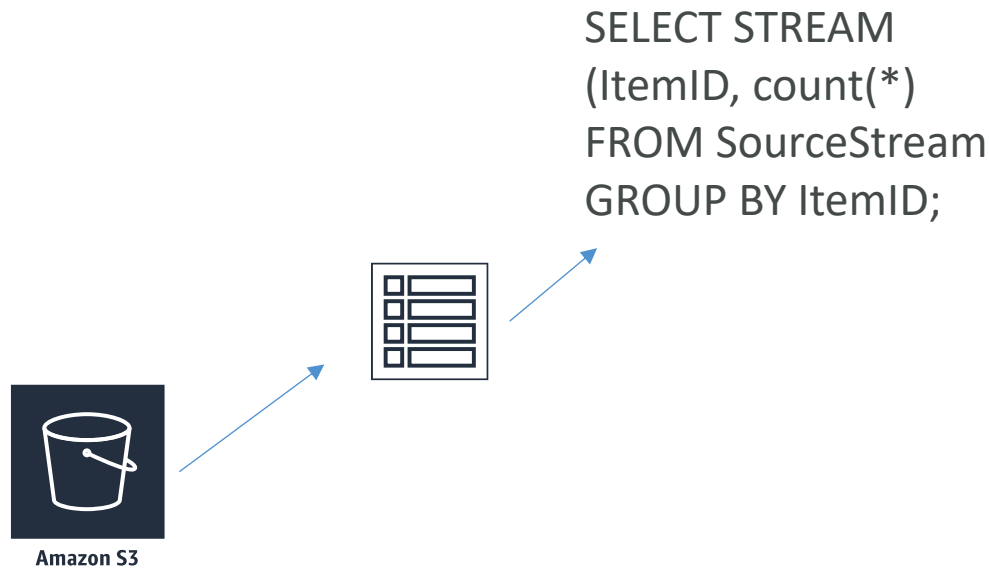


In more depth...



Reference tables are cool

- Inexpensive way to “join” data for quick lookups
 - i.e., look up the city associated with a zip code
 - Mapping is stored in S3 which is very inexpensive
 - Just use a “JOIN” command to use the data in your queries

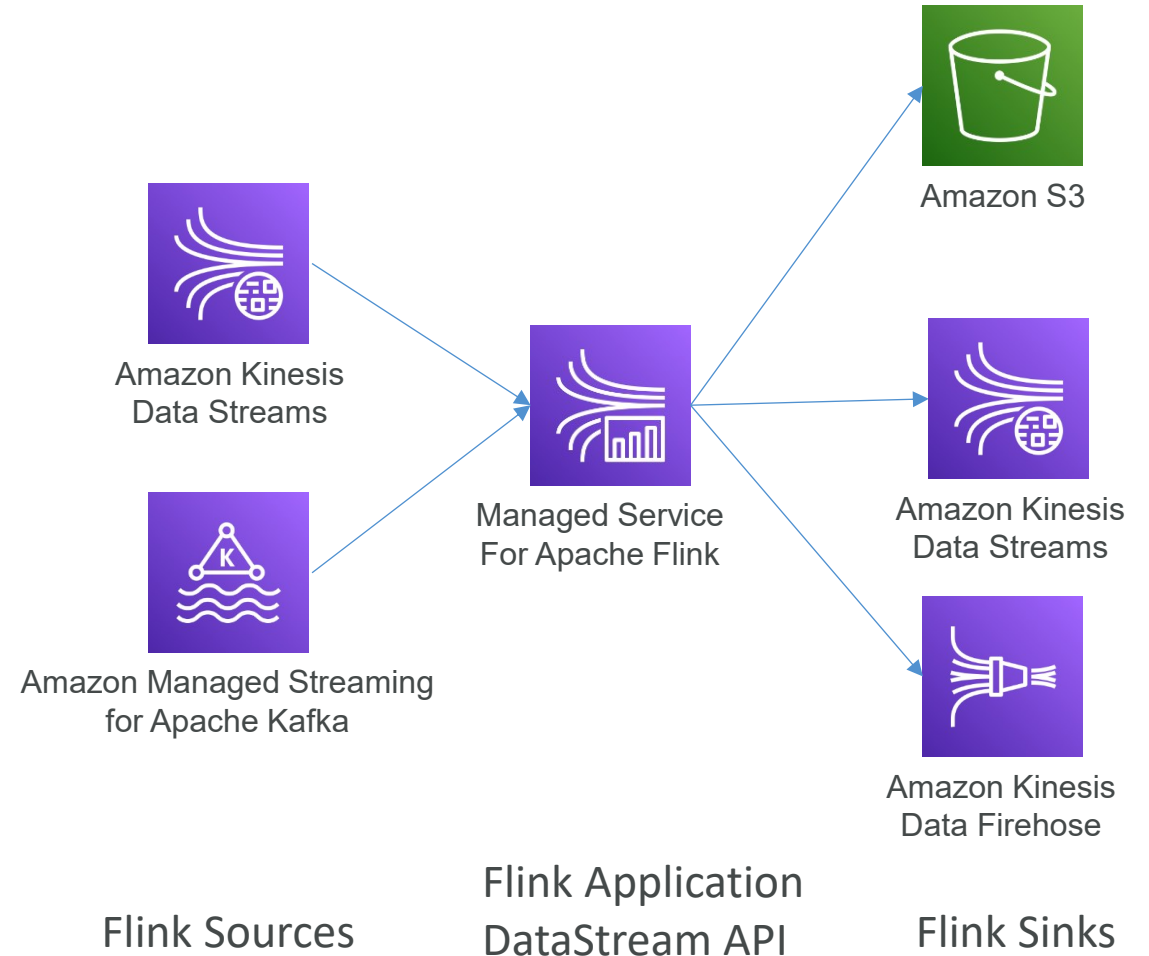


Kinesis Data Analytics + Lambda

- AWS Lambda can be a destination as well
- Allows lots of flexibility for post-processing
 - Aggregating rows
 - Translating to different formats
 - Transforming and enriching data
 - Encryption
- Opens up access to other services & destinations
 - S3, DynamoDB, Aurora, Redshift, SNS, SQS, CloudWatch

Managed Service for Apache Flink

- Formerly Kinesis Data Analytics for Apache Flink or for Java
 - Kinesis Data Analytics always used Flink under the hood
 - But now supports Python and Scala
 - Flink is a framework for processing data streams
- MSAF integrates Flink with AWS
 - Instead of using SQL, you can develop your own Flink application from scratch and load it into MSAF via S3
- In addition to the DataStream API, there is a Table API for SQL access
- Serverless



Common use-cases

- Streaming ETL
- Continuous metric generation
- Responsive analytics



**Amazon
Kinesis Data
Analytics**

Kinesis Analytics

- Pay only for resources consumed (but it's not cheap)
 - Charged by Kinesis Processing Units (KPU's) consumed per hour
 - 1 KPU = 1 vCPU + 4GB
- Serverless; scales automatically
- Use IAM permissions to access streaming source and destination(s)
- Schema discovery

RANDOM_CUT_FOREST

- SQL function used for anomaly detection on numeric columns in a stream
- They're especially proud of this because they published a paper on it
- It's a novel way to identify outliers in a data set so you can handle them however you need to
- Example: detect anomalous subway ridership during the NYC marathon

Robust Random Cut Forest Based Anomaly Detection On Streams

Sudipto Guha

University of Pennsylvania, Philadelphia, PA 19104.

SUDIPTO@CIS.UPENN.EDU

Nina Mishra

Amazon, Palo Alto, CA 94303.

NMISHRA@AMAZON.COM

Gourav Roy

Amazon, Bangalore, India 560055.

GOURAVR@AMAZON.COM

Okke Schrijvers

Stanford University, Palo Alto, CA 94305.

OKKES@CS.STANFORD.EDU

Abstract

In this paper we focus on the anomaly detection problem for dynamic data streams through the lens of random cut forests. We investigate a robust random cut data structure that can be used as a sketch or synopsis of the input stream. We provide a plausible definition of non-parametric anomalies based on the influence of an unseen point on the remainder of the data, i.e., the externality imposed by that point. We show how the sketch can be efficiently updated in a dynamic data stream. We demonstrate the viability of the algorithm on publicly available real data.

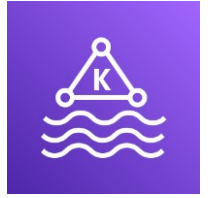
a point is data dependent and corresponds to the externality imposed by the point in explaining the remainder of the data. We extend this notion of externality to handle "outlier masking" that often arises from duplicates and near duplicate records. Note that the notion of model complexity has to be amenable to efficient computation in dynamic data streams. This relates question (1) to question (2) which we discuss in greater detail next. However it is worth noting that anomaly detection is not well understood even in the simpler context of static batch processing and (2) remains relevant in the batch setting as well.

For question (2), we explore a randomized approach, akin to (Liu et al., 2012), due in part to the practical success reported in (Emmott et al., 2013). Randomization is a powerful tool and known to be valuable in supervised learning (Breiman, 2001). But its technical exploration in the context of anomaly detection is not well-understood and the same comment applies to the algorithm put forth in (Liu et al., 2012). Moreover that algorithm has several limitations as described in Section 4.1. In particular, we show that in the presence of irrelevant dimensions, crucial anomalies are missed. In addition, it is unclear how to extend this work to a stream. Prior work attempted solutions (Tan et al., 2011) that extend to streaming, however those were not found to be effective (Emmott et al., 2013). To address these limitations, we put forward a sketch or synopsis termed *robust random cut forest* (RRCF) formally

1. Introduction

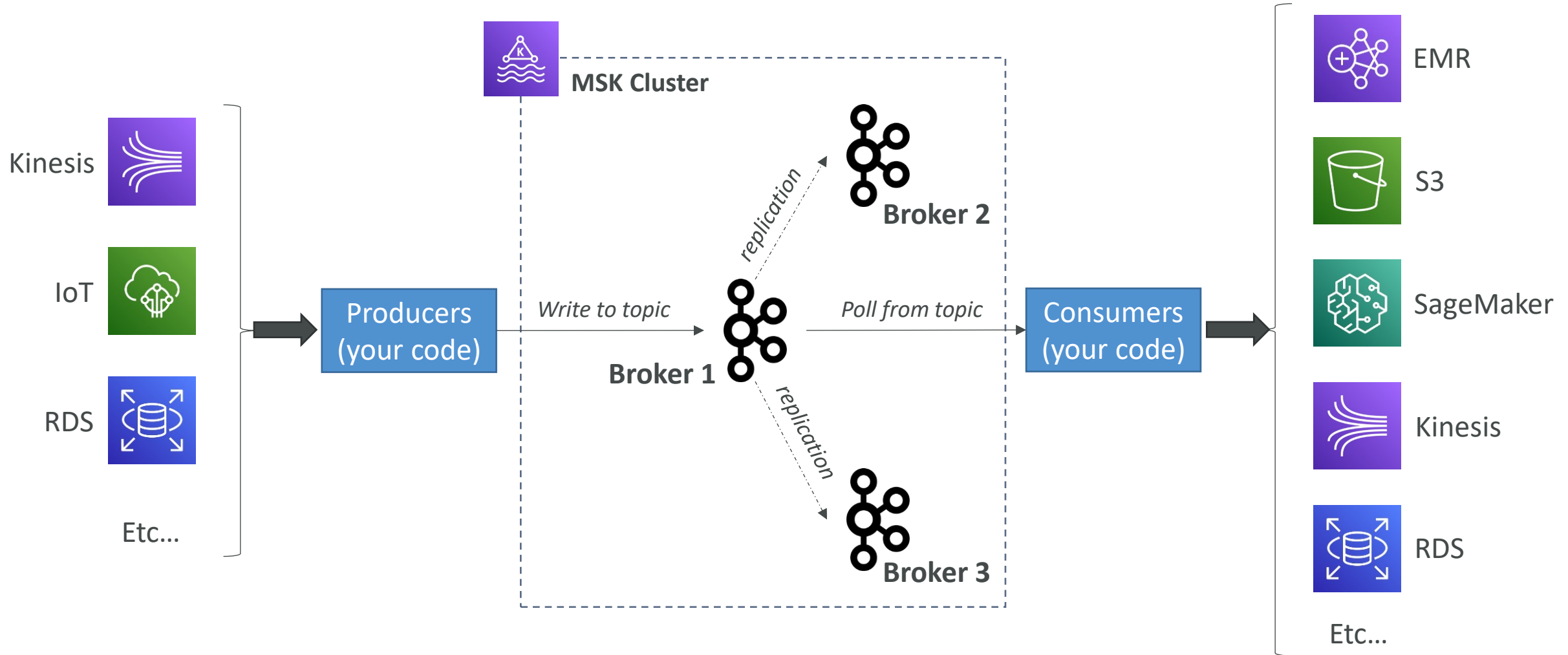
Anomaly detection is one of the cornerstone problems in data mining. Even though the problem has been well studied over the last few decades, the emerging explosion of data from the internet of things and sensors leads us to reconsider the problem. In most of these contexts the data is streaming and well-understood prior models do not exist. Furthermore the input streams need not be append only, there may be corrections, updates and a variety of other dynamic changes. Two central questions in this regard are (1) how do we define anomalies? and (2) what data struc-

Amazon Managed Streaming for Apache Kafka (Amazon MSK)



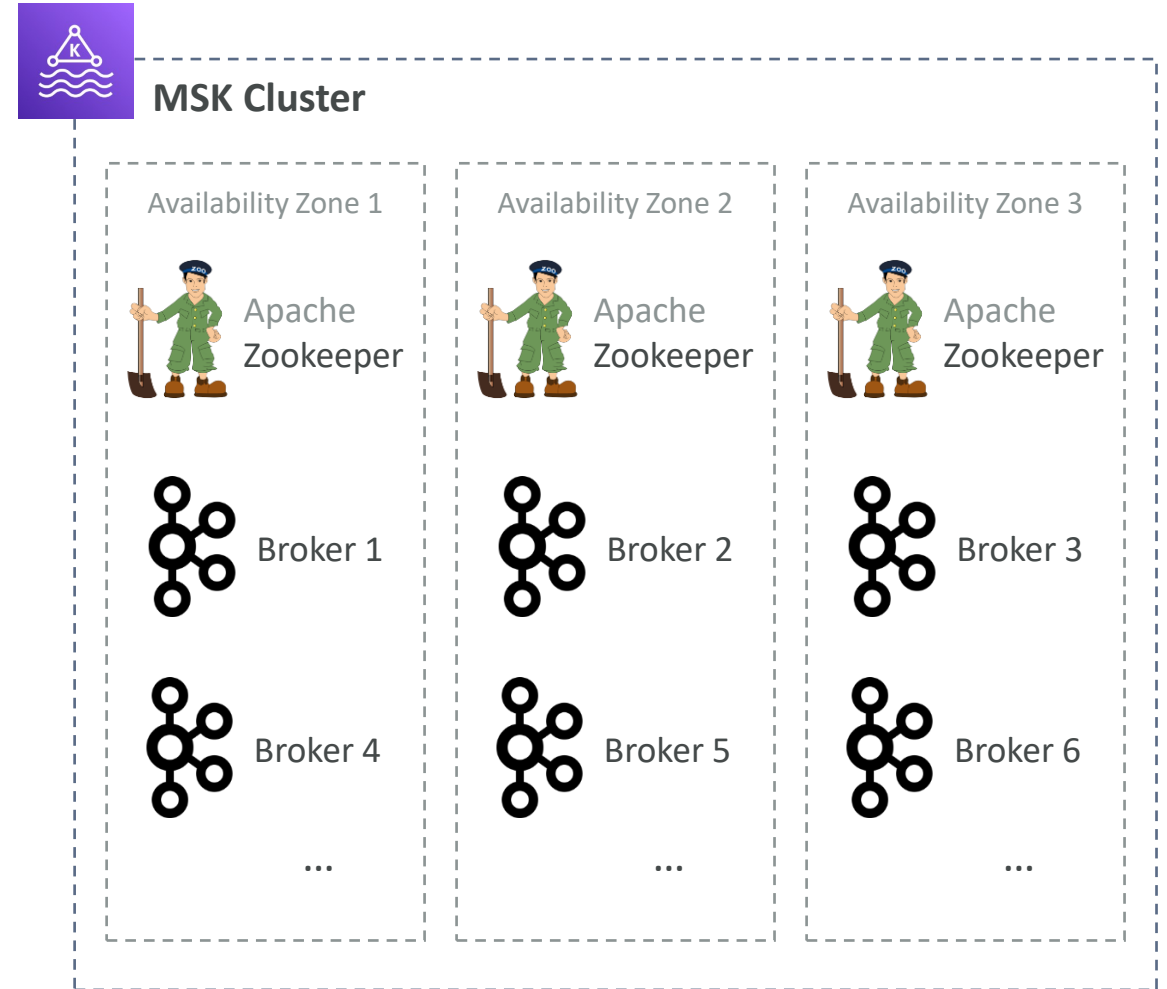
- Alternative to Kinesis (Kafka vs Kinesis next lecture)
- Fully managed Apache Kafka on AWS
 - Allow you to create, update, delete clusters
 - MSK creates & manages Kafka brokers nodes & Zookeeper nodes for you
 - Deploy the MSK cluster in your VPC, multi-AZ (up to 3 for HA)
 - Automatic recovery from common Apache Kafka failures
 - Data is stored on EBS volumes
- You can build producers and consumers of data
- Can create custom configurations for your clusters
 - Default message size of 1MB
 - **Possibilities of sending large messages (ex: 10MB) into Kafka after custom configuration**

Apache Kafka at a high level



MSK – Configurations

- Choose the number of AZ (3 – recommended, or 2)
- Choose the VPC & Subnets
- The broker instance type (ex: kafka.m5.large)
- The number of brokers per AZ (can add brokers later)
- Size of your EBS volumes (1GB – 16TB)



MSK – Security

- **Encryption:**

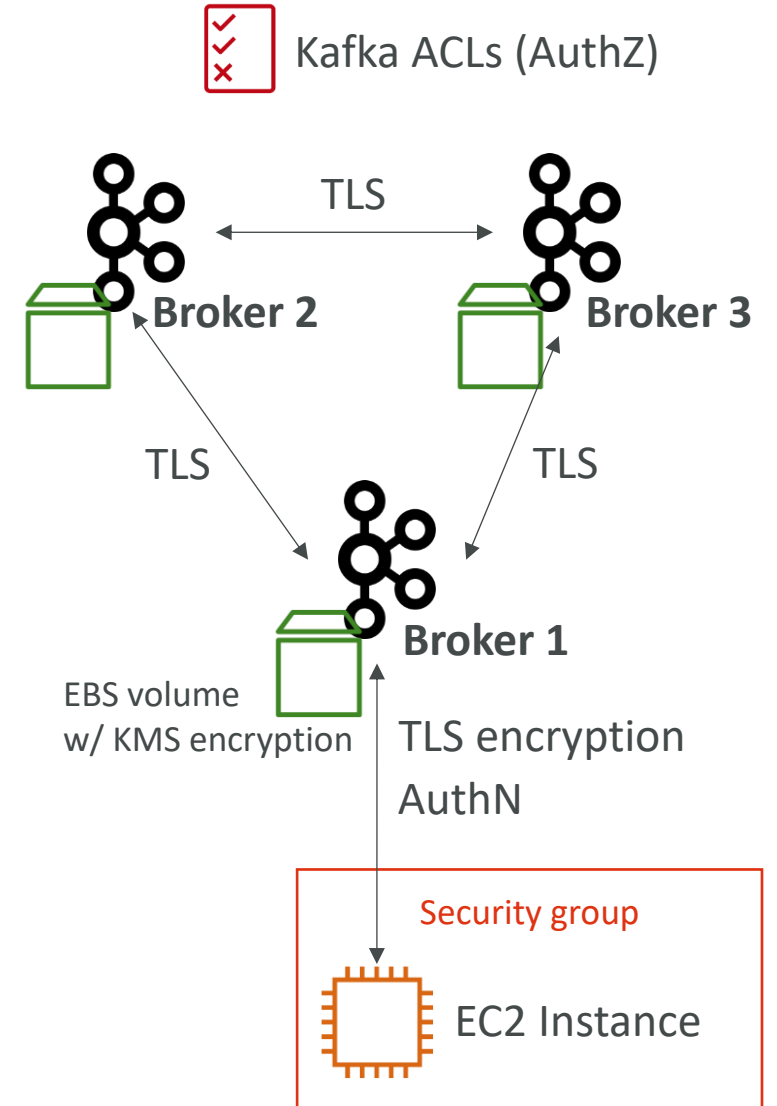
- Optional in-flight using TLS between the brokers
- Optional in-flight with TLS between the clients and brokers
- At rest for your EBS volumes using KMS

- **Network Security:**

- Authorize specific security groups for your Apache Kafka clients

- **Authentication & Authorization (important):**

- Define who can read/write to which topics
- Mutual TLS (AuthN) + Kafka ACLs (AuthZ)
- SASL/SCRAM (AuthN) + Kafka ACLs (AuthZ)
- IAM Access Control (AuthN + AuthZ)



MSK – Monitoring

- **CloudWatch Metrics**

- Basic monitoring (cluster and broker metrics)
- Enhanced monitoring (++enhanced broker metrics)
- Topic-level monitoring (++enhanced topic-level metrics)

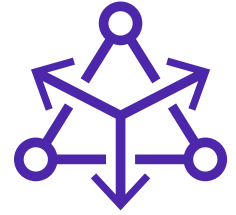
- **Prometheus (Open-Source Monitoring)**

- Opens a port on the broker to export cluster, broker and topic-level metrics
- Setup the JMX Exporter (metrics) or Node Exporter (CPU and disk metrics)

- **Broker Log Delivery**

- Delivery to CloudWatch Logs
- Delivery to Amazon S3
- Delivery to Kinesis Data Streams

MSK Connect



- Managed Kafka Connect workers on AWS
- Auto-scaling capabilities for workers
- You can deploy any Kafka Connect connectors to MSK Connect as a **plugin**
 - Amazon S3, Amazon Redshift, Amazon OpenSearch, Debezium, etc...
- Example pricing: Pay \$0.11 per worker per hour



MSK Serverless

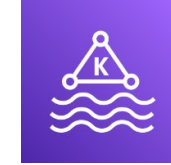
- **Run Apache Kafka on MSK without managing the capacity**
- **MSK automatically provisions resources and scales compute & storage**
- You just define your topics and your partitions and you're good to go!
- Security: IAM Access Control for all clusters
- Example Pricing:
 - \$0.75 per cluster per hour = \$558 monthly per cluster
 - \$0.0015 per partition per hour = \$1.08 monthly per partition
 - \$0.10 per GB of storage each month
 - \$0.10 per GB in
 - \$0.05 per GB out

Kinesis Data Streams vs Amazon MSK



Kinesis Data Streams

- 10 MB message size limit
- Data Streams with Shards
- Shard Splitting & Merging
- TLS In-flight encryption
- KMS At-rest encryption
- Security:
 - IAM policies for AuthN/AuthZ



Amazon MSK

- 1MB default, configure for higher (ex: 10MB)
- Kafka Topics with Partitions
- Can only add partitions to a topic
- PLAINTEXT or TLS In-flight Encryption
- KMS At-rest encryption
- Security:
 - Mutual TLS (AuthN) + Kafka ACLs (AuthZ)
 - SASL/SCRAM (AuthN) + Kafka ACLs (AuthZ)
 - IAM Access Control (AuthN + AuthZ)

Data Transformation, Integrity, and Feature Engineering

EMR

Elastic MapReduce

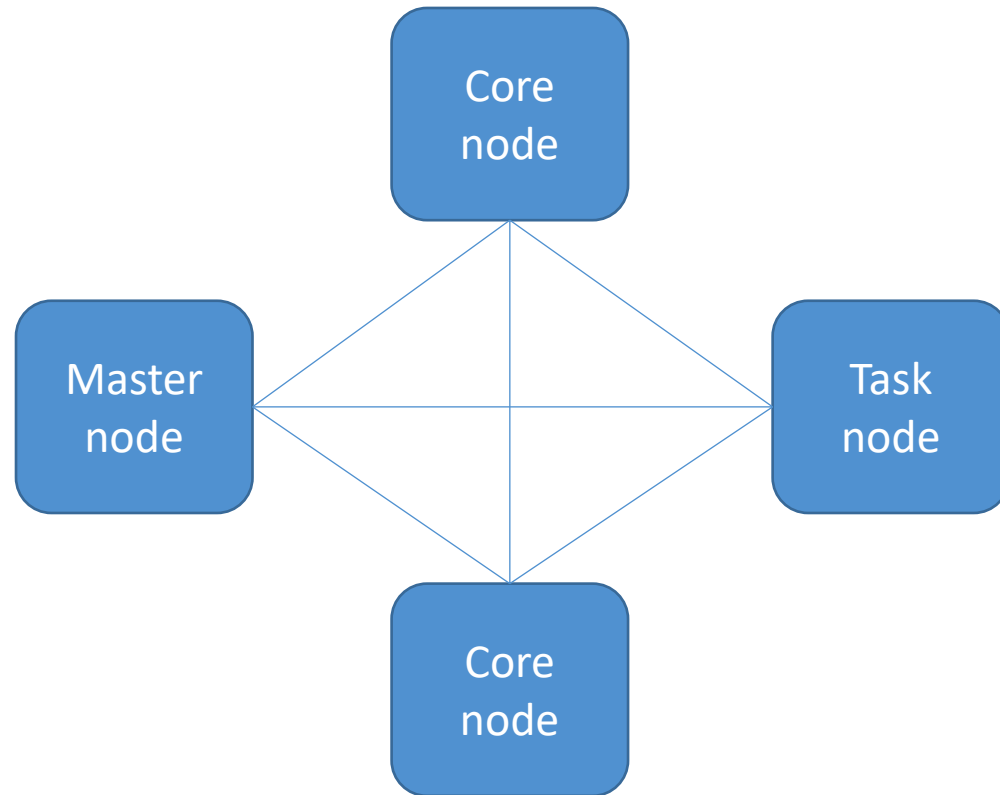
What is EMR?

- Elastic MapReduce
- Managed Hadoop framework on EC2 instances
- Includes Spark, HBase, Presto, Flink, Hive & more
- EMR Notebooks
- Several integration points with AWS



Amazon EMR

An EMR Cluster



- **Master node:** manages the cluster
 - Single EC2 instance
- **Core node:** Hosts HDFS data and runs tasks
 - Can be scaled up & down, but with some risk
- **Task node:** Runs tasks, does not host data
 - No risk of data loss when removing
 - Good use of **spot instances**

EMR Usage

- Transient vs Long-Running Clusters
 - Can spin up task nodes using Spot instances for temporary capacity
 - Can use reserved instances on long-running clusters to save \$
- Connect directly to master to run jobs
- Submit ordered steps via the console
- EMR Serverless lets AWS scale your nodes automatically

EMR / AWS Integration

- Amazon EC2 for the instances that comprise the nodes in the cluster
- Amazon VPC to configure the virtual network in which you launch your instances
- Amazon S3 to store input and output data
- Amazon CloudWatch to monitor cluster performance and configure alarms
- AWS IAM to configure permissions
- AWS CloudTrail to audit requests made to the service
- AWS Data Pipeline to schedule and start your clusters

EMR Storage

- HDFS
- EMRFS: access S3 as if it were HDFS
 - **EMRFS Consistent View** – Optional for S3 consistency
 - Uses DynamoDB to track consistency
- Local file system
- EBS for HDFS



EMR promises

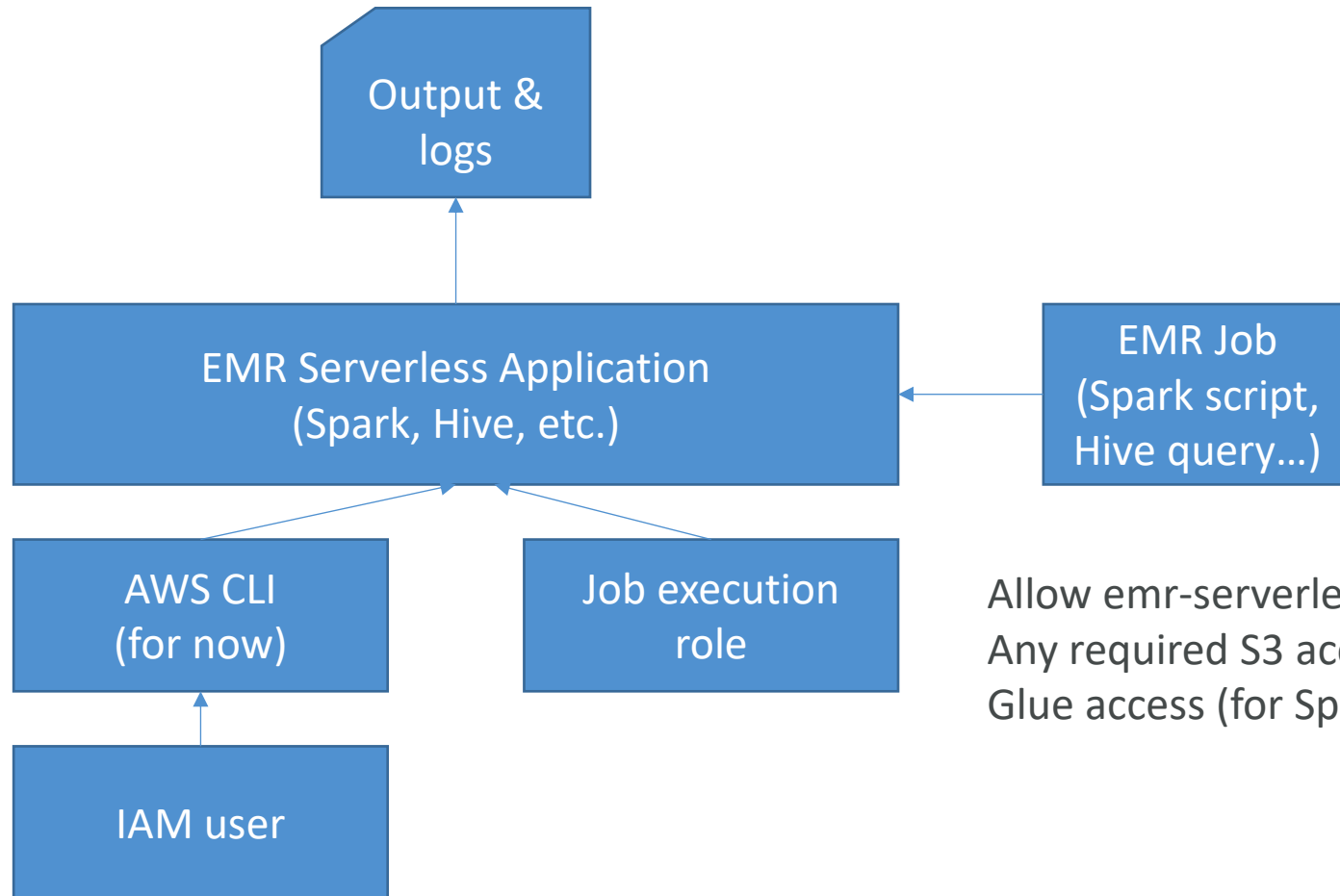
- EMR charges by the hour
 - Plus EC2 charges
- Provisions new nodes if a core node fails
- Can add and remove tasks nodes on the fly
- Can resize a running cluster's core nodes



EMR Serverless

- Choose an EMR Release and Runtime (Spark, Hive, Presto)
- Submit queries / scripts via job run requests
- EMR manages underlying capacity
 - But you can specify default worker sizes & pre-initialized capacity
 - EMR computes resources needed for your job & schedules workers accordingly
 - All within one region (across multiple AZ's)
- Why is this a big deal?
 - You no longer have to estimate how many workers are needed for your workloads – they are provisioned as needed, automatically.
- Serverless? Really?
 - TBH you still need to think about worker nodes and how they are configured.

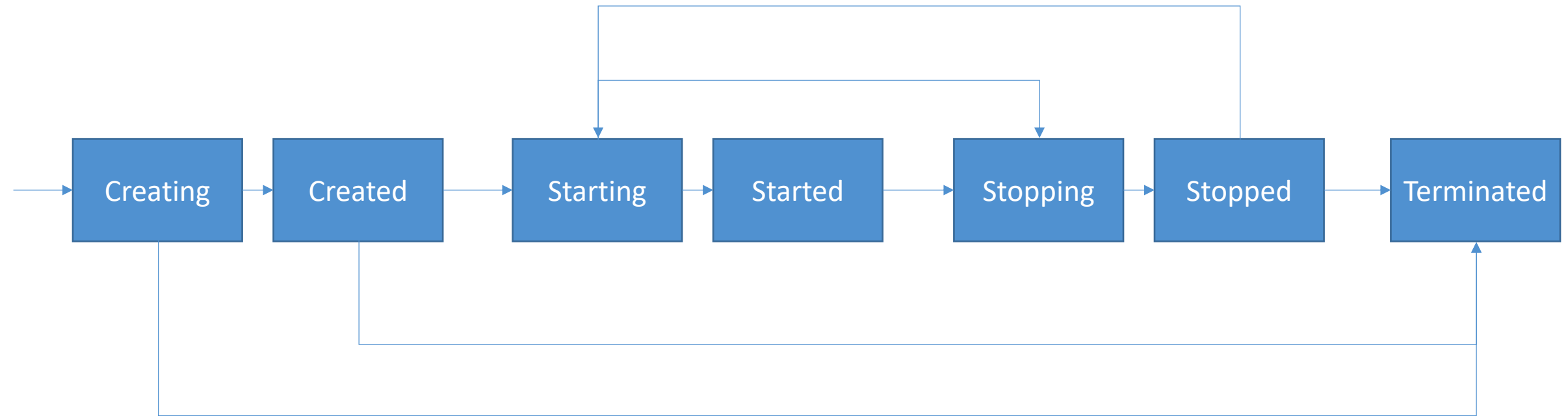
Using EMR Serverless



```
aws emr-serverless start-job-run \
  --application-id <application_id> \
  --execution-role-arn <execution_role_arn> \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://us-east-1.elasticmapreduce/emr-
containers/samples/wordcount/scripts/wordcount.py",
      "entryPointArguments": ["s3://DOC-EXAMPLE-
BUCKET/output"],
      "sparkSubmitParameters": "--conf spark.executor.cores=1 --
conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
spark.driver.memory=4g --conf spark.executor.instances=1"
    }
  }' \
  --configuration-overrides '{
    "monitoringConfiguration": {
      "s3MonitoringConfiguration": {
        "logUri": "s3://DOC-EXAMPLE-BUCKET/logs"
      }
    }
  }'
```

Allow `emr-serverless.amazonaws.com` service
Any required S3 access for scripts & data
Glue access (for SparkSQL), KMS keys

EMR Serverless Application Lifecycle



This isn't all automatic!

Must call API's such as CreateApplication, StartApplication, StopApplication, and importantly DeleteApplication to avoid excess charges.

Pre-Initialized Capacity

- Spark adds 10% overhead to memory requested for drivers & executors
- Be sure initial capacity is at least 10% more than requested by the job

```
aws emr-serverless create-application \  
  --type "SPARK" \  
  --name <"my_application_name"> \  
  --release-label "emr-6.5.0-preview" \  
  --initial-capacity '{  
    "DRIVER": {  
      "workerCount": 5,  
      "resourceConfiguration": {  
        "cpu": "2vCPU",  
        "memory": "4GB"  
      }  
    },  
    "EXECUTOR": {  
      "workerCount": 50,  
      "resourceConfiguration": {  
        "cpu": "4vCPU",  
        "memory": "8GB"  
      }  
    }  
  } \  
  --maximum-capacity '{  
    "cpu": "400vCPU",  
    "memory": "1024GB"  
  }'
```

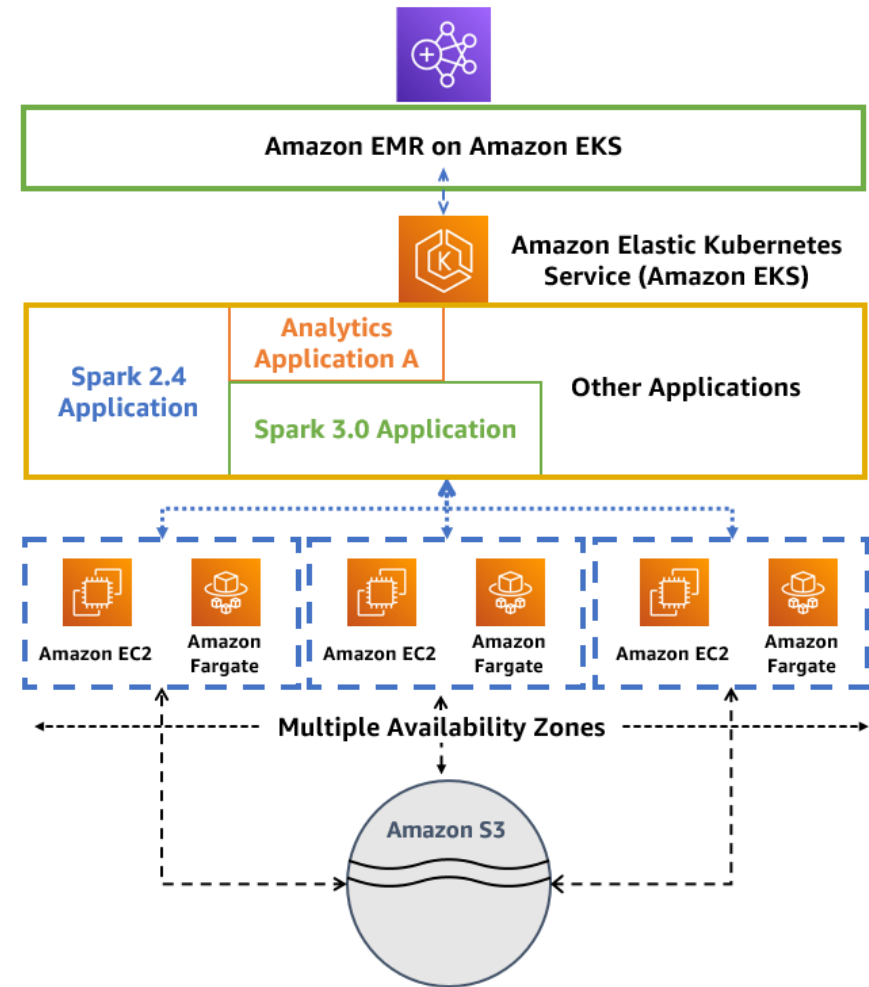
EMR Serverless Security

- Basically the same as EMR
- EMRFS
 - S3 encryption (SSE or CSE) at rest
 - TLS in transit between EMR nodes and S3
- S3
 - SSE-S3, SSE-KMS
- Local disk encryption
- Spark communication between drivers & executors is encrypted
- Hive communication between Glue Metastore and EMR uses TLS
- Force HTTPS (TLS) on S3 policies with `aws:SecureTransport`



EMR on EKS

- Allows submitting Spark job on Elastic Kubernetes Service without provisioning clusters
- Fully managed
- Share resources between Spark and other apps on Kubernetes



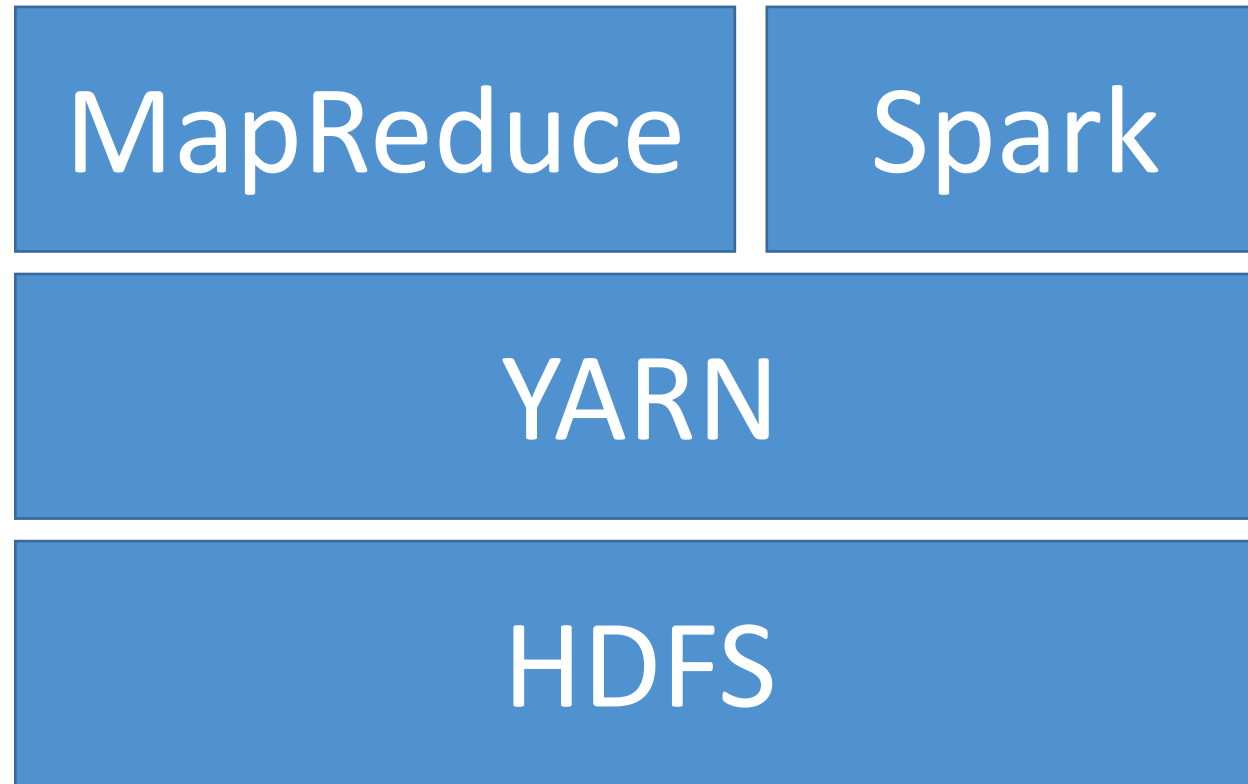
So... what's Hadoop?

MapReduce

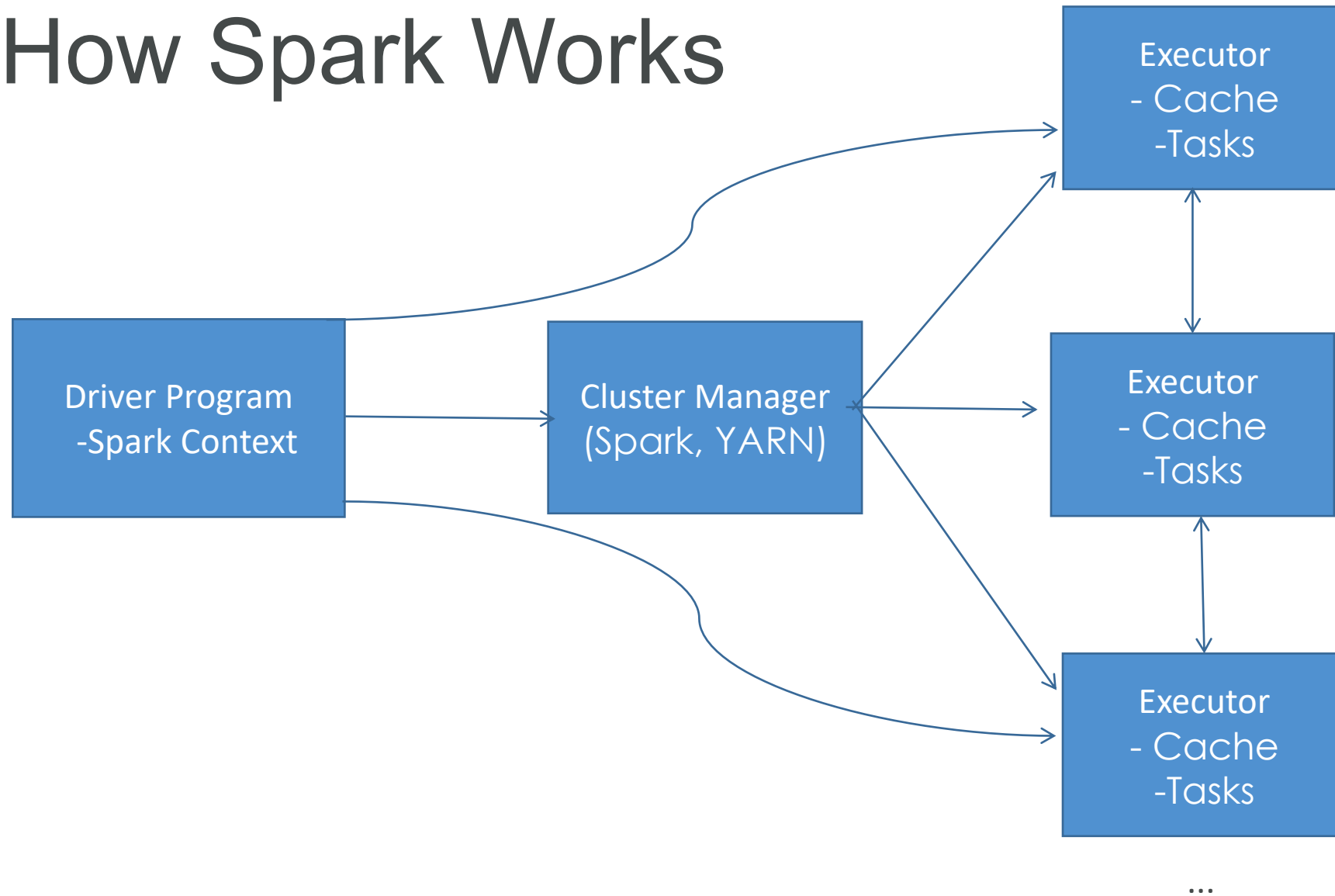
YARN

HDFS

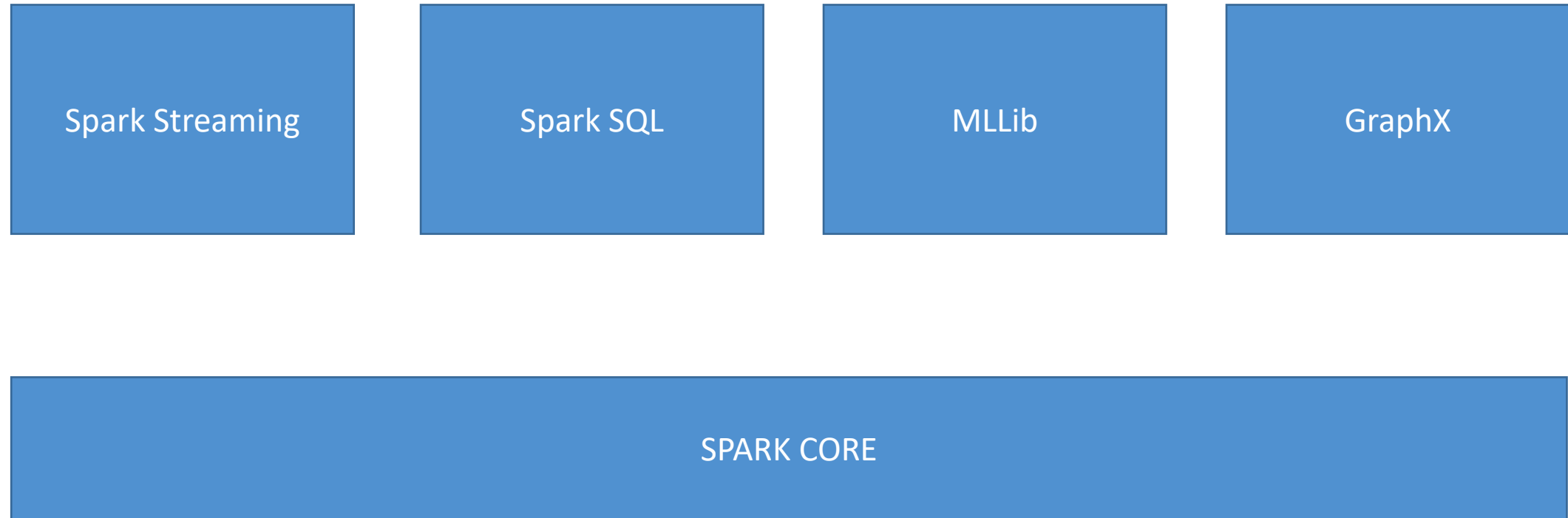
Apache Spark



How Spark Works



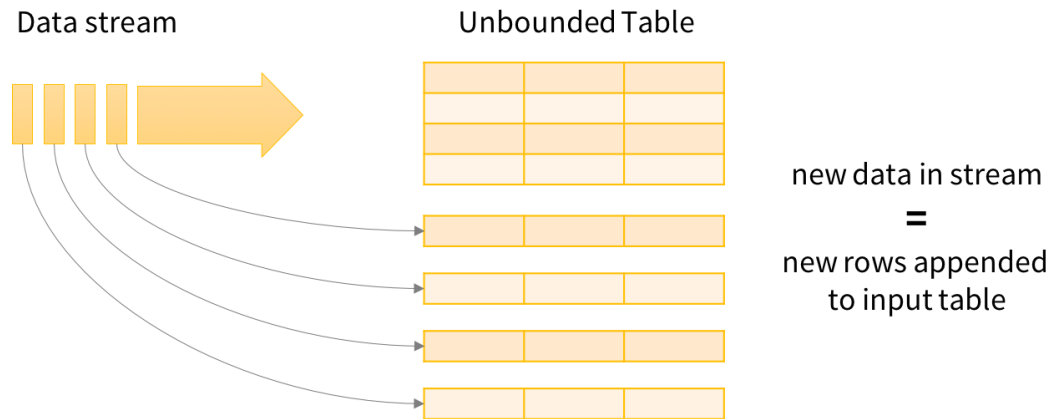
Spark Components



Spark MLlib

- Classification: logistic regression, naïve Bayes
- Regression
- Decision trees
- Recommendation engine (ALS)
- Clustering (K-Means)
- LDA (topic modeling)
- ML workflow utilities (pipelines, feature transformation, persistence)
- SVD, PCA, statistics

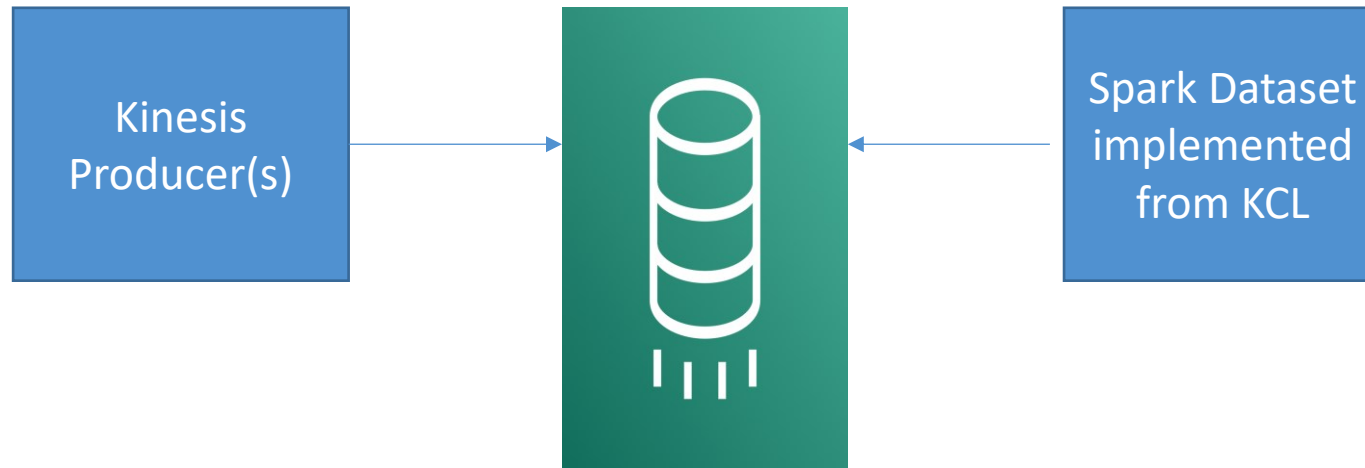
Spark Structured Streaming



Data stream as an unbounded Input Table

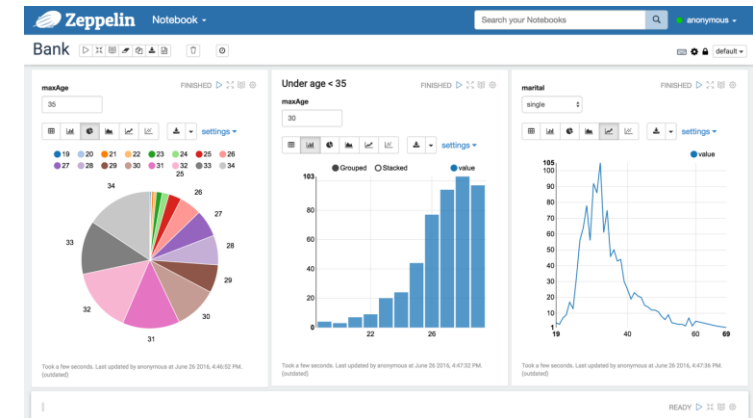
```
val inputDF = spark.readStream.json("s3://logs")
inputDF.groupBy($"action", window($"time", "1 hour")).count()
.writeStream.format("jdbc").start("jdbc:mysql://...")
```

Spark Streaming + Kinesis



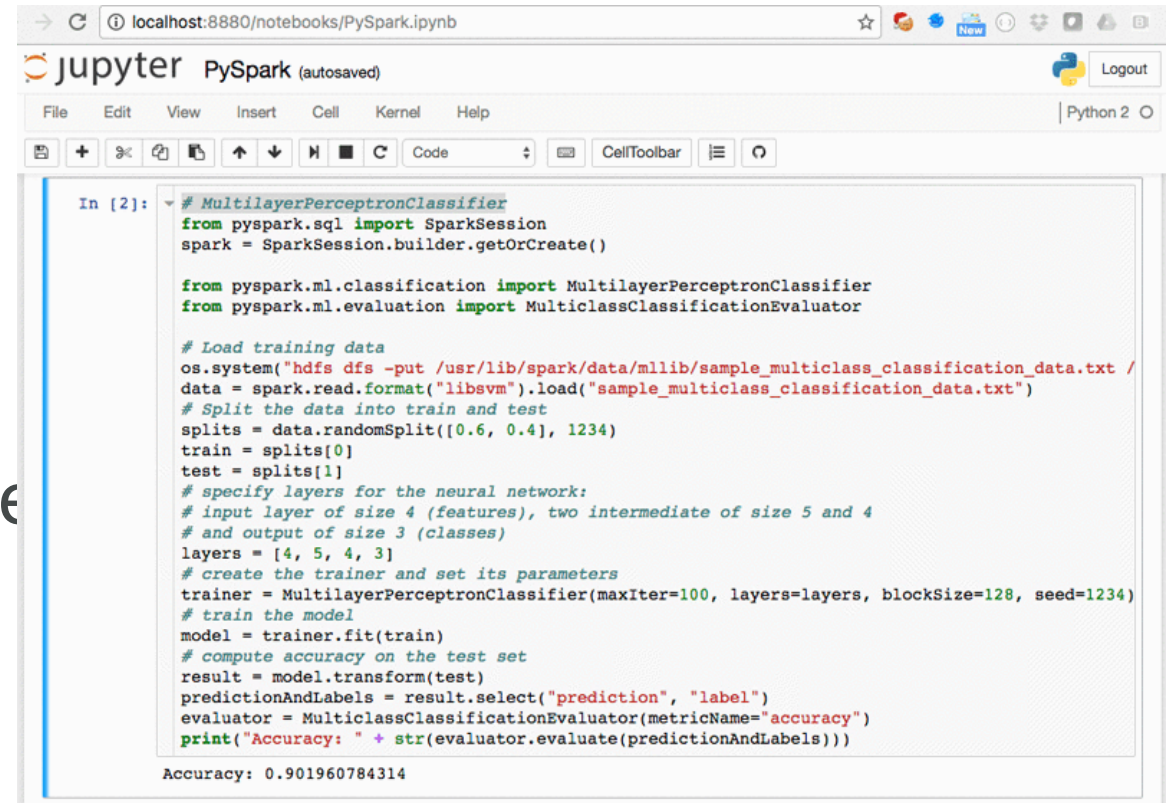
Zeppelin + Spark

- Can run Spark code interactively (like you can in the Spark shell)
 - This speeds up your development cycle
 - And allows easy experimentation and exploration of your big data
- Can execute SQL queries directly against SparkSQL
- Query results may be visualized in charts and graphs
- Makes Spark feel more like a data science tool!



EMR Notebook

- Similar concept to Zeppelin, with more AWS integration
- Notebooks backed up to S3
- Provision clusters from the notebook!
- Hosted inside a VPC
- Accessed only via AWS console



```
In [2]: # MultilayerPerceptronClassifier
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

from pyspark.ml.classification import MultilayerPerceptronClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Load training data
os.system("hdfs dfs -put /usr/lib/spark/data/mllib/sample_multiclass_classification_data.txt /")
data = spark.read.format("libsvm").load("sample_multiclass_classification_data.txt")
# Split the data into train and test
splits = data.randomSplit([0.6, 0.4], 1234)
train = splits[0]
test = splits[1]
# specify layers for the neural network:
# input layer of size 4 (features), two intermediate of size 5 and 4
# and output of size 3 (classes)
layers = [4, 5, 4, 3]
# create the trainer and set its parameters
trainer = MultilayerPerceptronClassifier(maxIter=100, layers=layers, blockSize=128, seed=1234)
# train the model
model = trainer.fit(train)
# compute accuracy on the test set
result = model.transform(test)
predictionAndLabels = result.select("prediction", "label")
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
print("Accuracy: " + str(evaluator.evaluate(predictionAndLabels)))

Accuracy: 0.901960784314
```

EMR Security

- IAM policies
- Kerberos
- SSH
- IAM roles
- Security configurations may be specified for Lake Formation
- Native integration with Apache Ranger
 - For data security on Hadoop / Hive



EMR: Choosing Instance Types

- Master node:
 - m4.large if < 50 nodes, m4.xlarge if > 50 nodes
- Core & task nodes:
 - m4.large is usually good
 - If cluster waits a lot on external dependencies (i.e. a web crawler), t2.medium
 - Improved performance: m4.xlarge
 - Computation-intensive applications: high CPU instances
 - Database, memory-caching applications: high memory instances
 - Network / CPU-intensive (NLP, ML) – cluster computer instances
 - Accelerated Computing / AI – GPU instances (g3, g4, p2, p3)
- Spot instances
 - Good choice for task nodes
 - Only use on core & master if you're testing or very cost-sensitive; you're risking partial data loss

Feature Engineering

What is feature engineering?

- Applying your knowledge of the data – and the model you’re using – to create better features to train your model with.
 - Which features should I use?
 - Do I need to transform these features in some way?
 - How do I handle missing data?
 - Should I create new features from the existing ones?
- You can’t just throw in raw data and expect good results
- This is the art of machine learning; where expertise is applied
- “Applied machine learning is basically feature engineering” – Andrew Ng

The Curse of Dimensionality

- Too many features can be a problem – leads to sparse data
- Every feature is a new dimension
- Much of feature engineering is selecting the features most relevant to the problem at hand
 - This often is where domain knowledge comes into play
- Unsupervised dimensionality reduction techniques can also be employed to distill many features into fewer features
 - PCA
 - K-Means



Lab: Preparing Data for TF-IDF on Spark and EMR

TF-IDF

- Stands for *Term Frequency* and *Inverse Document Frequency*
- Important data for search – figures out what terms are most relevant for a document
- Sounds fancy!



TF-IDF Explained

- *Term Frequency* just measures how often a word occurs in a document
 - A word that occurs frequently is probably important to that document's meaning
- *Document Frequency* is how often a word occurs in an entire set of documents, i.e., all of Wikipedia or every web page
 - This tells us about common words that just appear everywhere no matter what the topic, like “a”, “the”, “and”, etc.

TF-IDF Explained

- So a measure of the relevancy of a word to a document might be:

$$\frac{\textit{Term Frequency}}{\textit{Document Frequency}}$$

Or: Term Frequency * Inverse Document Frequency

That is, take how often the word appears in a document, over how often it just appears everywhere. That gives you a measure of how important and unique this word is for this document

TF-IDF In Practice

- We actually use the log of the IDF, since word frequencies are distributed exponentially. That gives us a better weighting of a words overall popularity
- TF-IDF assumes a document is just a “bag of words”
 - Parsing documents into a bag of words can be most of the work
 - Words can be represented as a hash value (number) for efficiency
 - What about synonyms? Various tenses? Abbreviations? Capitalizations? Misspellings?
- Doing this at scale is the hard part
 - That’s where Spark comes in!

Unigrams, bigrams, etc.

- An extension of TF-IDF is to not only compute relevancy for individual words (terms) but also for *bi-grams* or, more generally, *n-grams*.
- “I love certification exams”
 - Unigrams: “I”, “love”, “certification”, “exams”
 - Bi-grams: “I love”, “love certification”, “certification exams”
 - Tri-grams: “I love certification”, “love certification exams”

Sample TF-IDF matrix with unigrams and bigrams

	I	Love	Certification	Exams	Puppies	I love	Love certification	Love puppies	Certification exams
"I love certification exams"									
"I love puppies"									

Using TF-IDF

- A very simple search algorithm could be:
 - Compute TF-IDF for every word in a corpus
 - For a given search word, sort the documents by their TF-IDF score for that word
 - Display the results

Let's use TF-IDF on Wikipedia

WIKIPEDIA
The Free Encyclopedia



Imputing Missing Data: Mean Replacement

- Replace missing values with the mean value from the rest of the column (columns, not rows! A column represents a single feature; it only makes sense to take the mean from other samples of the same feature.)
- Fast & easy, won't affect mean or sample size of overall data set
- Median may be a better choice than mean when outliers are present
- But it's generally pretty terrible.
 - Only works on column level, misses correlations between features
 - Can't use on categorical features (imputing with most frequent value can work in this case, though)
 - Not very accurate

```
In [3]: import pandas as pd

masses_data = pd.read_csv('mammographic_masses.data.txt',
masses_data.head()
```

```
Out[3]:
```

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.0	1
1	4.0	43.0	1.0	1.0	NaN	1
2	5.0	58.0	4.0	5.0	3.0	1
3	4.0	28.0	1.0	1.0	3.0	0
4	5.0	74.0	1.0	5.0	NaN	1

```
In [6]: mean_imputed = masses_data.fillna(masses_data.mean())
mean_imputed.head()
```

```
Out[6]:
```

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.000000	1
1	4.0	43.0	1.0	1.0	2.910734	1
2	5.0	58.0	4.0	5.0	3.000000	1
3	4.0	28.0	1.0	1.0	3.000000	0
4	5.0	74.0	1.0	5.0	2.910734	1

Imputing Missing Data: Dropping

- If not many rows contain missing data...
 - ...and dropping those rows doesn't bias your data...
 - ...and you don't have a lot of time...
 - ...maybe it's a reasonable thing to do.
- But, it's never going to be the right answer for the “best” approach.
- Almost anything is better. Can you substitute another similar field perhaps? (i.e., review summary vs. full text)

```
In [3]: import pandas as pd

masses_data = pd.read_csv('mammographic_masses.data')
masses_data.head()
```

```
Out[3]:
```

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.0	1
1	4.0	43.0	1.0	1.0	NaN	1
2	5.0	58.0	4.0	5.0	3.0	1
3	4.0	28.0	1.0	1.0	3.0	0
4	5.0	74.0	1.0	5.0	NaN	1

```
In [7]: mean_imputed = masses_data.dropna()
mean_imputed.head()
```

```
Out[7]:
```

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.0	1
2	5.0	58.0	4.0	5.0	3.0	1
3	4.0	28.0	1.0	1.0	3.0	0
8	5.0	57.0	1.0	5.0	3.0	1
10	5.0	76.0	1.0	4.0	3.0	1

Imputing Missing Data: Machine Learning

- KNN: Find K “nearest” (most similar) rows and average their values
 - Assumes numerical data, not categorical
 - There are ways to handle categorical data (Hamming distance), but categorical data is probably better served by...
- Deep Learning
 - Build a machine learning model to impute data for your machine learning model!
 - Works well for categorical data. Really well. But it’s complicated.
- Regression
 - Find linear or non-linear relationships between the missing feature and other features
 - Most advanced technique: MICE (Multiple Imputation by Chained Equations)

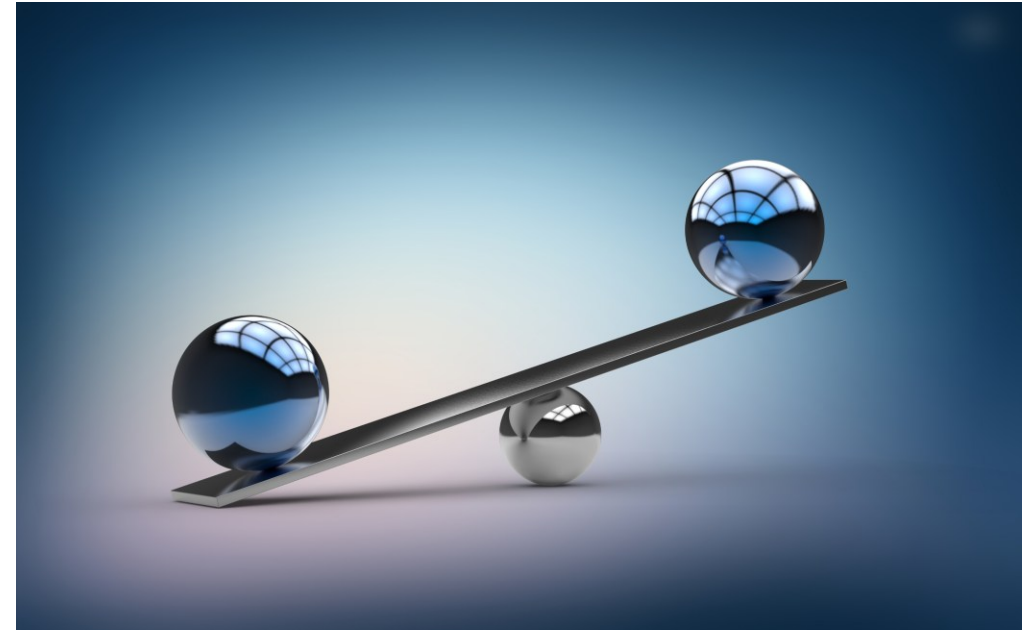
Imputing Missing Data: Just Get More Data

- What's better than imputing data? Getting more real data!
- Sometimes you just have to try harder or collect more data

Handling Unbalanced Data

What is unbalanced data?

- Large discrepancy between “positive” and “negative” cases
 - i.e., fraud detection. Fraud is rare, and most rows will be not-fraud
 - Don’t let the terminology confuse you; “positive” doesn’t mean “good”
 - It means the thing you’re testing for is what happened.
 - If your machine learning model is made to detect fraud, then fraud is the positive case.
- Mainly a problem with neural networks



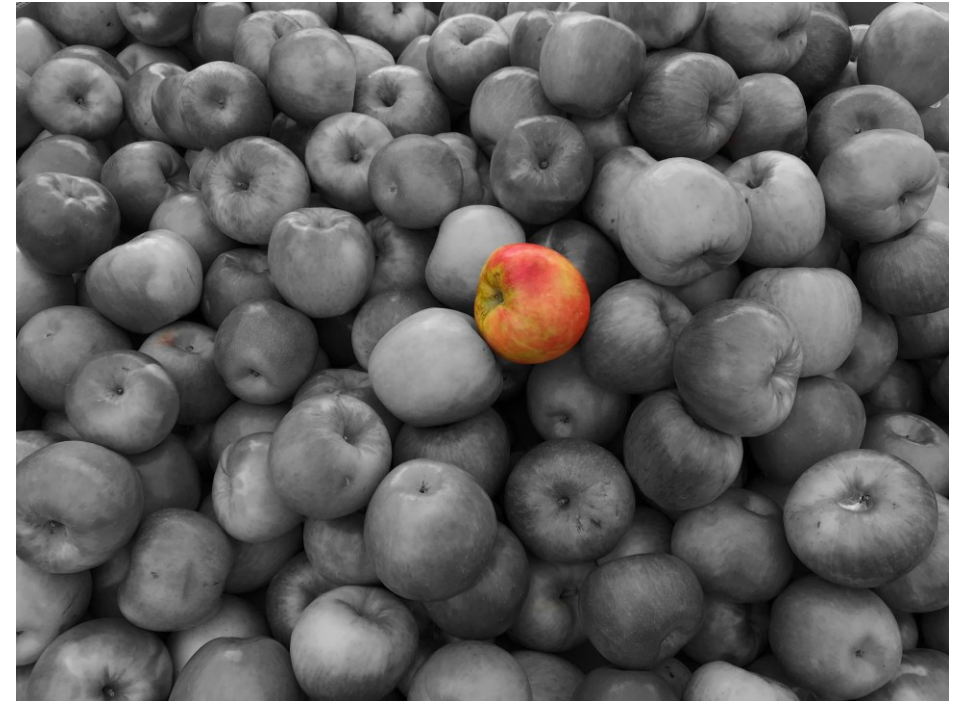
Oversampling

- Duplicate samples from the minority class
- Can be done at random



Undersampling

- Instead of creating more positive samples, remove negative ones
- Throwing data away is usually not the right answer
 - Unless you are specifically trying to avoid “big data” scaling issues



SMOTE

- Synthetic Minority Over-sampling TEchnique
- Artificially generate new samples of the minority class using nearest neighbors
 - Run K-nearest-neighbors of each sample of the minority class
 - Create a new sample from the KNN result (mean of the neighbors)
- Both generates new samples and undersamples majority class
- Generally better than just oversampling

Adjusting thresholds

- When making predictions about a classification (fraud / not fraud), you have some sort of threshold of probability at which point you'll flag something as the positive case (fraud)
- If you have too many false positives, one way to fix that is to simply increase that threshold.
 - Guaranteed to reduce false positives
 - But, could result in more false negatives



Handling Outliers

Variance measures how “spread-out” the data is.

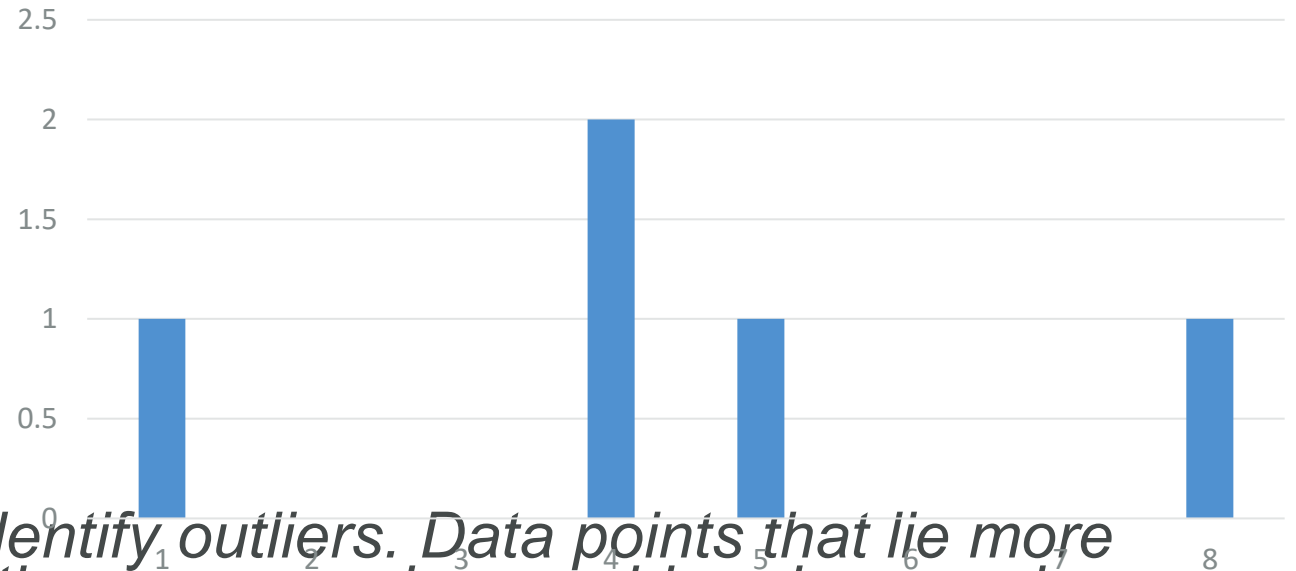
- Variance (σ^2) is simply the **average of the squared differences from the mean**
- Example: What is the variance of the data set (1, 4, 5, 4, 8)?
 - First find the mean: $(1+4+5+4+8)/5 = 4.4$
 - Now find the differences from the mean: (-3.4, -0.4, 0.6, -0.4, 3.6)
 - Find the squared differences: (11.56, 0.16, 0.36, 0.16, 12.96)
 - Find the average of the squared differences:
 - $\sigma^2 = (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 5 = 5.04$

Standard Deviation σ is just the square root of the variance.

$$\sigma^2 = 5.04$$

$$\sigma = \sqrt{5.04} = 2.24$$

So the standard deviation of (1, 4, 5, 4, 8) is 2.24.



This is usually used as a way to identify outliers. Data points that lie more than one standard deviation from the mean can be considered unusual.

You can talk about how extreme a data point is by talking about “how many sigmas” away from the mean it is.

Dealing with Outliers

- Sometimes it's appropriate to remove outliers from your training data
- Do this responsibly! Understand why you are doing this.
- For example: in collaborative filtering, a single user who rates thousands of movies could have a big effect on everyone else's ratings. That may not be desirable.
- Another example: in web log data, outliers may represent bots or other agents that should be discarded.
- But if someone really wants the mean income of US citizens for example, don't toss out billionaires just because you want to.



Dealing with Outliers

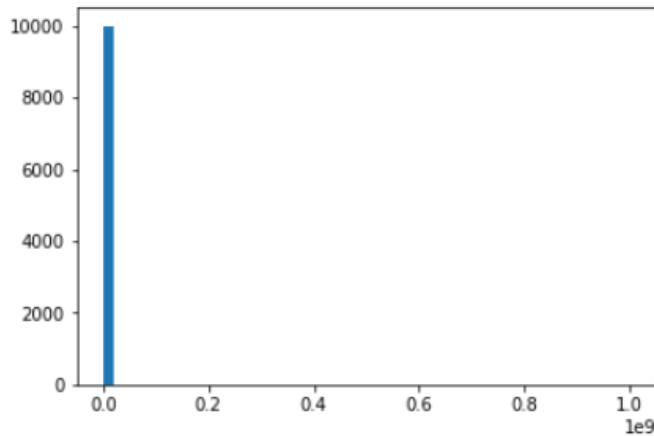
- Our old friend standard deviation provides a principled way to classify outliers.
- Find data points more than some multiple of a standard deviation in your training data.
- What multiple? You just have to use common sense.
- Remember AWS's Random Cut Forest algorithm creeps into many of its services – it is made for outlier detection
 - Found within QuickSight, Kinesis Analytics, SageMaker, and more

Example: Income Inequality

```
%matplotlib inline
import numpy as np

incomes = np.random.normal(27000, 15000, 10000)
incomes = np.append(incomes, [1000000000])

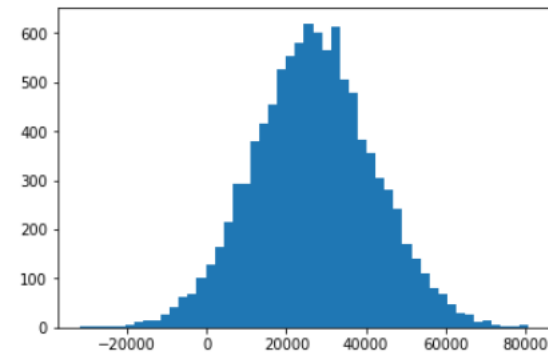
import matplotlib.pyplot as plt
plt.hist(incomes, 50)
plt.show()
```



```
def reject_outliers(data):
    u = np.median(data)
    s = np.std(data)
    filtered = [e for e in data if (u - 2 * s < e < u + 2 * s)]
    return filtered

filtered = reject_outliers(incomes)

plt.hist(filtered, 50)
plt.show()
```



That looks better. And, our mean is more, well, meaningful now as well:

```
np.mean(filtered)

26894.643431053548
```


Binning

- Bucket observations together based on ranges of values.
- Example: estimated ages of people
 - Put all 20-somethings in one classification, 30-somethings in another, etc.
- Quantile binning categorizes data by their place in the data distribution
 - Ensures even sizes of bins
- Transforms numeric data to ordinal data
- Especially useful when there is uncertainty in the measurements



Transforming

- Applying some function to a feature to make it better suited for training
- Feature data with an exponential trend may benefit from a logarithmic transform
- Example: YouTube recommendations
 - A numeric feature x is also represented by x^2 and $\sqrt{x}$
 - This allows learning of super and sub-linear functions
- (ref: <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45530.pdf>)

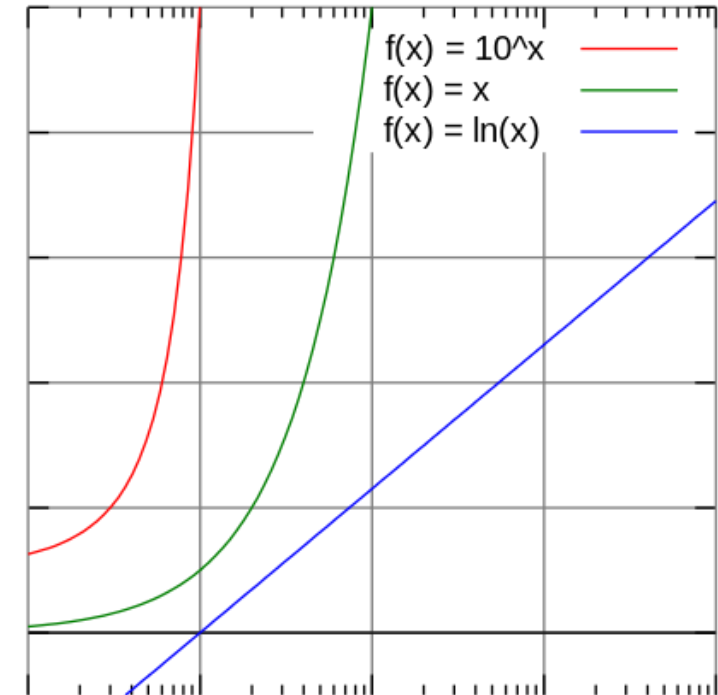
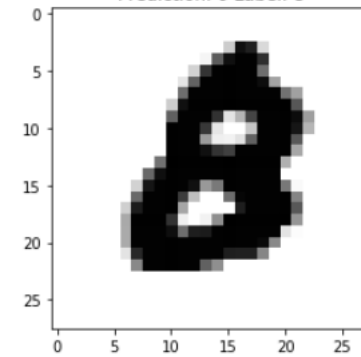


Image: By Autopilot - Own work, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=10733854>

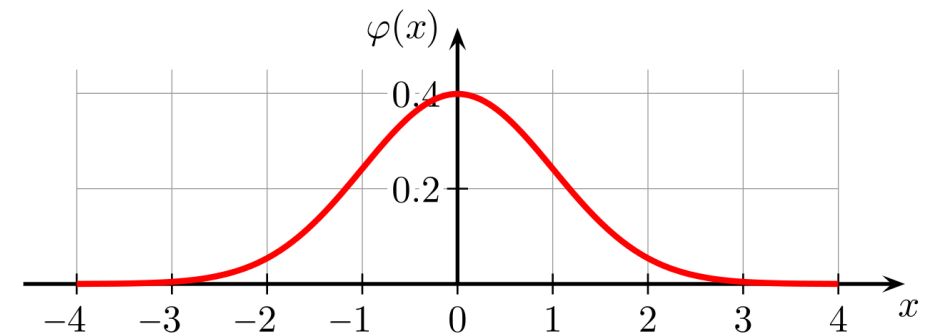
Encoding

- Transforming data into some new representation required by the model
- One-hot encoding
 - Create “buckets” for every category
 - The bucket for your category has a 1, all others have a 0
 - Very common in deep learning, where categories are represented by individual output “neurons”



Scaling / Normalization

- Some models prefer feature data to be normally distributed around 0 (most neural nets)
- Most models require feature data to at least be scaled to comparable values
 - Otherwise features with larger magnitudes will have more weight than they should
 - Example: modeling age and income as features – incomes will be much higher values than ages
- Scikit_learn has a preprocessor module that helps (MinMaxScaler, etc)
- Remember to scale your results back up



Geek3 [CC BY 3.0 (<https://creativecommons.org/licenses/by/3.0/>)]

Shuffling

- Many algorithms benefit from shuffling their training data
- Otherwise they may learn from residual signals in the training data resulting from the order in which they were collected



Advanced Text Pre-processing for AI

- Tokenization splits text into tokens a model can process
- Tokens can be words, subwords, or model-specific units – the tokenizer you use changes the input representation and the token count
- Normalize text first: lowercase, remove or handle unwanted characters, and filter out irrelevant text
- Domain-specific augmentation: add or modify training examples to cover your domain's terminology, phrasing, scenarios, and underrepresented cases
- Goal: make training data representative and useful without changing its intended meaning
- SageMaker Data Wrangler can featurize / tokenize text and apply custom transforms – the exam cares about the concepts, not console steps

Example: same text, different tokens

"Patient shows signs of AFib"

Word-level:

[Patient] [shows] [signs] [of] [AFib]

Subword-level:

[Patient] [shows] [signs] [of] [A] [Fib]

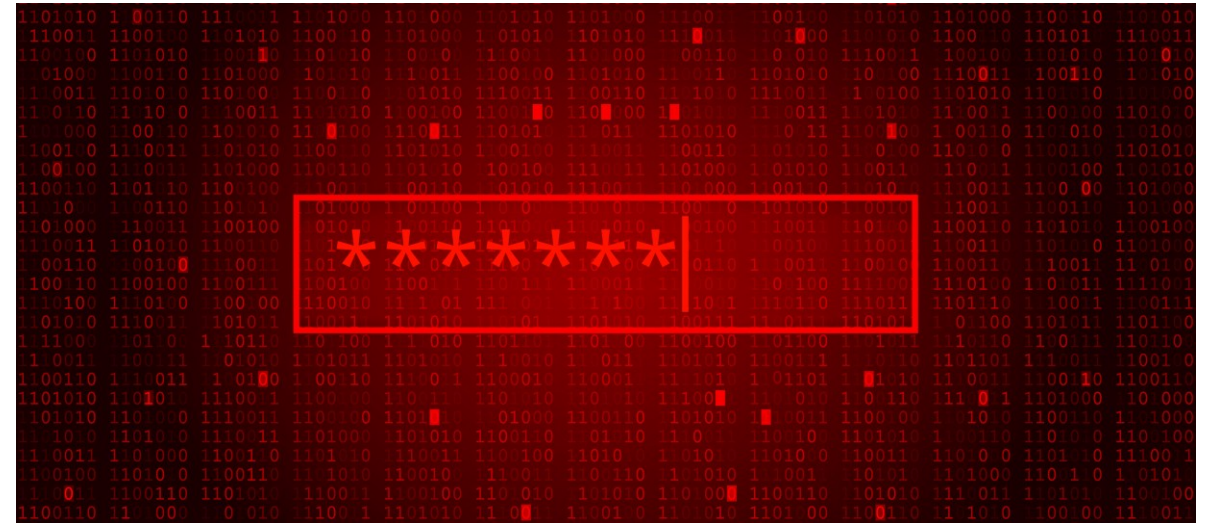
Data Masking and Anonymization

- Dealing with PII or other sensitive data
- Masking obfuscates data
 - For example, masking all but the last 4 digits of a credit card or social security number
 - Masking passwords
 - Supported in Glue DataBrew and Redshift

--create a masking policy that fully masks the credit card number
`CREATE MASKING POLICY mask_credit_card_full WITH (credit_card
VARCHAR(256)) USING ('000000XXXX0000'::TEXT);`

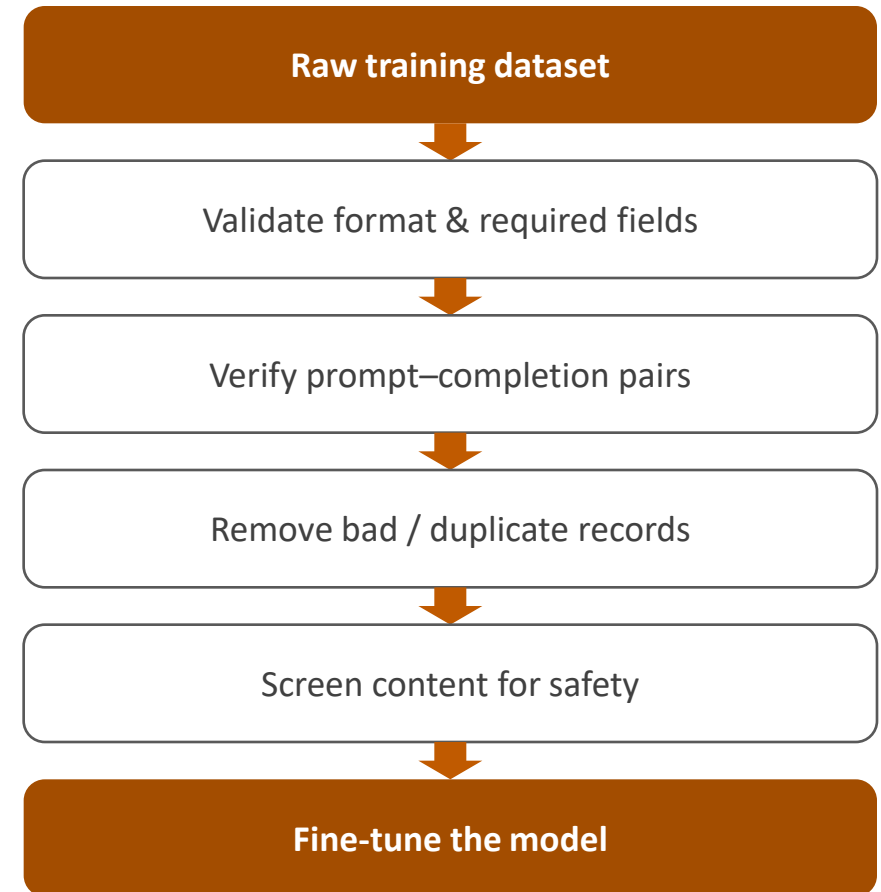
Data Masking and Anonymization

- Anonymization techniques
 - Replace with random
 - Shuffle
 - Encrypt (deterministic or probabilistic)
 - Hashing
- Or just delete it or don't import it in the first place.



AI Training-Data Integrity

- Good training data = structurally valid AND semantically trustworthy
- Validate format & required fields before training
 - e.g., JSONL with prompt + completion fields
- Fine-tuning: each prompt paired with the intended high-quality completion
- Detect & remove bad records
 - Malformed, missing fields, duplicates
 - Contradictory, mislabeled, poor quality
- Keep datasets representative of the production workload



AI Training-Data Integrity

- Screen training content before using it
 - Unsafe, inappropriate, sensitive (e.g., PII), or policy-violating material
- Poor or inconsistent examples directly teach the model undesirable behavior
- These are dataset preparation steps YOU perform – separate from AWS-managed runtime safeguards (e.g., Amazon Bedrock Guardrails)

GOOD PAIR

Prompt: “What is the return window?”

Completion: “Items can be returned within 30 days of delivery.”

BAD PAIR – CONTRADICTORY

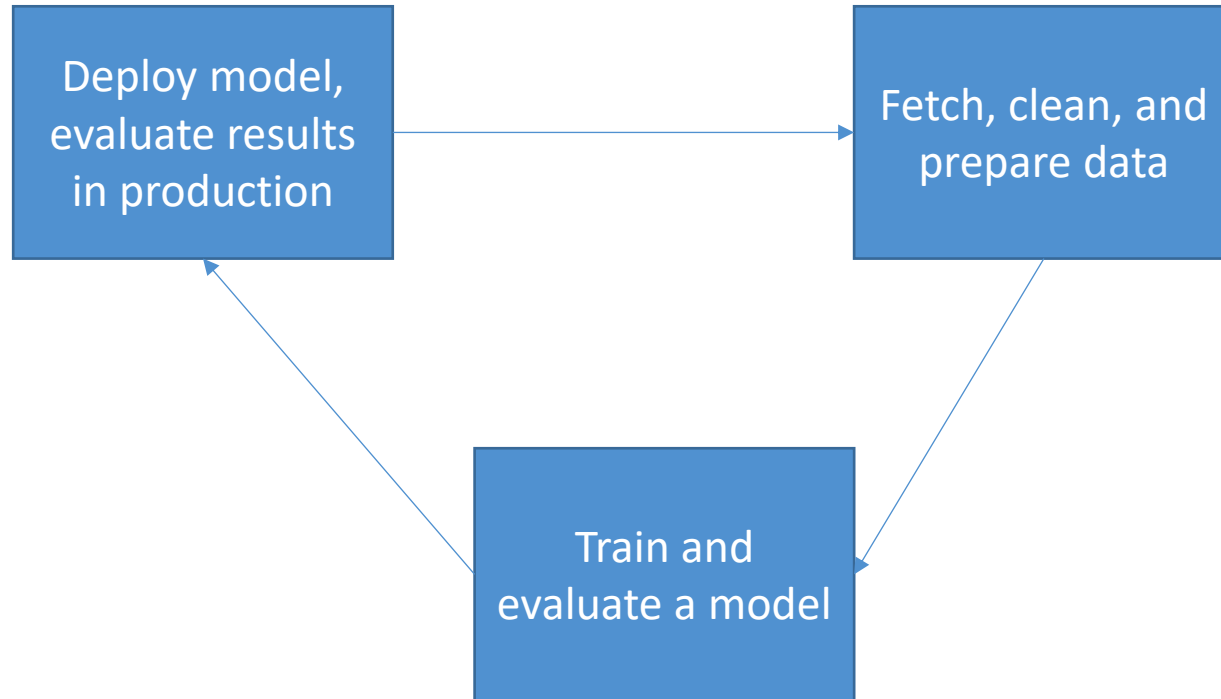
Prompt: “What is the return window?”

Completion: “Returns are not accepted.”

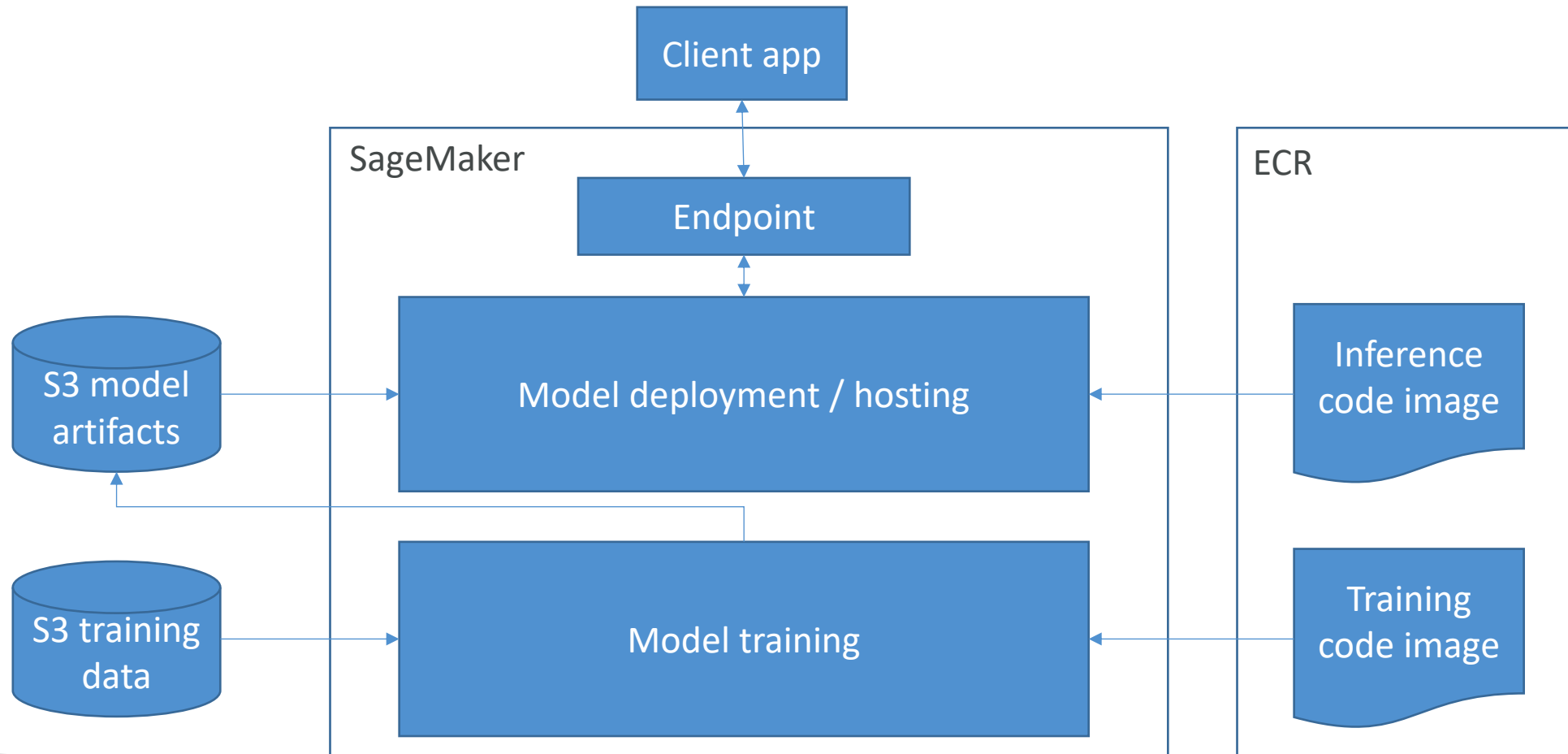
Result: the model learns the wrong answer

Amazon SageMaker AI

SageMaker AI is built to handle the entire machine learning workflow.

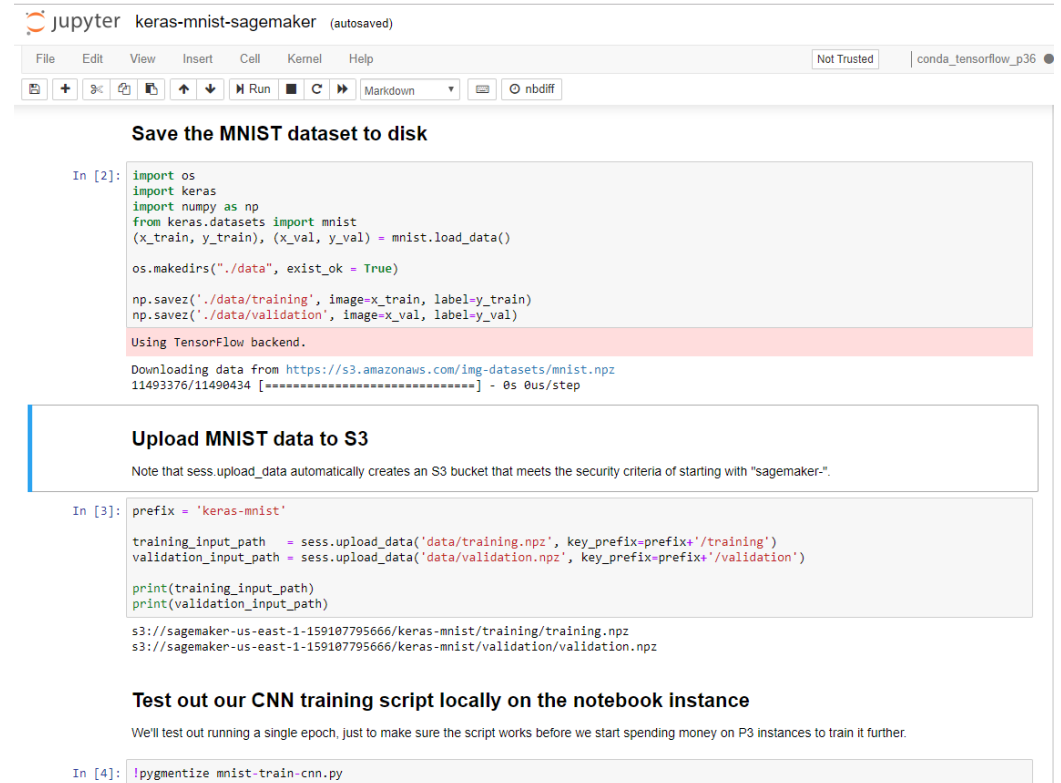


SageMaker AI Training & Deployment



SageMaker Notebooks can direct the process

- Notebook Instances on EC2 are spun up from the console
 - S3 data access
 - Scikit_learn, Spark, Tensorflow
 - Wide variety of built-in models
 - Ability to spin up training instances
 - Ability to deploy trained models for making predictions at scale



The screenshot shows a Jupyter Notebook interface with the title 'keras-mnist-sagemaker (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running cells, and markdown. The notebook content is divided into sections:

- Save the MNIST dataset to disk**: This section contains a code cell (In [2]) that imports necessary libraries, loads the MNIST dataset, and saves it to local files. The code is as follows:

```
import os
import keras
import numpy as np
from keras.datasets import mnist
(x_train, y_train), (x_val, y_val) = mnist.load_data()

os.makedirs("./data", exist_ok = True)

np.savez('./data/training', image=x_train, label=y_train)
np.savez('./data/validation', image=x_val, label=y_val)

Using TensorFlow backend.
```

Below the code, a status message indicates: "Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz 11493376/11490434 [=====] - 0s 0us/step".
- Upload MNIST data to S3**: This section contains a code cell (In [3]) that uploads the local training and validation data to an S3 bucket. The code is as follows:

```
prefix = 'keras-mnist'

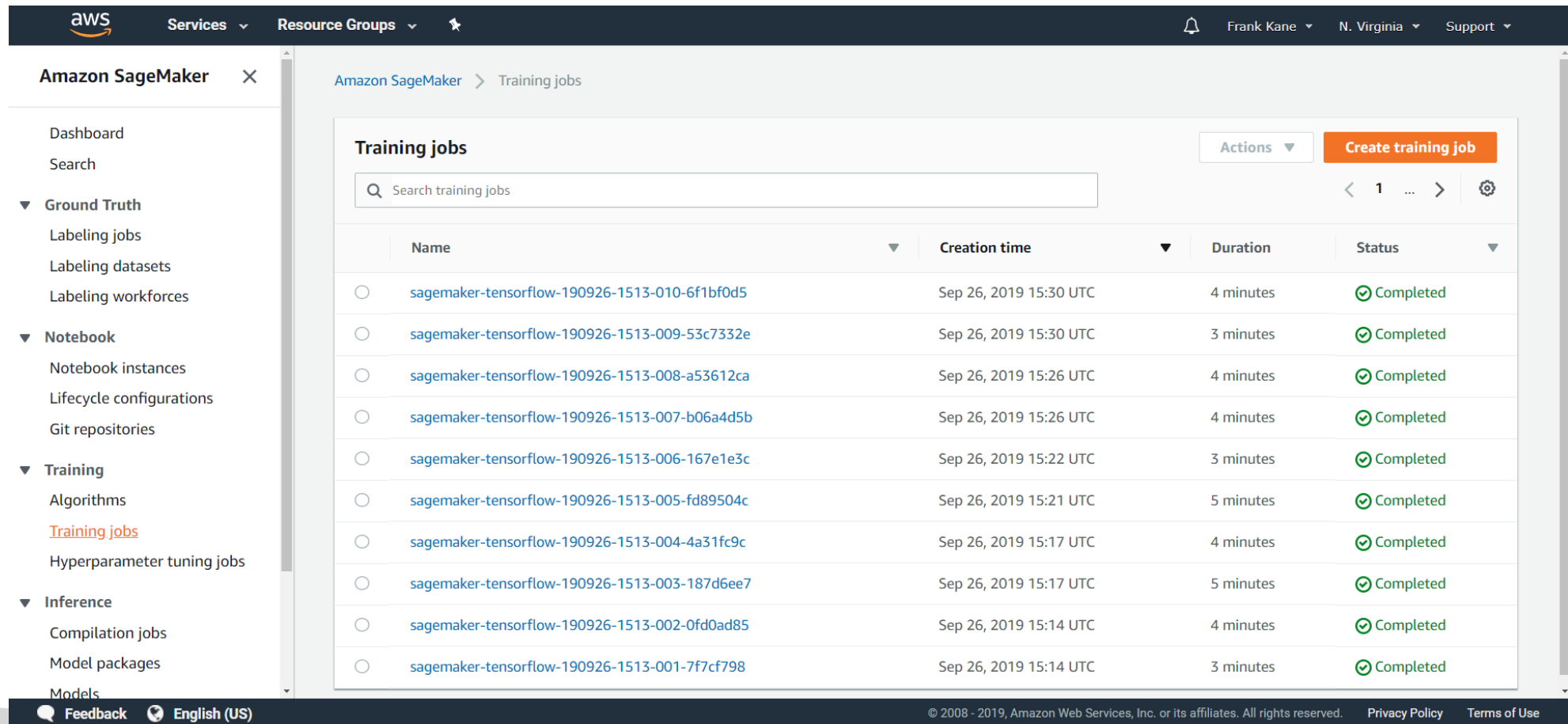
training_input_path = sess.upload_data('data/training.npz', key_prefix=prefix+'/training')
validation_input_path = sess.upload_data('data/validation.npz', key_prefix=prefix+'/validation')

print(training_input_path)
print(validation_input_path)

s3://sagemaker-us-east-1-159107795666/keras-mnist/training/training.npz
s3://sagemaker-us-east-1-159107795666/keras-mnist/validation/validation.npz
```

Below the code, a note states: "Note that sess.upload_data automatically creates an S3 bucket that meets the security criteria of starting with 'sagemaker-'."
- Test out our CNN training script locally on the notebook instance**: This section contains a code cell (In [4]) that runs the command: `!pygmentize mnist-train-cnn.py`. Below the code, a note states: "We'll test out running a single epoch, just to make sure the script works before we start spending money on P3 instances to train it further."

So can the SageMaker AI console

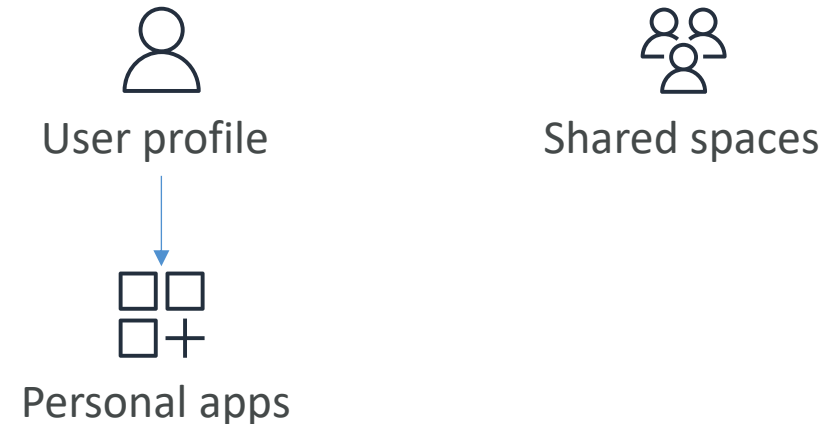
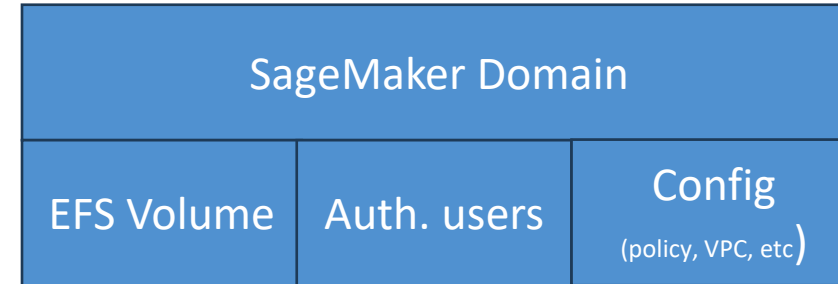


The screenshot displays the Amazon SageMaker console interface. The left sidebar contains navigation links for Dashboard, Search, Ground Truth, Notebook, Training, and Inference. The main content area shows the 'Training jobs' page with a search bar and a table of training jobs. The table has columns for Name, Creation time, Duration, and Status. All jobs listed are in a 'Completed' status.

Name	Creation time	Duration	Status
sagemaker-tensorflow-190926-1513-010-6f1bf0d5	Sep 26, 2019 15:30 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-009-53c7332e	Sep 26, 2019 15:30 UTC	3 minutes	Completed
sagemaker-tensorflow-190926-1513-008-a53612ca	Sep 26, 2019 15:26 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-007-b06a4d5b	Sep 26, 2019 15:26 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-006-167e1e3c	Sep 26, 2019 15:22 UTC	3 minutes	Completed
sagemaker-tensorflow-190926-1513-005-fd89504c	Sep 26, 2019 15:21 UTC	5 minutes	Completed
sagemaker-tensorflow-190926-1513-004-4a31fc9c	Sep 26, 2019 15:17 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-003-187d6ee7	Sep 26, 2019 15:17 UTC	5 minutes	Completed
sagemaker-tensorflow-190926-1513-002-0fd0ad85	Sep 26, 2019 15:14 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-001-7f7cf798	Sep 26, 2019 15:14 UTC	3 minutes	Completed

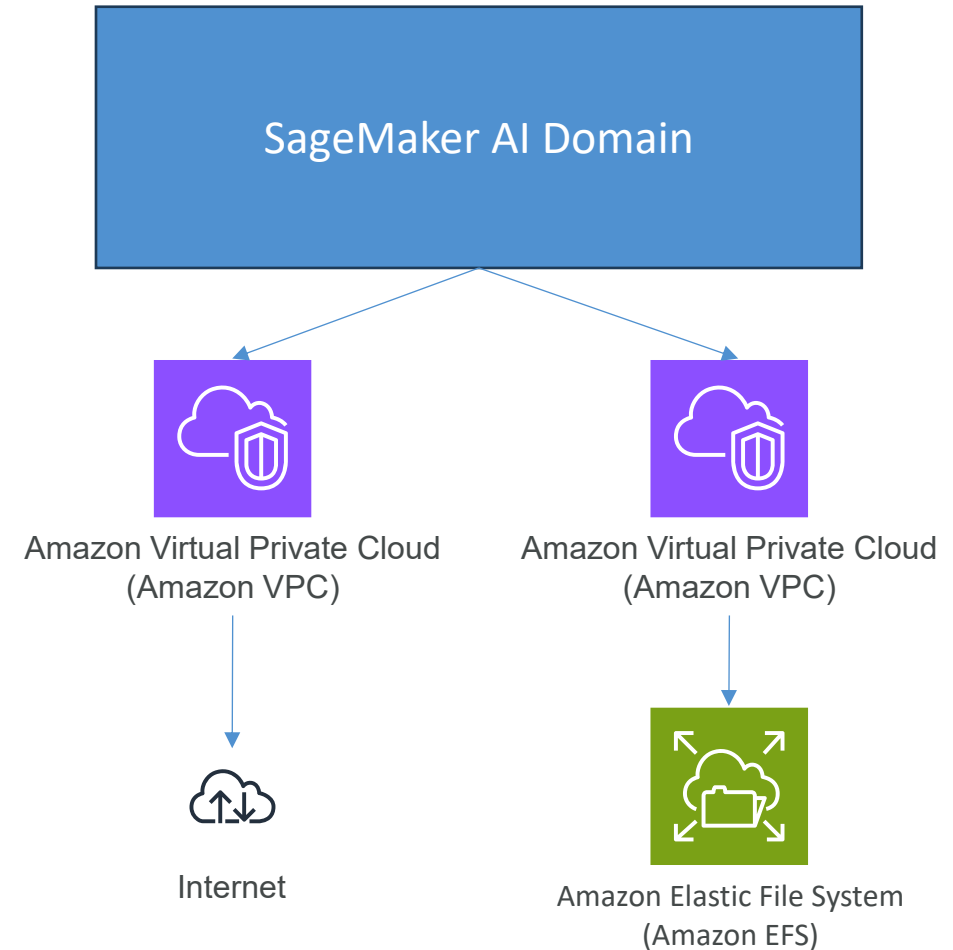
SageMaker AI Domains

- Domains organize users, apps, and resources
 - Share an EFS volume
- User profiles own personal apps
 - SageMaker Studio instances
 - Private EFS directory
 - Shared resources across other users
- Shared spaces
 - Shared EFS directory
 - Communal IDE app



VPC's for SageMaker AI Domains

- By default, a domain has two VPC's
 - One for Internet access
 - Managed by SageMaker AI
 - And your own
 - Encrypted traffic to your EFS volume
 - You specify the VPC, its subnets, and security groups
- You can change this
 - Send all traffic to your own VPC
 - Select “VPC Only” when creating the domain



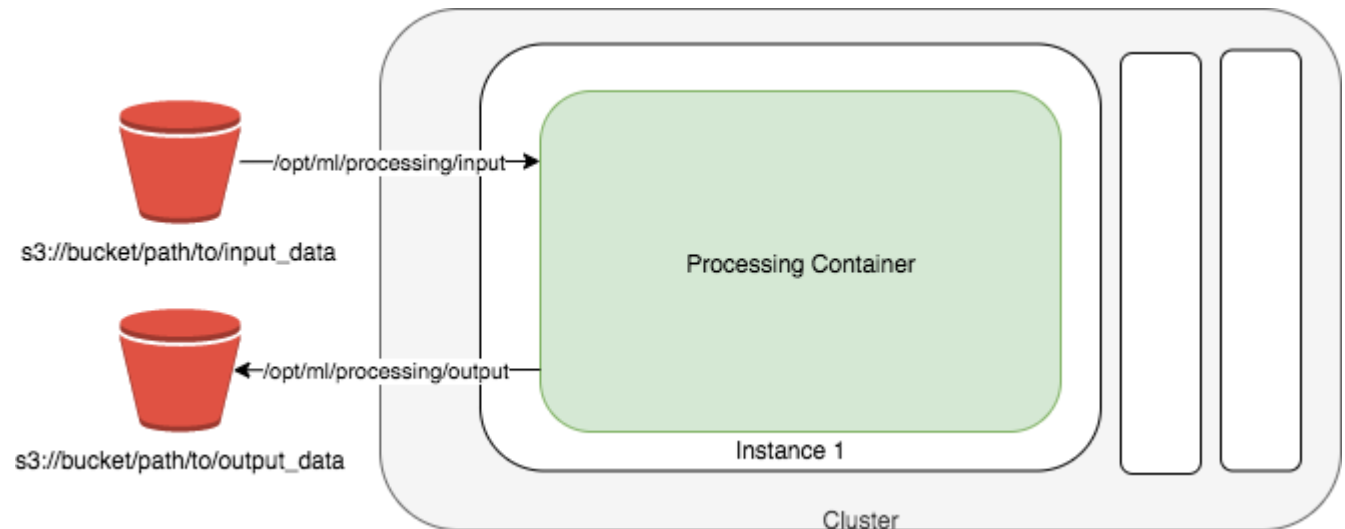
Data prep on SageMaker AI

- Data usually comes from S3
 - Ideal format varies with algorithm – often it is RecordIO / Protobuf
- Can also ingest from Athena, EMR, Redshift, and Amazon Keyspaces DB
- Apache Spark integrates with SageMaker
 - More on this later...
- Scikit_learn, numpy, pandas all at your disposal within a notebook



SageMaker Processing

- Processing jobs
 - Copy data from S3
 - Spin up a processing container
 - SageMaker built-in or user provided
 - Output processed data to S3



Training on SageMaker AI

- Create a training job
 - URL of S3 bucket with training data
 - ML compute resources
 - URL of S3 bucket for output
 - ECR path to training code
- Training options
 - Built-in training algorithms
 - Spark MLlib
 - Custom Python Tensorflow / MXNet code
 - PyTorch, Scikit-Learn, RLEstimator
 - XGBoost, Hugging Face, Chainer
 - Your own Docker image
 - Algorithm purchased from AWS marketplace



Deploying Trained Models

- Save your trained model to S3
- Can deploy two ways:
 - Persistent endpoint for making individual predictions on demand
 - SageMaker Batch Transform to get predictions for an entire dataset
- Lots of cool options
 - Inference Pipelines for more complex processing
 - SageMaker Neo for deploying to edge devices
 - Elastic Inference for accelerating deep learning models
 - Automatic scaling (increase # of endpoints as needed)
 - Shadow Testing evaluates new models against currently deployed model to catch errors



Labeling and Annotating Training Data

Labeling and Annotating Training Data

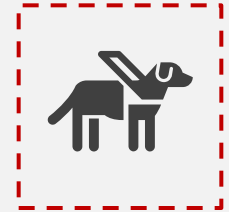
- Supervised learning needs labeled examples: inputs paired with a known target — a category, value, or annotation
- A label answers one question about the whole item: “this photo contains a dog”
- An annotation is richer — it marks regions, objects, entities, or relationships within the data
- Same data, different depth — it depends on what the model must learn

Classification



Label: “dog”

Object detection



Bounding box + class

Semantic segmentation



Every pixel classified

Named entity recognition

Amazon (ORG) opened an
office in Seattle (LOC)

Common annotation tasks

Image classification



One label for the whole image

Text classification



Topic, sentiment, spam / not spam

Multi-label classification



Several labels can apply at once

Object detection



Bounding box + class for each object

Semantic segmentation



Every pixel assigned a class

Named entity recognition



Tag people, places, orgs in text

Video & multimodal



Track objects across frames;
audio & 3D too

Where do labels come from?



Existing business data

- Outcomes you already track become labels
- Purchases, churn, clicks, fraud flags, returns
- Cheapest source — check it first



Human annotation

- People apply labels by hand
- In-house experts, vendors, or crowd workforces
- Best for judgment calls — but slow and costly

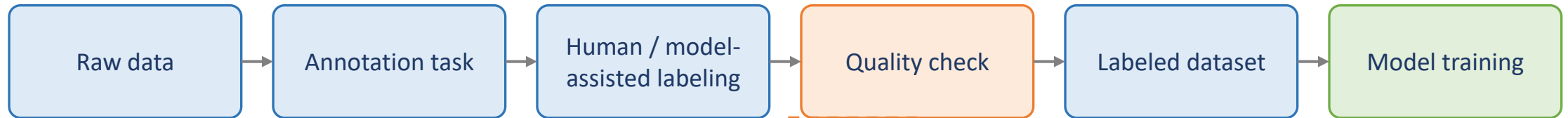


Model-assisted labeling

- A model does the first pass: Rekognition for images, Comprehend for text, or any pre-trained / unsupervised technique
- Humans review the uncertain ones (active learning)

Heads-up: SageMaker Ground Truth historically managed human labeling workflows on AWS, and Mechanical Turk provided the crowd workforce. Both are now in maintenance mode / deprecated — know they existed, but don't treat them as the go-forward answer.

Label quality matters



What hurts label quality

- Ambiguous instructions — annotators guess
- Inconsistent annotators — conflicting answers
- Incorrect labels — the model learns the mistake
- Biased labeling — a biased model
- Downstream: data quality, class imbalance, bias, and training-data integrity problems

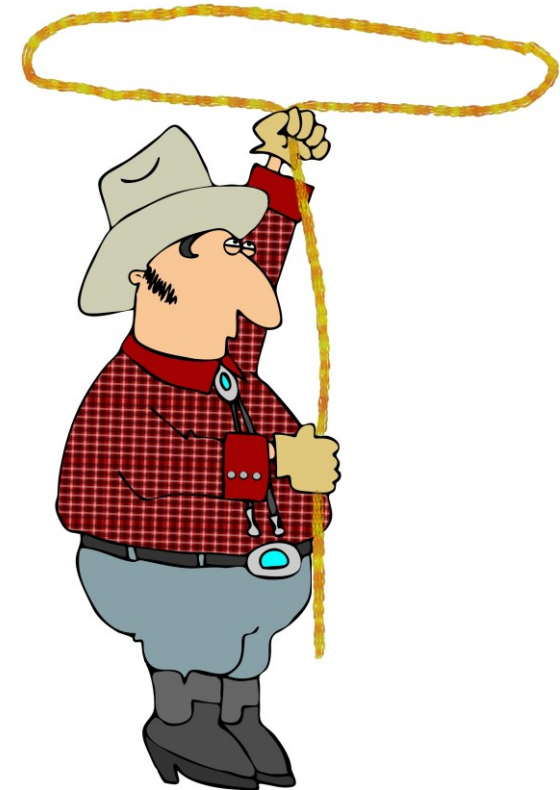
Quality control

- Clear annotation guidelines, with examples
- Multiple annotators per item — compare for agreement
- Consolidate / adjudicate disagreements
- Spot-check uncertain or low-confidence labels
- More reviews = better labels, but more cost and time

Exam tip: MLA-C02 tests the skill “label and annotate data” — don’t assume a question is asking for a specific legacy labeling service.

SageMaker Data Wrangler

- Visual interface (in SageMaker Studio) to prepare data for machine learning
- Import data
- Visualize data
- Transform data (300+ transformations to choose from)
 - Or integrate your own custom xforms with pandas, PySpark, PySpark SQL
- “Quick Model” to train your model with your data and measure its results

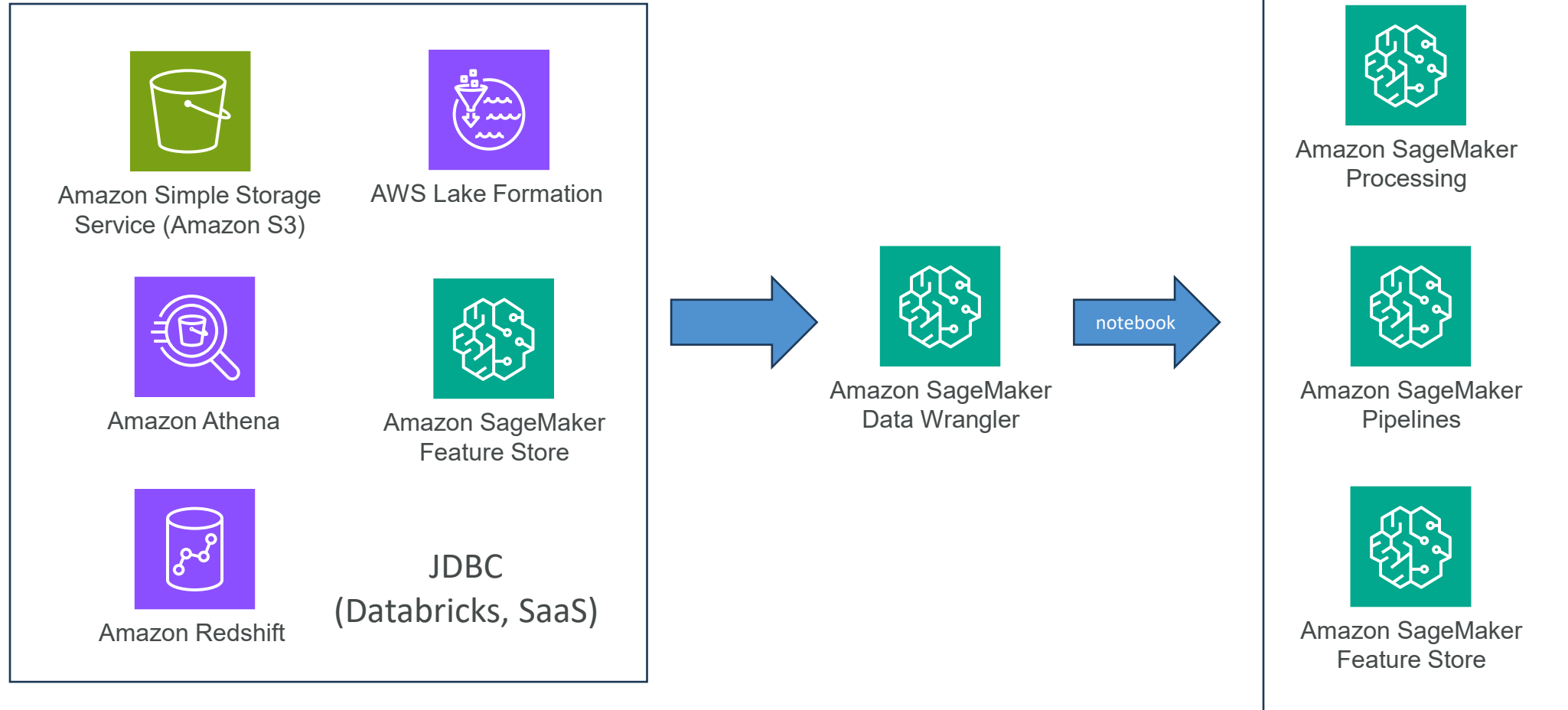


Data Wrangler: Notable Capabilities

- Transforming image data
 - Resize, enhance, corrupt images & more
- Balance data
 - Random oversampling, undersampling, SMOTE
- Impute missing data
- Handle outliers
- Dimensionality reduction (PCA)



Data Wrangler sources



Data Wrangler: Import Data

The screenshot shows the Amazon SageMaker Studio interface with the Data Wrangler tool open. The 'Import' tab is active, displaying a table of S3 objects for the dataset 'titanic.train.csv'. The table has columns for Object Name, Size, and Last Modified. Below the table, there is a 'Preview' section showing the first five rows of the dataset. On the right side, the 'DETAILS' panel shows the dataset name, URI, file type (csv), and options to add headers and enable sampling. An 'Import dataset' button is visible at the bottom of the details panel.

Object Name | **Size** | **Last Modified**

Object Name	Size	Last Modified
titanic-train.csv	58.89KB	2020-10-15 08:49:45+00:00

Preview

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
1	0	3	Braund, Mr. Owen Harris	male	22	1
2	1	1	Cumings, Mrs. John Bra...	female	38	1
3	1	3	Heikkinen, Miss. Laina	female	26	0
4	1	1	Futrelle, Mrs. Jacques H...	female	35	1
5	0	3	Allen, Mr. William Henry	male	35	0

DETAILS

Name: titanic-train.csv

Required

URI: s3://sagemaker-us-east-2-

File type: csv

Required

☒ Add header to table

☒ Enable sampling

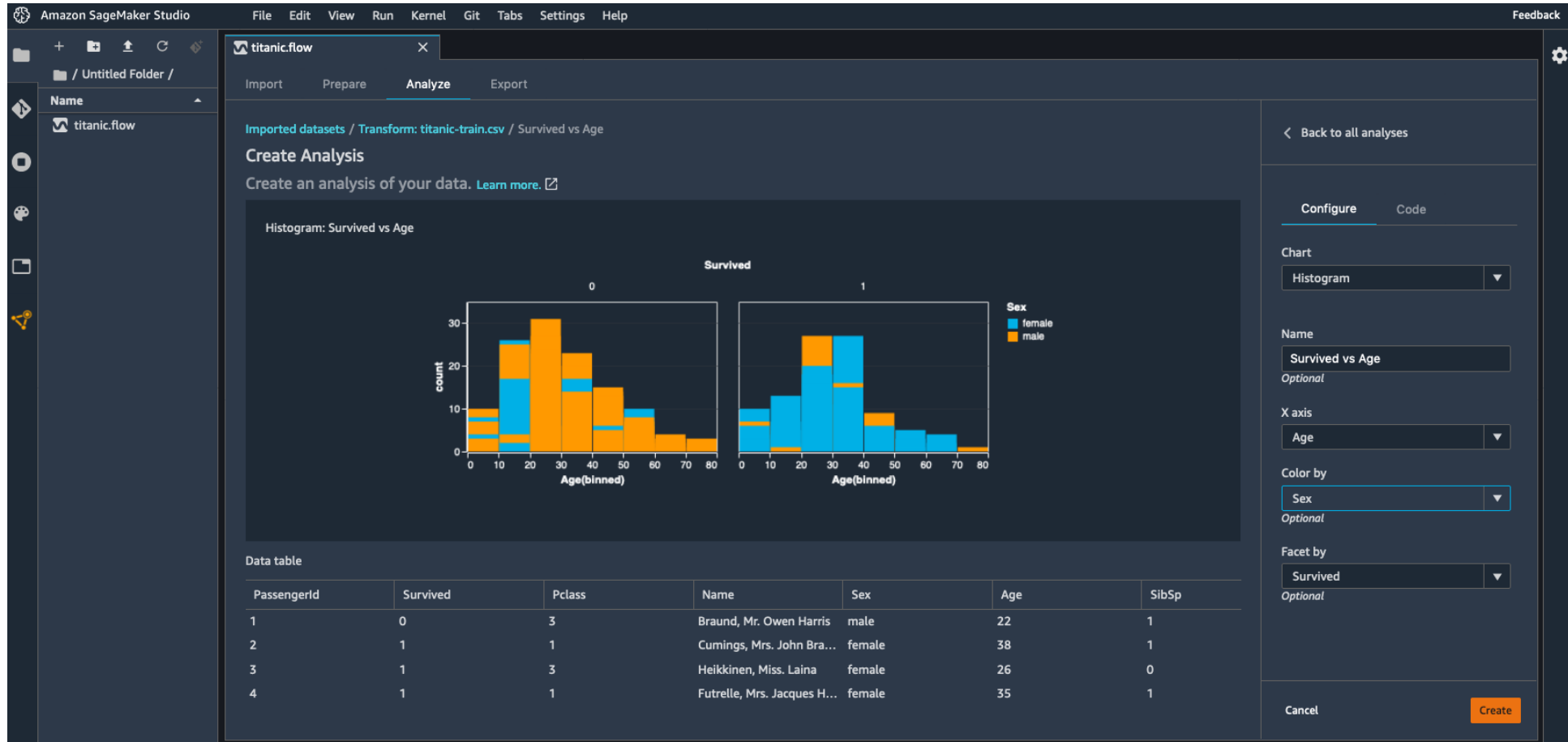
Import dataset

Data Wrangler: Preview Data

The screenshot displays the Amazon SageMaker Studio interface, specifically the Data Wrangler tool. The main window shows a data preview for a file named 'titanic.train.csv'. The data is presented in a table with columns: Passengerid (long), Survived (long), Pclass (long), Name (string), Sex (string), and Age (long). The preview shows 28 rows of data. To the right of the preview, there is a 'CONFIGURE TYPES' panel where the data types for each column are set. The types are: Passengerid (Long), Survived (Long), Pclass (Long), Name (String), Sex (String), Age (Long), SibSp (Long), Parch (Long), Ticket (String), Fare (Float), Cabin (String), and Embarked (String). At the bottom of the configuration panel, there are buttons for 'Clear', 'Preview', and 'Apply'.

Passengerid (long)	Survived (long)	Pclass (long)	Name (string)	Sex (string)	Age (long)
7	0	1	McCarthy, Mr. Timothy J	male	54
8	0	3	Palsson, Master. Gosta ...	male	2
9	1	3	Johnson, Mrs. Oscar W (...)	female	27
10	1	2	Nasser, Mrs. Nicholas (A...	female	14
11	1	3	Sandstrom, Miss. Margu...	female	4
12	1	1	Bonnell, Miss. Elizabeth	female	58
13	0	3	Saunderscock, Mr. Willia...	male	20
14	0	3	Andersson, Mr. Anders J...	male	39
15	0	3	Vestrom, Miss. Hulda A...	female	14
16	1	2	Hewlett, Mrs. (Mary D K...	female	55
17	0	3	Rice, Master. Eugene	male	2
18	1	2	Williams, Mr. Charles Eu...	male	
19	0	3	Vander Planke, Mrs. Juli...	female	31
20	1	3	Maselmani, Mrs. Fatima	female	
21	0	2	Fynney, Mr. Joseph J	male	35
22	1	2	Beesley, Mr. Lawrence	male	34
23	1	3	McGowan, Miss. Anna "...	female	15
24	1	1	Sloper, Mr. William Tho...	male	28
25	0	3	Palsson, Miss. Torborg ...	female	8
26	1	3	Asplund, Mrs. Carl Osca...	female	38
27	0	3	Emir, Mr. Farred Chehab	male	
28	0	1	Fortune, Mr. Charles Ale...	male	19

Data Wrangler: Visualize Data



Data Wrangler: Transform Data

The screenshot shows the Amazon SageMaker Studio interface with the Data Wrangler tool open. The 'Prepare' tab is active, and the 'Encode categorical' transform is being configured for the 'Pclass' column. The transform is set to 'One-hot encode' with 'Keep' as the invalid handling strategy. The output format is 'Flattened'.

Transform: titanic-train.csv

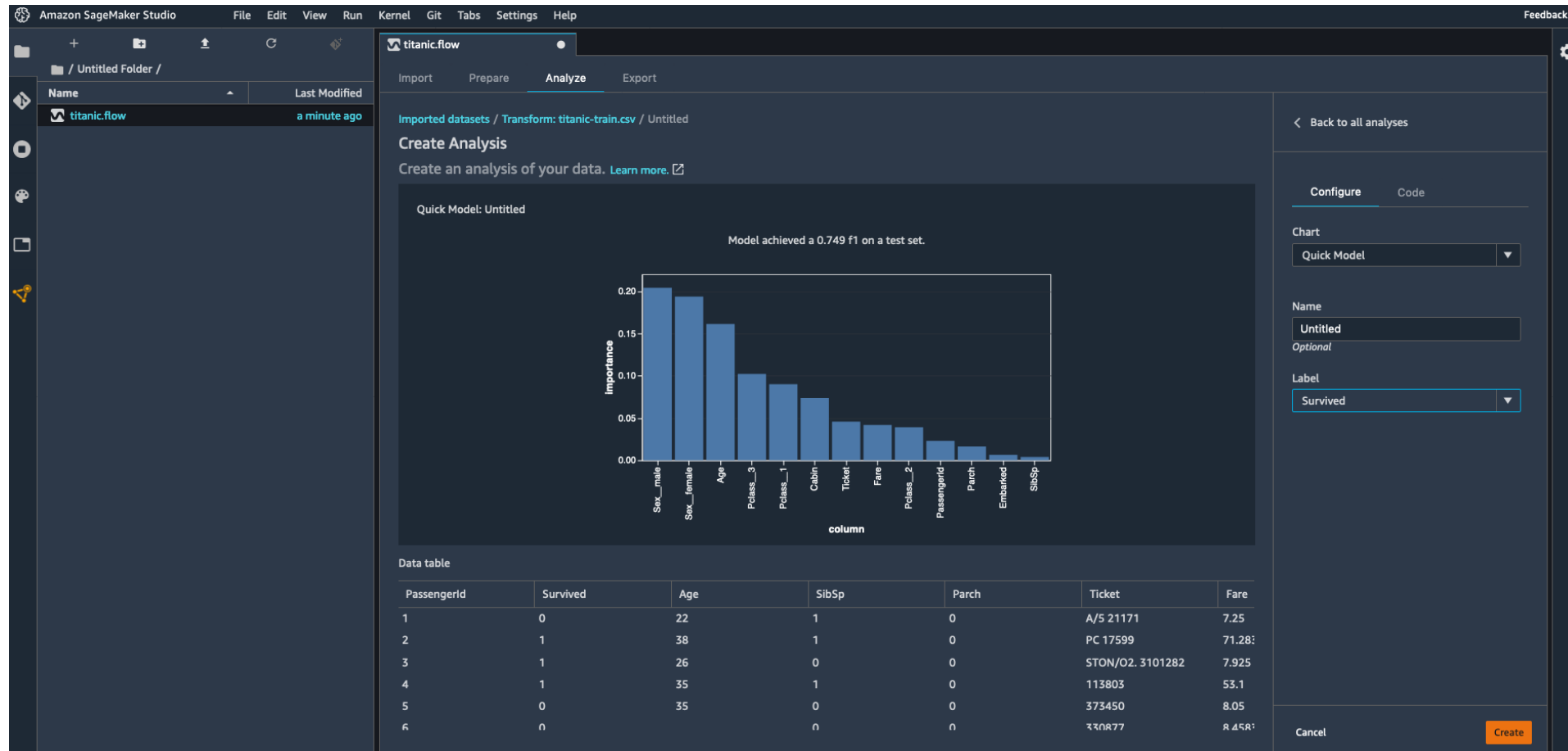
t)	Cabin (string)	Embarked (string)	Pclass_3 (float)	Pclass_1 (float)	Pclass_2 (float)
		S	1	0	0
	C85	C	0	1	0
		S	1	0	0
	C123	S	0	1	0
		S	1	0	0
		Q	1	0	0
	E46	S	0	1	0
		S	1	0	0
		C	0	0	1
	G6	S	1	0	0
	C103	S	0	1	0
		S	1	0	0
		S	1	0	0
		S	1	0	0
		S	0	0	1
		Q	1	0	0
		S	0	0	1
		S	1	0	0

Configuration:

- Input column: Pclass
- Output column name: Pclass_
- Transform: One-hot encode
- Invalid handling strategy: Keep
- Drop last category: Select...
- Is input ordinal encoded?: Select...
- Output format: Flattened

Buttons: Clear, Preview, Add

Data Wrangler: Quick Model



Data Wrangler: Export Data Flow

The screenshot displays the Amazon SageMaker Studio interface, specifically the Data Wrangler tool. The top menu bar includes 'Amazon SageMaker Studio', 'File', 'Edit', 'View', 'Run', 'Kernel', 'Git', 'Tabs', 'Settings', and 'Help'. The left sidebar shows a file explorer with a folder named 'Untitled Folder' containing a file 'titanic.flow' modified 'a minute ago'. The main workspace is titled 'titanic.flow' and has tabs for 'Import', 'Prepare', 'Analyze', and 'Export'. The 'Export' tab is active, showing the 'Export data flow' section. Below this, a text prompt says 'Select the steps in the data flow that you want to export and then select an export option.' The data flow diagram consists of three steps: 'Source - sampled' (S3: titanic-train.csv), 'Data types' (Transform: titanic-train.csv), and 'Steps (5)' (highlighted with a blue box). On the right, a panel titled 'Export step' lists four options: 'Processing Job' (Export a Jupyter Notebook that launches a processing job.), 'Pipeline' (Export a Jupyter Notebook that creates a Pipeline with your data flow.), 'Python Code' (Export your data flow to python code.), and 'Feature Store' (Export a Jupyter Notebook that creates a Feature Store feature group and adds features to an offline or online feature store.).

Data Wrangler Troubleshooting

- Make sure your Studio user has appropriate IAM roles
- Make sure permissions on your data sources allow Data Wrangler access
 - Add AmazonSageMakerFullAccess policy
- EC2 instance limit
 - If you get “The following instance type is not available...” errors
 - May need to request a quota increase
 - Service Quotas / Amazon SageMaker / Studio KernelGateway Apps running on ml.m5.4xlarge instance

Pre-training Bias Metrics

- Class Imbalance (CI)
 - One facet (demographic group) has fewer training values than another
- Difference in Proportions of Labels (DPL)
 - Imbalance of positive outcomes between facet values
- Kullback-Leibler Divergence (KL), Jensen-Shannon Divergence(JS)
 - How much outcome distributions of facets diverge
- Lp-norm (LP)
 - P-norm difference between distributions of outcomes from facets
- Total Variation Distance (TVD)
 - L1-norm difference between distributions of outcomes from facets
- Kolmogorov-Smirnov (KS)
 - Maximum divergence between outcomes in distributions from facets
- Conditional Demographic Disparity (CDD)
 - Disparity of outcomes between facets as a whole, and by subgroups



Mitigating Bias

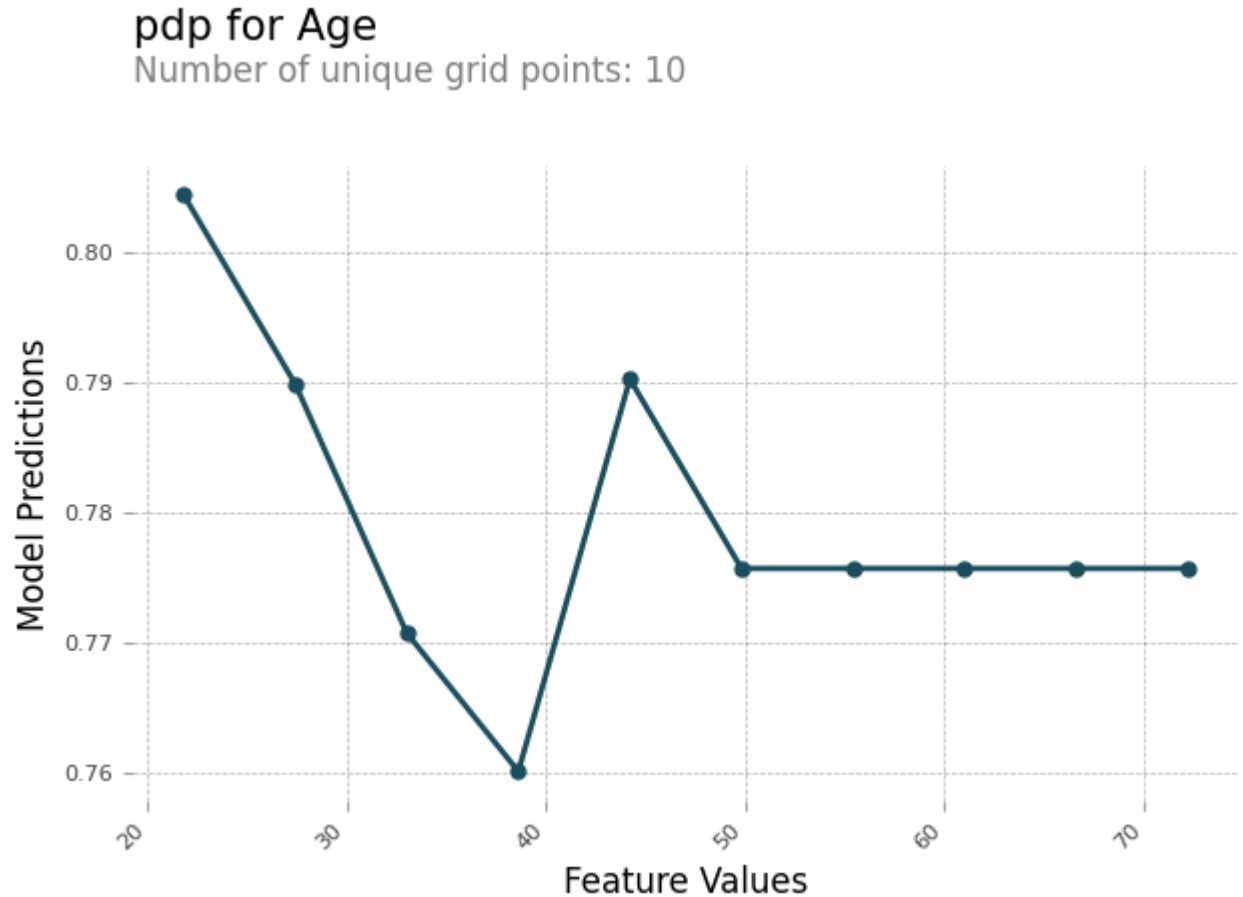
- If your training data has hidden biases, so will your LLM
- One solution: disambiguation (clarify the prompt)
 - Make the user specify what race / gender / orientation / etc. they want an image of
 - Text-to-image disambiguation framework (TIED)
 - Text-to-image ambiguity benchmark (TAB)
 - Clarify with few-shot learning
 - Use a system prompt to enforce diversity in the results
- Fix/enhance the training data
 - Analyze your training data for biases, rebalance it or synthesize balanced data
 - Analyze your output (image recognition to detect imbalances in gender / race / orientation etc.
- Counterfactual data augmentation (change images after the fact)
 - Detect / Segment / Augment



“Generate an image of 2 pizzas, surrounded by 10 software engineers” – DALL-E

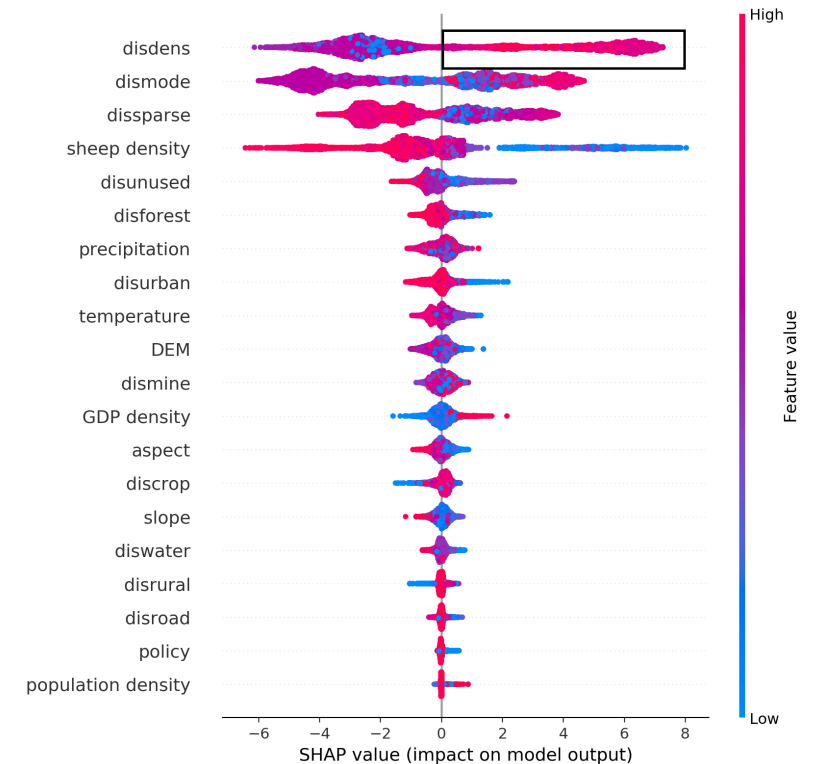
Partial Dependence Plots (PDPs)

- Shows dependence of predicted target response on a set of input features
- Plots can show you how feature values influence the predictions
 - In this example, higher values result in the same predictions
- You can also get back data distributions for each bucket of values



Shapley Values

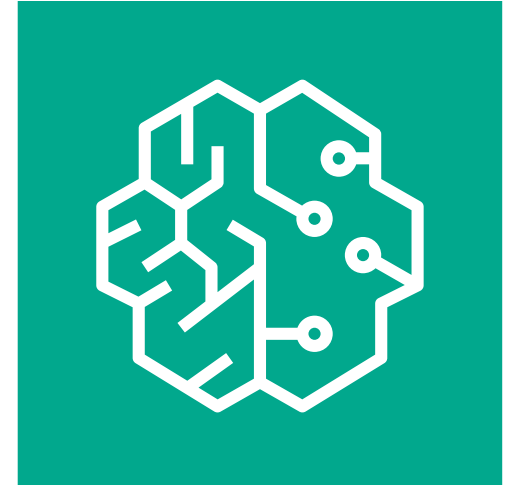
- Shapley values are the algorithm used to determine the contribution of each feature toward a model's predictions
 - Originated in game theory, adapted to ML
 - Basically measures the impact of dropping individual features
 - Gets complicated with lots of features
 - SageMaker Clarify (RIP) used Shapley Additive exPlanations (SHAP) as an approximation technique
- Asymmetric Shapley Values:
 - For Time Series
 - The algorithm used to determine the contribution of input features at each time step toward forecasted predictions



Batunacun, Wieland, R., Lakes, T., and Nendel, C.: Using Shapley additive explanations to interpret extreme gradient boosting predictions of grassland degradation in Xilingol, China, Geosci. Model Dev., 14, 1493–1510, <https://doi.org/10.5194/gmd-14-1493-2021>, 2021.

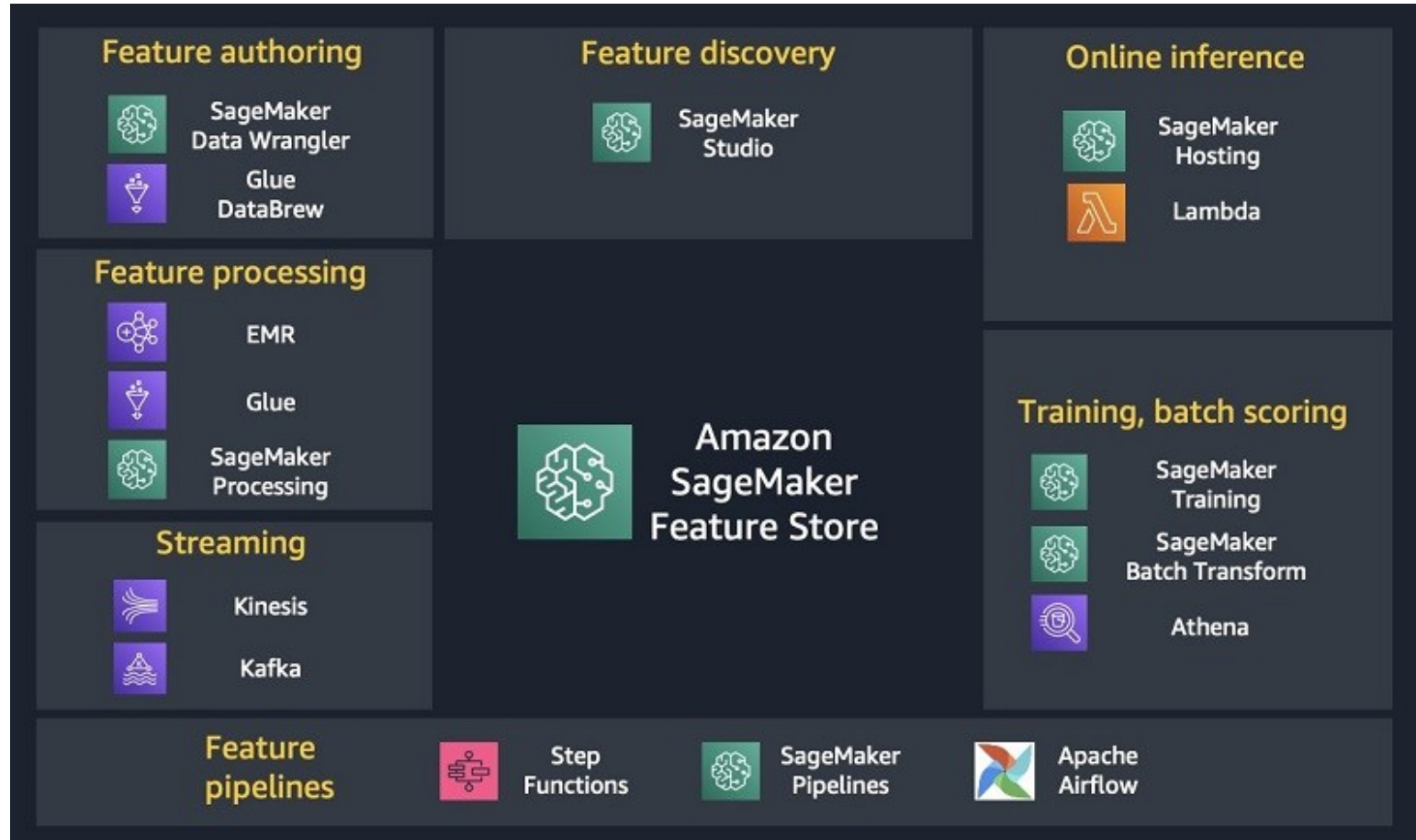
SageMaker Feature Store

- A “feature” is just a property used to train a machine learning model.
 - Like, you might predict someone’s political party based on “features” such as their address, income, age, etc.
- Machine learning models require fast, secure access to feature data for training.
- It’s also a challenge to keep it organized and share features across different models.



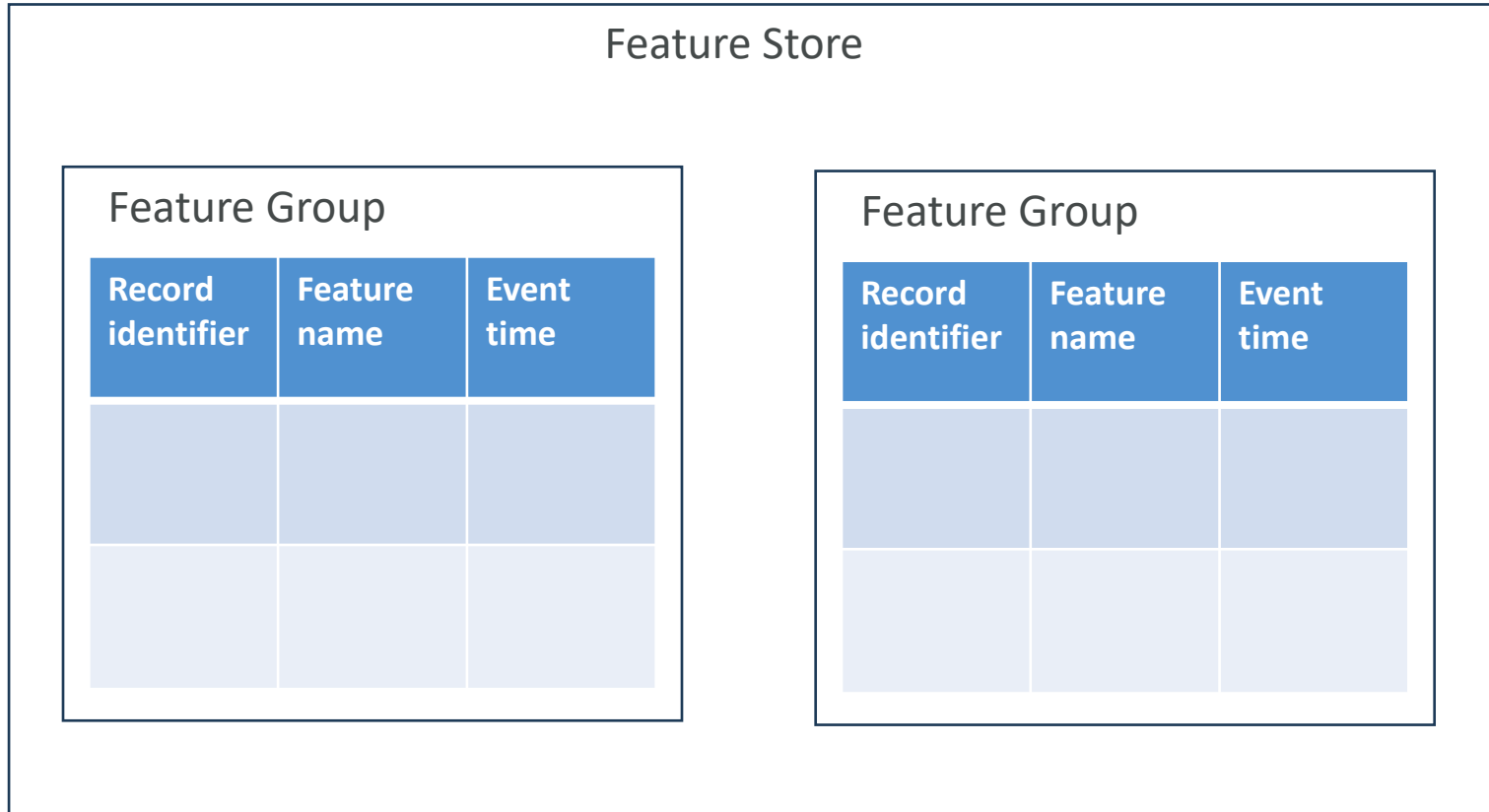
Amazon SageMaker
Feature Store

Where the features come from is up to you.

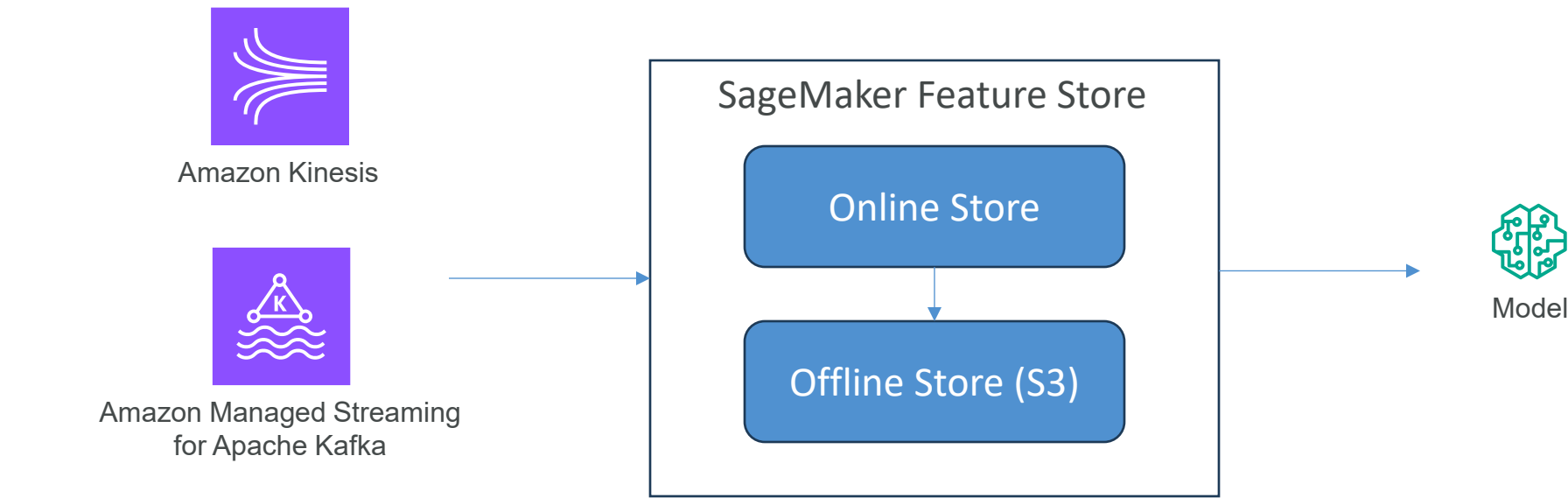


AWS

How SageMaker Feature Store Organizes Your Data



Data Ingestion (streaming or batch)



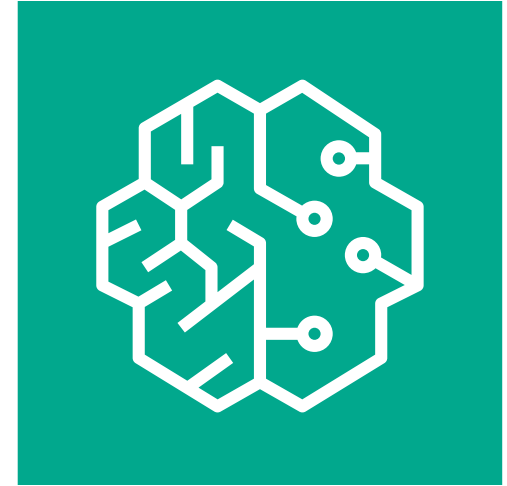
Spark, Data Wrangler, others

STREAMING access via PutRecord / GetRecord API's

BATCH access via the offline S3 store (use with anything that hits S3, like Athena, Data Wrangler. Automatically creates a Glue Data Catalog for you.)

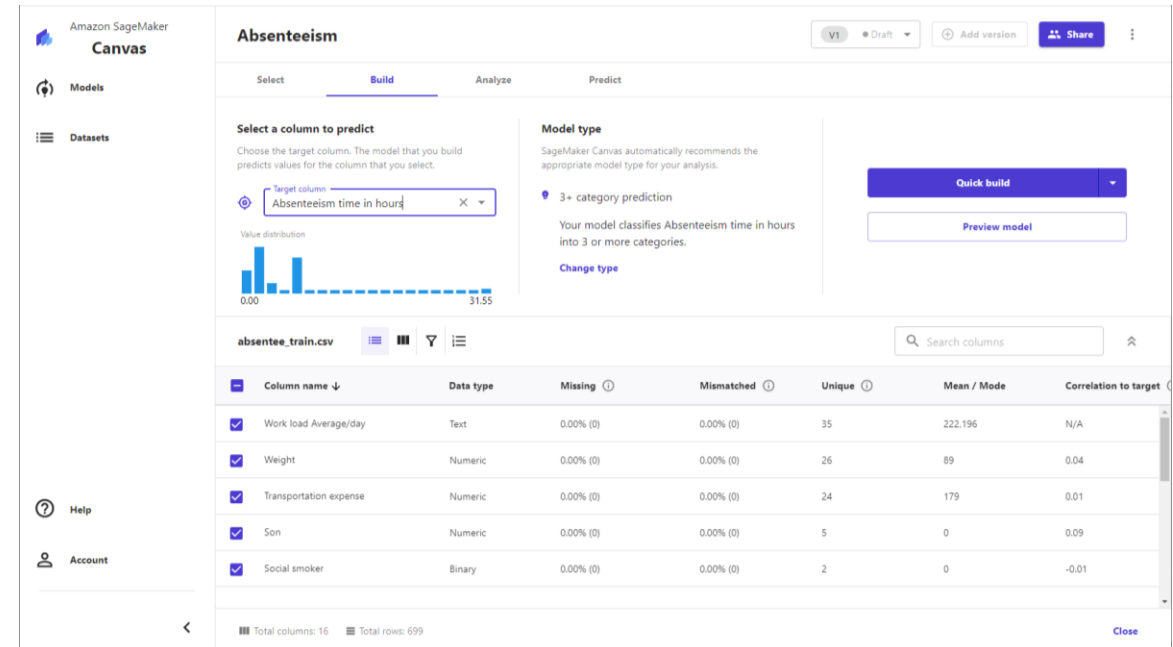
SageMaker Feature Store Security

- Encrypted at rest and in transit
- Works with KMS customer master keys
- Fine-grained access control with IAM
- May also be secured with AWS PrivateLink



SageMaker Canvas

- No-code machine learning for business analysts
- Upload csv data (csv only for now), select a column to predict, build it, and make predictions
- Can also join datasets
- Classification or regression
- Automatic data cleaning
 - Missing values
 - Outliers
 - Duplicates
- Share models & datasets with SageMaker Studio
- Also has generative AI support via Bedrock or JumpStart foundation models
 - Many are fine-tunable within Canvas



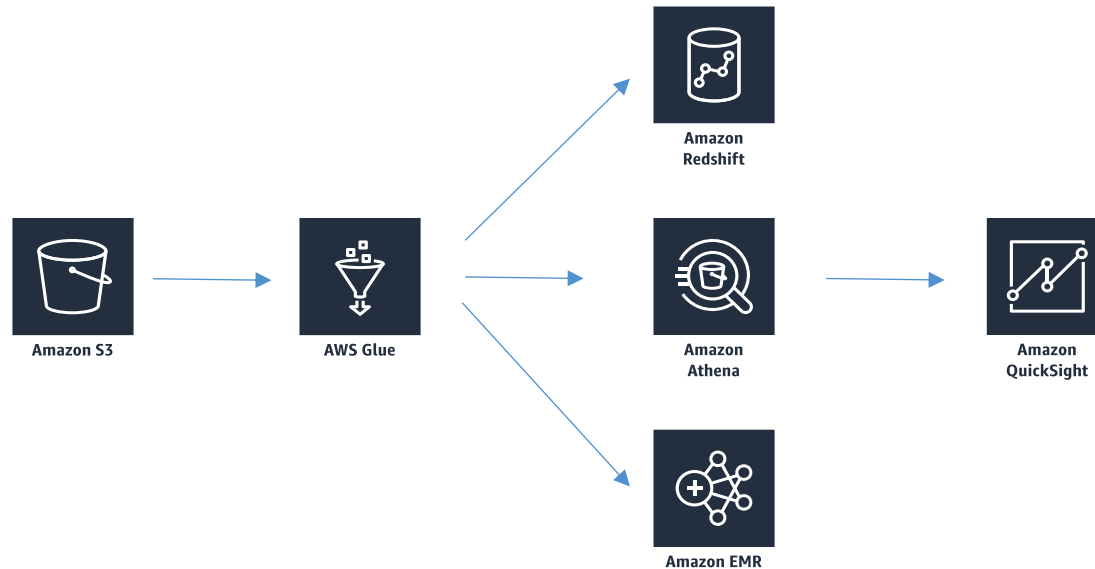
AWS Glue

Table definitions and ETL

What is Glue?

- Serverless discovery and definition of table definitions and schema
 - S3 “data lakes”
 - RDS
 - Redshift
 - DynamoDB
 - Most other SQL databases
- Custom ETL jobs
 - Trigger-driven, on a schedule, or on demand
 - Fully managed

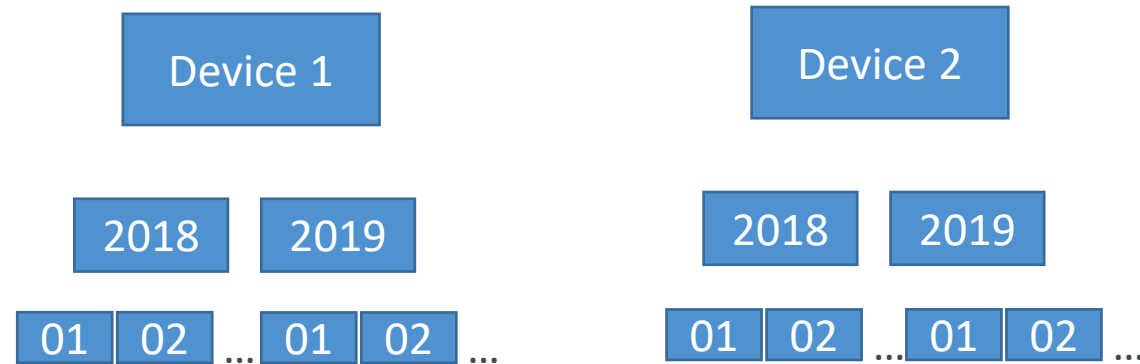
Glue Crawler / Data Catalog



- Glue crawler scans data in S3, creates schema
- Can run periodically
- Populates the Glue Data Catalog
 - Stores only table definition
 - Original data stays in S3
- Once cataloged, you can treat your unstructured data like it's structured
 - Redshift Spectrum
 - Athena
 - EMR
 - Quicksight

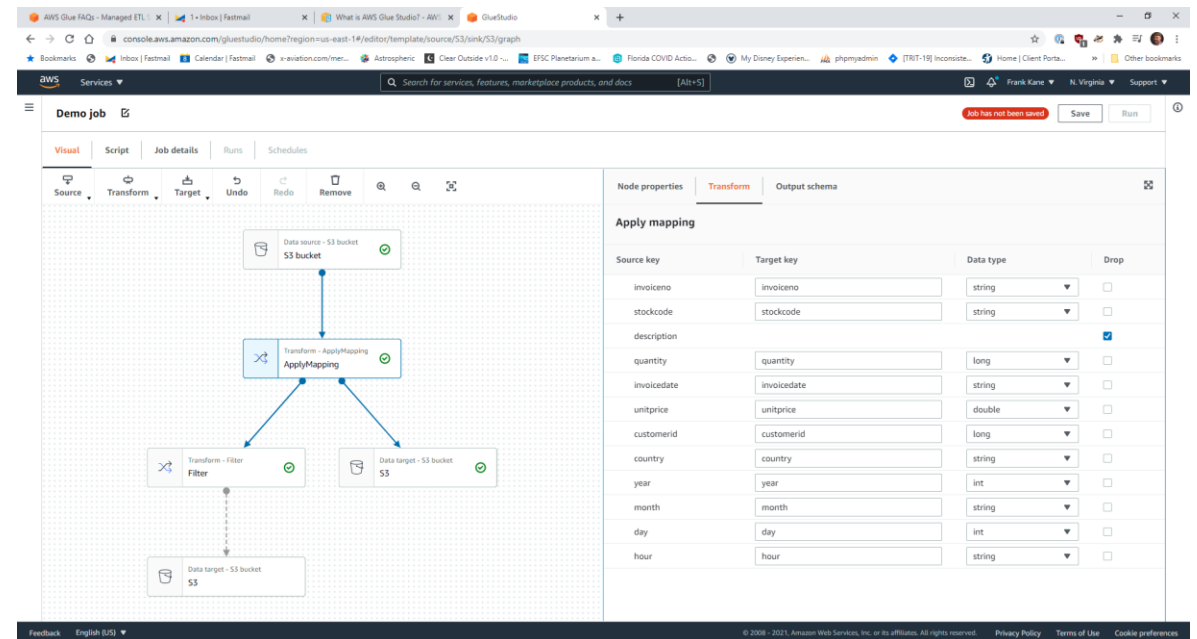
Glue and S3 Partitions

- Glue crawler will extract partitions based on how your S3 data is organized
- Think up front about how you will be querying your data lake in S3
- Example: devices send sensor data every hour
- Do you query primarily by time ranges?
 - If so, organize your buckets as yyyy/mm/dd/device
- Do you query primarily by device?
 - If so, organize your buckets as device/yyyy/mm/dd



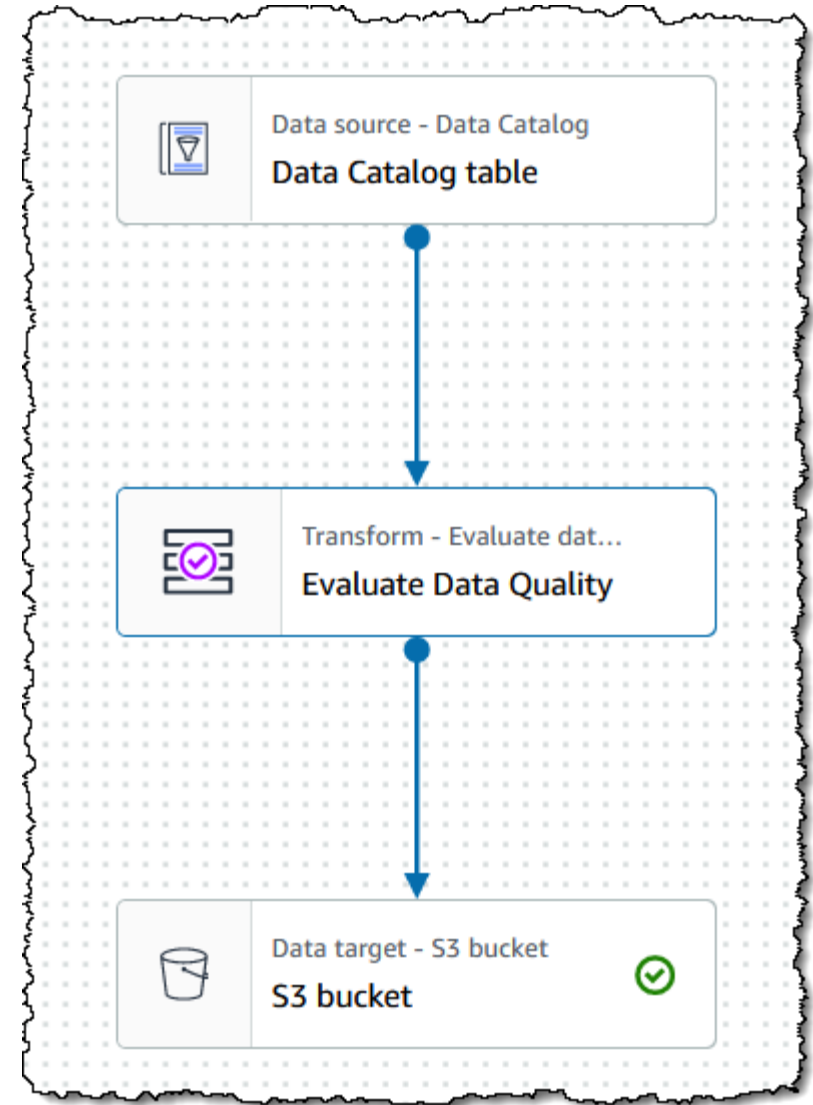
AWS Glue Studio

- Visual interface for ETL workflows
- Visual job editor
 - Create DAG's for complex workflows
 - Sources include S3, Kinesis, Kafka, JDBC
 - Transform / sample / join data
 - Target to S3 or Glue Data Catalog
 - Support partitioning
- Visual job dashboard
 - Overviews, status, run times



AWS Glue Data Quality

- Data quality rules may be created manually or recommended automatically
- Integrates into Glue jobs
- Uses Data Quality Definition Language (DQDL)
- Results can be used to fail the job, or just be reported to CloudWatch



AWS Glue Data Quality

AWS Glue > Tables > tickets > Ruleset details

RS1 Info

⌂

Actions ▾

Evaluate ruleset

Ruleset details

Description

Created by Data Quality Rule Recommendation

Database

yyz-tickets

Table

tickets

Created on

-

Last modified

November 22, 2022 at 17:48:14

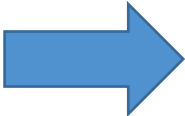
Tags

Rules (DQDL)

Data quality definition language

Copy

Rules = [
 RowCount between 1021908 and 4087632,
 IsComplete "TAG_NUMBER_MASKED",
 ColumnLength "TAG_NUMBER_MASKED" between 6 and 9,
 IsComplete "DATE_OF_INFRACTION",
 StandardDeviation "DATE_OF_INFRACTION" between 324.19 and 358.31,
 ColumnValues "DATE_OF_INFRACTION" between 20180100 and 20181232,
 IsComplete "TICKET_DATE",
 ColumnLength "TICKET_DATE" = 10,
 IsComplete "TICKET_NUMBER",
 StandardDeviation "TICKET_NUMBER" between 16289083634.63 and 18003724017.22,
 ColumnValues "TICKET_NUMBER" between 10102501 and 80081081053,
 IsComplete "OFFICER",
 StandardDeviation "OFFICER" between 107.99 and 119.35,
 ColumnValues "OFFICER" <= 394,
]



Rulesets

Data quality results

dqrun-89769d34ba974a7b9117d6cdf4a26b1108239b38

Name	DQ status	Status detail
RS1	DQ failed	Data quality score is 94.44%
Rule_1 RowCount between 1021908 and 4087632	Rule passed	
Rule_2 IsComplete "TAG_NUMBER_MASKED"	Rule passed	
Rule_3 ColumnLength "TAG_NUMBER_MASKED" between 6 and 9	Rule passed	
Rule_4 IsComplete "DATE_OF_INFRACTION"	Rule passed	
Rule_5 StandardDeviation "DATE_OF_INFRACTION" between 324.19 and 358.31	Rule failed	Expected type of column D
Rule_6 ColumnValues "DATE_OF_INFRACTION" between 20180100 and 20181232	Rule failed	Expected type of column D
Rule_7 IsComplete "TICKET_DATE"	Rule passed	
Rule_8 ColumnLength "TICKET_DATE" = 10	Rule passed	
Rule_9 IsComplete "TICKET_NUMBER"	Rule passed	
Rule_10 StandardDeviation "TICKET_NUMBER" between 16289083634.63 and 18003724017.22	Rule passed	
Rule_11 ColumnValues "TICKET_NUMBER" between 10102501 and 80081081053	Rule passed	
Rule_12 IsComplete "OFFICER"	Rule passed	

AWS Glue DataBrew

- A visual data preparation tool
 - UI for pre-processing large data sets
 - Input from S3, data warehouse, or database
 - Output to S3
- Over 250 ready-made transformations
- You create “recipes” of transformations that can be saved as jobs within a larger project
- May define data quality rules
- May create datasets with custom SQL from Redshift and Snowflake
- Security
 - Can integrate with KMS (with customer master keys only)
 - SSL in transit
 - IAM can restrict who can do what
 - CloudWatch & CloudTrail

```
{  
  "RecipeAction": {  
    "Operation": "NEST_TO_MAP",  
    "Parameters": {  
      "sourceColumns":  
        "[\"age\", \"weight_kg\", \"height_cm\"]",  
      "targetColumn": "columnName",  
      "removeSourceColumns": "true"  
    }  
  }  
}
```

Example: convert selected columns to key-value pairs

Handling Personally Identifiable Information (PII) in DataBrew Transformations

- Enable PII statistics in a DataBrew profile job to identify PII
- Substitution (REPLACE_WITH_RANDOM...)
- Shuffling (SHUFFLE_ROWS)
- Deterministic encryption (DETERMINISTIC_ENCRYPT)
- Probabilistic encryption (ENCRYPT)
- Decryption (DECRYPT)
- Nulling out or deletion (DELETE)
- Masking out (MASK_CUSTOM, _DATE, _DELIMITER, _RANGE)
- Hashing (CRYPTOGRAPHIC_HASH)



Amazon Athena

Serverless interactive queries of S3 data

What is Athena?

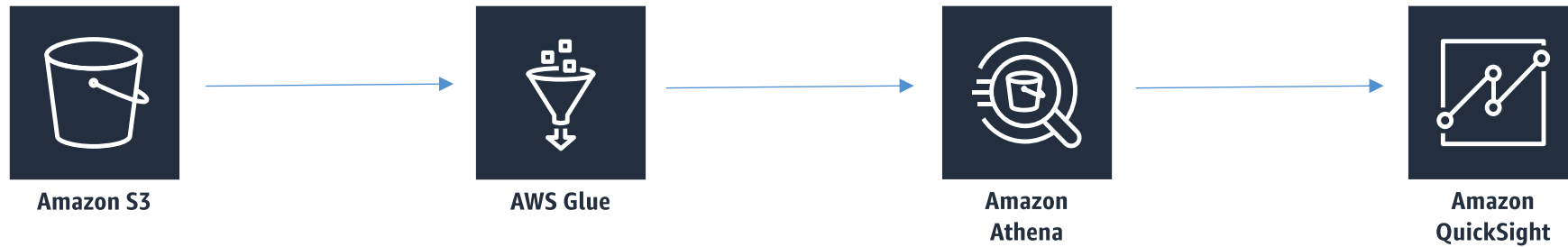
- Interactive query service for S3 (SQL)
 - No need to load data, it stays in S3
- Presto under the hood
- Serverless!
- Supports many data formats
 - CSV, TSV (human readable)
 - JSON (human readable)
 - ORC (columnar, splittable)
 - Parquet (columnar, splittable)
 - Avro (splittable)
 - Snappy, Zlib, LZO, Gzip compression
- Unstructured, semi-structured, or structured

Some examples

- Ad-hoc queries of web logs
- Querying staging data before loading to Redshift
- Analyze CloudTrail / CloudFront / VPC / ELB etc logs in S3
- Integration with Jupyter, Zeppelin, RStudio notebooks
- Integration with QuickSight
- Integration via ODBC / JDBC with other visualization tools

```
44 GO
45 SELECT p.Name AS ProductName,
46 NonDiscountSales = (OrderQty * UnitPr
47 Discounts = ((OrderQty * UnitPr
48 FROM Production.Product AS p
49 INNER JOIN Sales.SalesOrderData
50 ON p.ProductID = sod.ProductID
51 ORDER BY ProductName DESC;
52 GO
```

Athena + Glue



Athena Workgroups

- Can organize users / teams / apps / workloads into Workgroups
- Can control query access and track costs by Workgroup
- Integrates with IAM, CloudWatch, SNS
- Each workgroup can have its own:
 - Query history
 - Data limits (*you can limit how much data queries may scan by workgroup*)
 - IAM policies
 - Encryption settings



Athena cost model

- Pay-as-you-go
 - \$5 per TB scanned
 - Successful or cancelled queries count, failed queries do not.
 - No charge for DDL (CREATE/ALTER/DROP etc.)
- Save LOTS of money by using columnar formats
 - ORC, Parquet
 - Save 30-90%, and get better performance
- Glue and S3 have their own charges



Athena Security

- Access control
 - IAM, ACLs, S3 bucket policies
 - AmazonAthenaFullAccess / AWSQuicksightAthenaAccess
- Encrypt results at rest in S3 staging directory
 - Server-side encryption with S3-managed key (SSE-S3)
 - Server-side encryption with KMS key (SSE-KMS)
 - Client-side encryption with KMS key (CSE-KMS)
- Cross-account access in S3 bucket policy possible
- Transport Layer Security (TLS) encrypts in-transit (between Athena and S3)



Athena anti-patterns

- Highly formatted reports / visualization
 - That's what QuickSight is for
- ETL
 - Use Glue instead



Athena: Optimizing performance

- Use columnar data (ORC, Parquet)
- Small number of large files performs better than large number of small files
- Use partitions
 - If adding p

E command



Athena ACID transactions

- Powered by Apache Iceberg
 - Just add 'table_type' = 'ICEBERG' in your CREATE TABLE command
- Concurrent users can safely make row-level modifications
- Compatible with EMR, Spark, anything that supports Iceberg table format.
- Removes need for custom record locking
- Time travel operations
 - Recover data recently deleted with a SELECT statement
- Remember governed tables in Lake Formation? This is another way of getting ACID features in Athena.
- Benefits from periodic compaction to preserve performance

```
OPTIMIZE table REWRITE DATA  
USING BIN_PACK  
WHERE catalog = 'c1'
```

Athena Fine-Grained Access to AWS Glue Data Catalog

- IAM-based Database and table-level security
 - Broader than data filters in Lake Formation
 - Cannot restrict to specific table versions
- At a minimum you must have a policy that grants access to your database and the Glue Data Catalog in each region.

```
{
  "Sid": "DatabasePermissions",
  "Effect": "Allow",
  "Action": [
    "glue:GetDatabase",
    "glue:GetDatabases",
    "glue>CreateDatabase"
  ],
  "Resource": [
    "arn:aws:glue:us-east-1:123456789012:catalog",
    "arn:aws:glue:us-east-1:123456789012:database/default"
  ]
}
```

Athena Fine-Grained Access to AWS Glue Data Catalog

- You might have policies to restrict access to:
 - ALTER or CREATE DATABASE
 - CREATE TABLE
 - DROP DATABASE or DROP TABLE
 - MSCK REPAIR TABLE
 - SHOW DATABASES or SHOW TABLES
- Just need to map these operations to their IAM actions
- Example: DROP TABLE
- More examples at <https://docs.aws.amazon.com/athena/latest/ug/fine-grained-access-to-glue-resources.html>

```
{
  "Effect": "Allow",
  "Action": [
    "glue:GetDatabase",
    "glue:GetTable",
    "glue:DeleteTable",
    "glue:GetPartitions",
    "glue:GetPartition",
    "glue:DeletePartition"
  ],
  "Resource": [
    "arn:aws:glue:us-east-1:123456789012:catalog",
    "arn:aws:glue:us-east-1:123456789012:database/example_db",
    "arn:aws:glue:us-east-1:123456789012:table/example_db/test"
  ]
}
```

CREATE TABLE AS SELECT

- You might see this in the context of Amazon Athena
 - But it exists in other databases as well.
- Creates a new table from query results (CTAS)
- Can be used to create a new table that's a subset of another
- Can ALSO be used to convert data into a new underlying format
 - So this is a trick to get Athena to convert data stored in S3

```
CREATE TABLE new_table  
WITH (  
    format = 'Parquet',  
    write_compression = 'SNAPPY')  
AS SELECT *  
FROM old_table;
```

```
CREATE TABLE my_orc_ctas_table  
WITH (  
    external_location =  
    's3://my_athena_results/my_orc_stas_table/',  
    format = 'ORC')  
AS SELECT *  
FROM old_table;
```

AWS Managed AI Services

Why AWS AI Managed Services?

- **AWS AI Services are pre-trained ML services for your use case**
- **Responsiveness and Availability**
- **Redundancy and Regional Coverage:** deployed across multiple Availability Zones and AWS regions
- **Performance:** specialized CPU and GPUs for specific use-cases for cost saving
- **Token-based pricing:** pay for what you use
- **Provisioned throughput:** for predictable workloads, cost savings and predictable performance



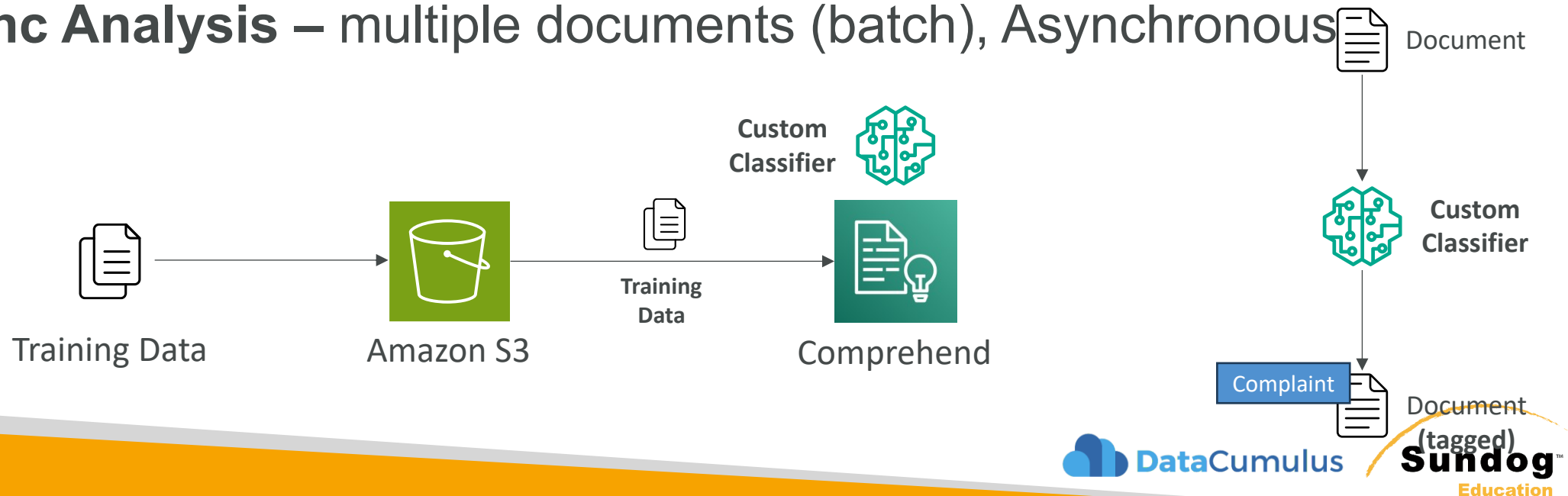
Amazon Comprehend



- For **Natural Language Processing – NLP**
- Fully managed and serverless service
- Uses machine learning to find insights and relationships in text
 - Language of the text
 - Extracts key phrases, places, people, brands, or events
 - Understands how positive or negative the text is
 - Analyzes text using tokenization and parts of speech
 - Automatically organizes a collection of text files by topic
- Sample use cases:
 - analyze customer interactions (emails) to find what leads to a positive or negative experience
 - Create and groups articles by topics that Comprehend will uncover

Comprehend – Custom Classification

- Organize documents into categories (classes) that you define
- Example: categorize customer emails so that you can provide guidance based on the type of the customer request
- Supports different document types (text, PDF, Word, images...)
- **Real-time Analysis** – single document, synchronous
- **Async Analysis** – multiple documents (batch), Asynchronous



Named Entity Recognition (NER)

- **NER** – Extracts predefined, general-purpose entities like people, places, organizations, dates, and other standard categories, **from text**

Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC credit card account 1111-0000-1111-0008 has a minimum payment of \$24.53 that is due by July 31st. Based on your autopay settings, we will withdraw your payment on the due date from your bank account number XXXXXX1111 with the routing number XXXXX0000.

Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to Alice at sunspa@mail.com.

I enjoyed visiting the spa. It was also very expensive. The amenities were ok but the service made the spa a great experience.

Entity 10: 123 Main St ✕

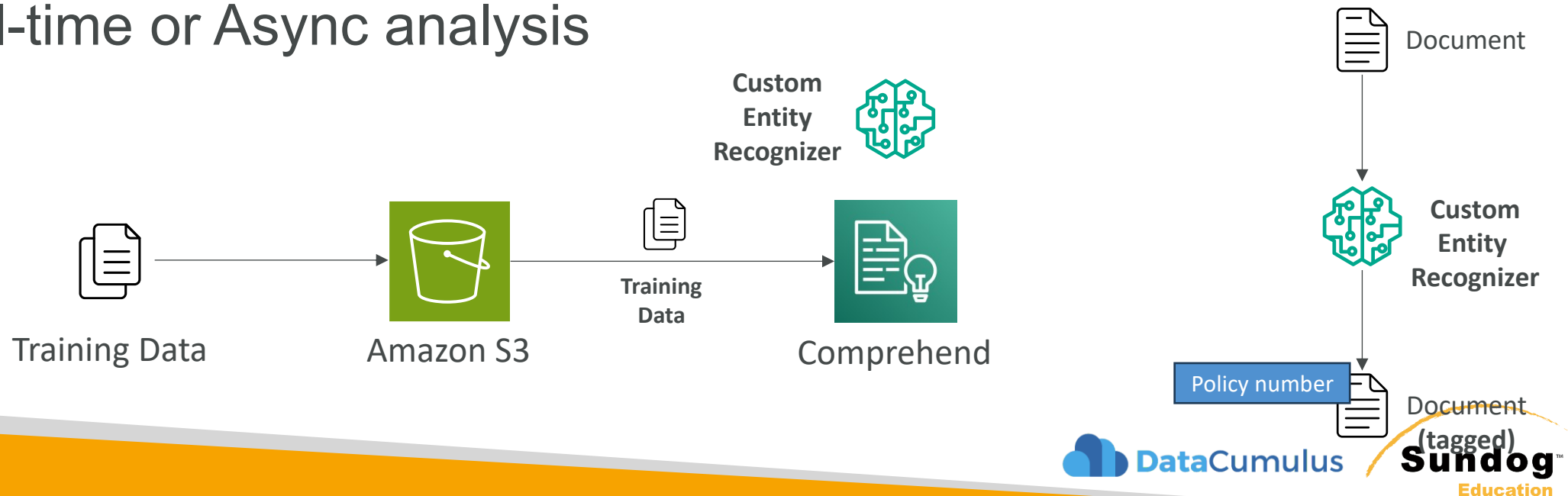
Entity: LOCATION

Confidence: 0.98

Entity ▾	Type ▾
Zhang Wei	Person
John	Person
AnyCompany Financial Services, LLC	Organization
1111-0000-1111-0008	Other
\$24.53	Quantity
July 31st	Date
XXXXXX1111	Other
XXXXX0000	Other
Sunshine Spa	Organization
123 Main St	Location

Comprehend – Custom Entity Recognition

- Analyze text for specific terms and noun-based phrases
- Extract terms like policy numbers, or phrases that imply a customer escalation, anything specific to your business
- Train the model with custom data such as a list of the entities and documents that contain them
- Real-time or Async analysis



Comprehend – Custom Models

- You can create custom models for entity recognition or document classification
 - Trained on your own data
- Comprehend manages your model versioning
- Custom models may be copied between AWS accounts
 - Attach IAM policy to a model version, authorizing the other account
 - Other account then imports the model
 - Must be in same region
 - Need its ARN, region, and optional KMS key
 - Can be done from the Comprehend console

Amazon Translate



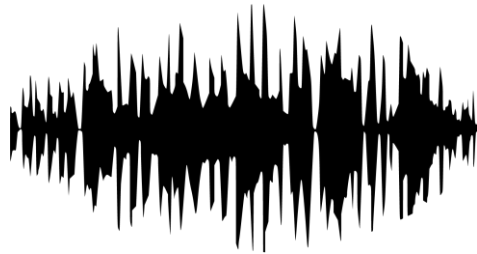
- Natural and accurate **language translation**
- Amazon Translate allows you to **localize content** - such as websites and applications - for **international users**, and to easily translate large volumes of text efficiently.

Source language		Target language
Auto (auto) ▼	↔	French (fr) ▼
Hi my name is Stéphane		Bonjour, je m'appelle Stéphane.
		Portuguese (pt) ▼
		Oi, meu nome é Stéphane.
		Hindi (hi) ▼
		हाय मेरा नाम स्टीफन है

Amazon Transcribe



- Automatically **convert speech to text**
- Uses a **deep learning process** called **automatic speech recognition (ASR)** to convert speech to text quickly and accurately
- **Automatically remove Personally Identifiable Information (PII) using Redaction**
- **Supports Automatic Language Identification for multi-lingual audio**
- Use cases:
 - transcribe customer service calls
 - automate closed captioning and subtitling
 - generate metadata for media assets to create a fully searchable archive



*"Hello my name is Stéphane.
I hope you're enjoying the course!"*

Transcribe – Toxicity Detection

- ML-powered, voice-based toxicity detection capability
- Leverages speech cues: tone and pitch, and text-based cues
- Toxicity categories: sexual harassment, hate speech, threat, abuse, profanity, insult, and graphic....

Transcription preview
You can see the first 5,000 characters of the transcription text below. To download the full text, choose Download full transcript.

Text Audio identification Subtitles **Toxicity**

Toxicity score -- 0.0 to 0.4 0.4 to 0.8 0.8 to 1.0 [Info](#)
Higher score means higher toxicity.

Filters [Info](#) [Reset](#)
Filter out toxic content by increasing threshold values below.

Toxicity Score 0 1
Profanity 0 1
Hate speech 0 1
Sexual 0 1
Insults 0 1

Transcription text (with toxicity highlights):
... You are a waste of a human being. You've done ABSOLUTELY NOTHING that's important. ... You know, you're not even human. You don't deserve the word human because you are that much of a ... disgrace to people all around the globe, worldwide. It's a ... Are you proud of having accomplished absolutely nothing in your life? You have only done bad things in this world. You do not deserve to live, y ... t. ...

Toxicity Categories

Profanity: Speech that contains words, phrases, or acronyms that are impolite, vulgar, or offensive.

Hate speech: Speech that criticizes, insults, denounces, or dehumanizes a person or group on the basis of an identity (such as race, ethnicity, gender, religion, sexual orientation, ability, and national origin).

Sexual: Speech that indicates sexual interest, activity, or arousal using direct or indirect references to body parts, physical traits, or sex.

Insults: Speech that includes demeaning, humiliating, mocking, insulting, or belittling language. This type of language is also labeled as bullying.

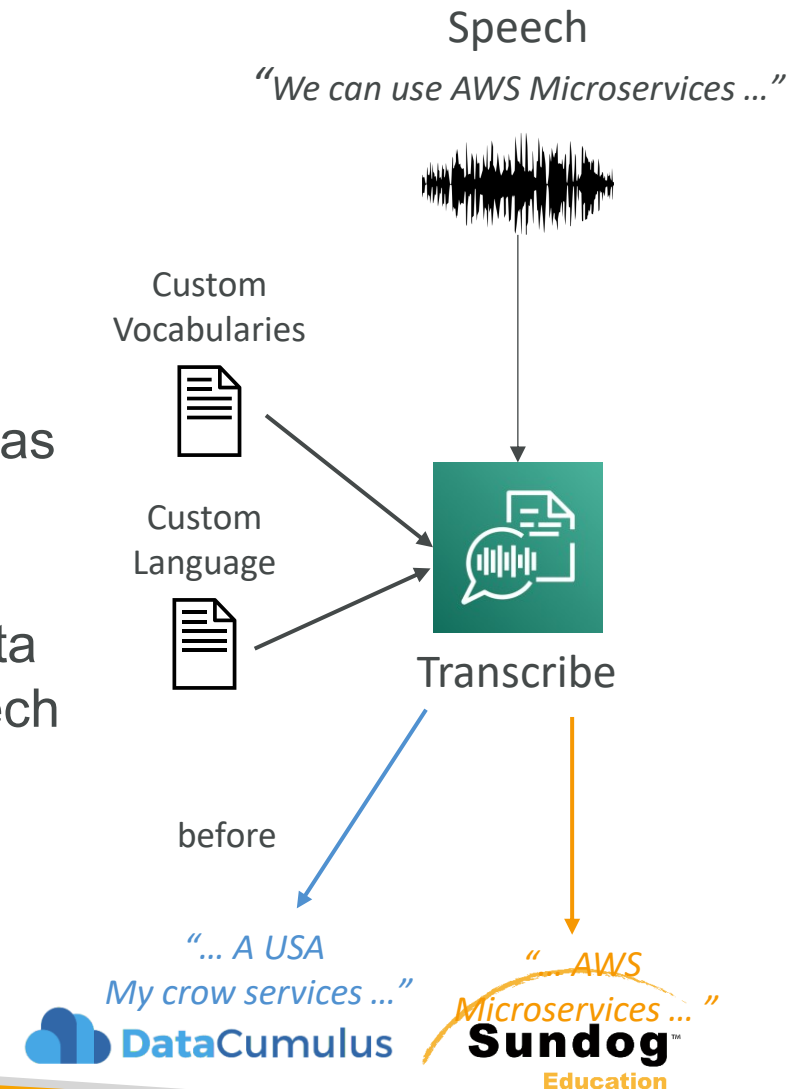
Violence or threat: Speech that includes threats seeking to inflict pain, injury, or hostility toward a person or group.

Graphic: Speech that uses visually descriptive and unpleasantly vivid imagery. This type of language is often intentionally verbose to amplify a recipient's discomfort.

Harassment or abusive: Speech intended to affect the psychological well-being of the recipient, including demeaning and objectifying terms.

Amazon Transcribe – Improving Accuracy

- Allows Transcribe to capture domain-specific or non-standard terms (e.g., technical words, acronyms, jargon...)
- **Custom Vocabularies (for words)**
 - Add specific words, phrases, domain-specific terms
 - Good for brand names, acronyms...
 - Increase recognition of a new word by providing hints (such as pronunciation..)
- **Custom Language Models (for context)**
 - Train Transcribe model on your own domain-specific text data
 - Good for transcribing large volumes of domain-specific speech
 - Learn the context associated with a given word
- **Note:** use both for the highest transcription accuracy

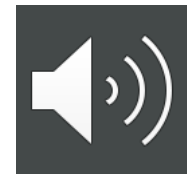


Amazon Polly



- Turn text into lifelike speech using deep learning
- Allowing you to create applications that talk

*Hi! My name is Stéphane
and this is a demo of Amazon Polly*



Polly – Advanced Features

- **Lexicons**

- Define how to read certain specific pieces of text
- AWS => “Amazon Web Services”
- W3C => “World Wide Web Consortium”

- **SSML - Speech Synthesis Markup Language**

- Markup for your text to indicate how to pronounce it
- Example: “Hello, <break> how are you?”

- **Voice engine:** generative, long-form, neural, standard...

- **Speech mark:**

- Encode where a sentence/word starts or ends in the audio
- Helpful for lip-syncing or highlight words as they’re spoken

Action	SSML tag
Adding a pause	<break>
Emphasizing words	<emphasis>
Specifying another language for specific words	<lang>
Placing a custom tag in your text	<mark>
Adding a pause between paragraphs	<p>
Using phonetic pronunciation	<phoneme>
Controlling volume, speaking rate, and pitch	<prosody>
Setting a maximum duration for synthesized speech	<prosody amazon:max-duration>
Adding a pause between sentences	<s>
Controlling how special types of words are spoken	<say-as>
Identifying SSML-enhanced text	<speak>
Pronouncing acronyms and abbreviations	<sub>
Improving pronunciation by specifying parts of speech	<w>
Adding the sound of breathing	<amazon:auto-breaths>
Newscaster speaking style	<amazon:domain name="news">
Adding dynamic range compression	<amazon:effect name="drc">
Speaking softly	<amazon:effect



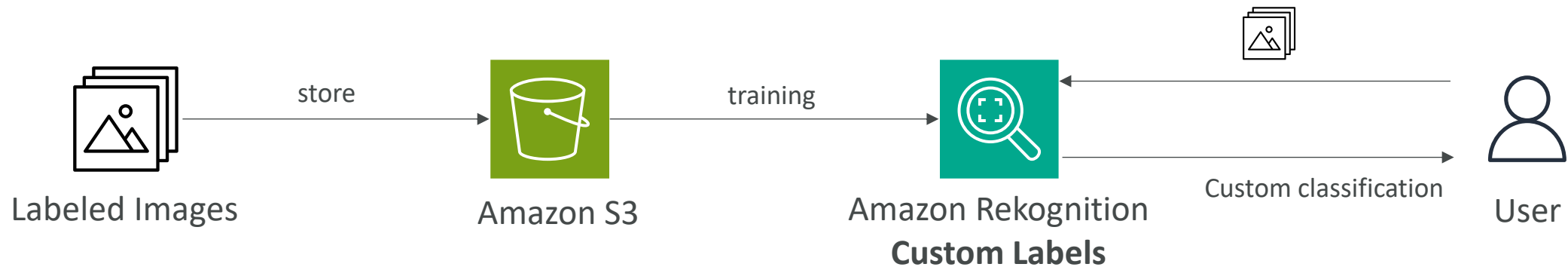
Amazon Rekognition



- Find **objects, people, text, scenes** in **images and videos** using ML
- **Facial analysis** and **facial search** to do user verification, people counting
- Create a database of “familiar faces” or compare against celebrities
- Use cases:
 - Labeling
 - Content Moderation
 - Text Detection
 - Face Detection and Analysis (gender, age range, emotions...)
 - Face Search and Verification
 - Celebrity Recognition
 - **Pathing** (ex: for sports game analysis)

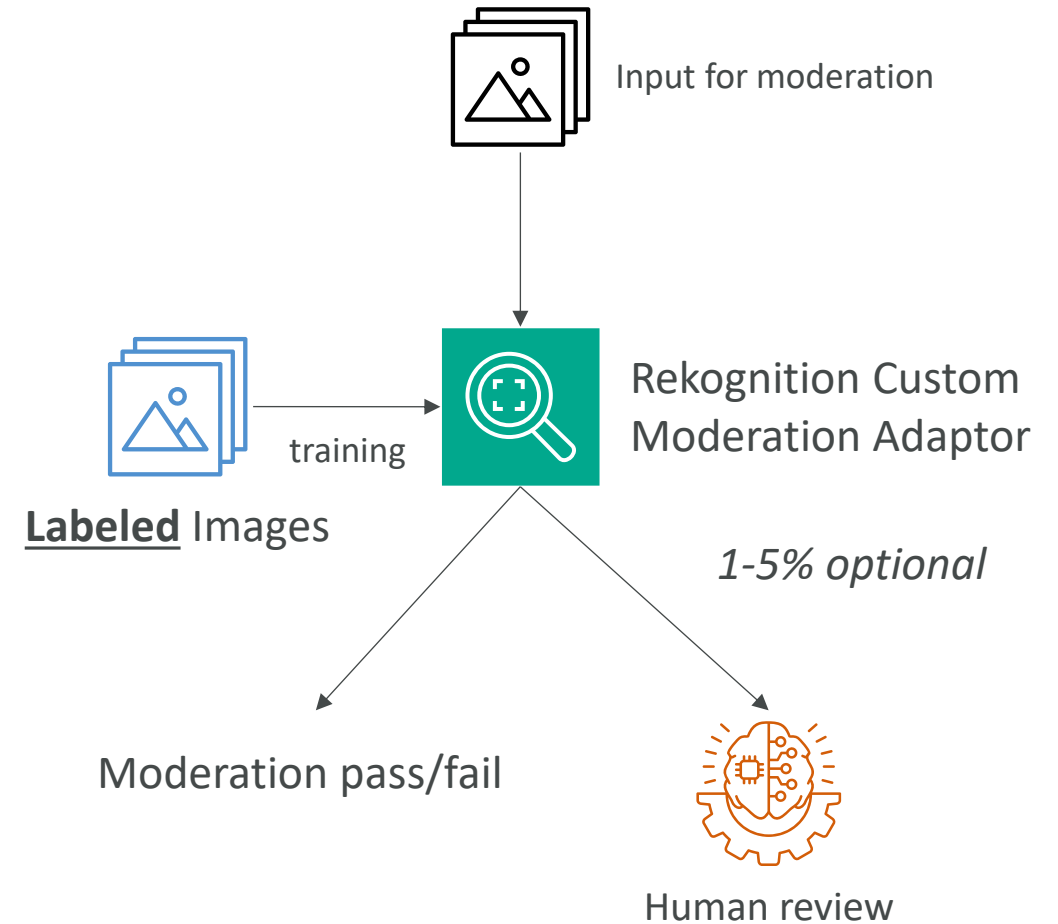
Amazon Rekognition – Custom Labels

- Examples: find your logo in social media posts, identify your products on stores shelves (National Football League – NFL – uses it to find their logo in pictures)
- **Label** your training images and upload them to Amazon Rekognition
- Only needs a few hundred images or less
- Amazon Rekognition creates a custom model on your images set
- New subsequent images will be categorized the custom way you have defined

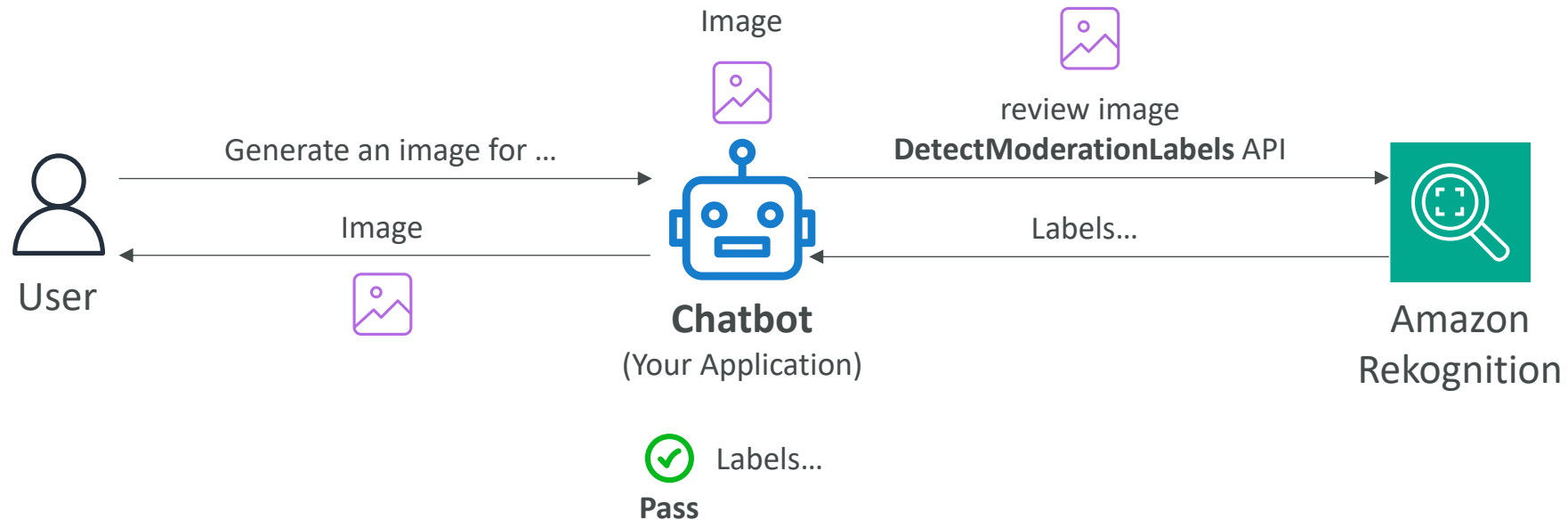


Amazon Rekognition – Content Moderation

- Automatically detect inappropriate, unwanted, or offensive content
- Example: filter out harmful images in social media, broadcast media, advertising...
- Bring down human review to 1-5% of total content volume
- **Custom Moderation Adaptors**
 - Extends Rekognition capabilities by providing your own **labeled** set of images
 - Enhances the accuracy of Content Moderation or create a specific use case of Moderation



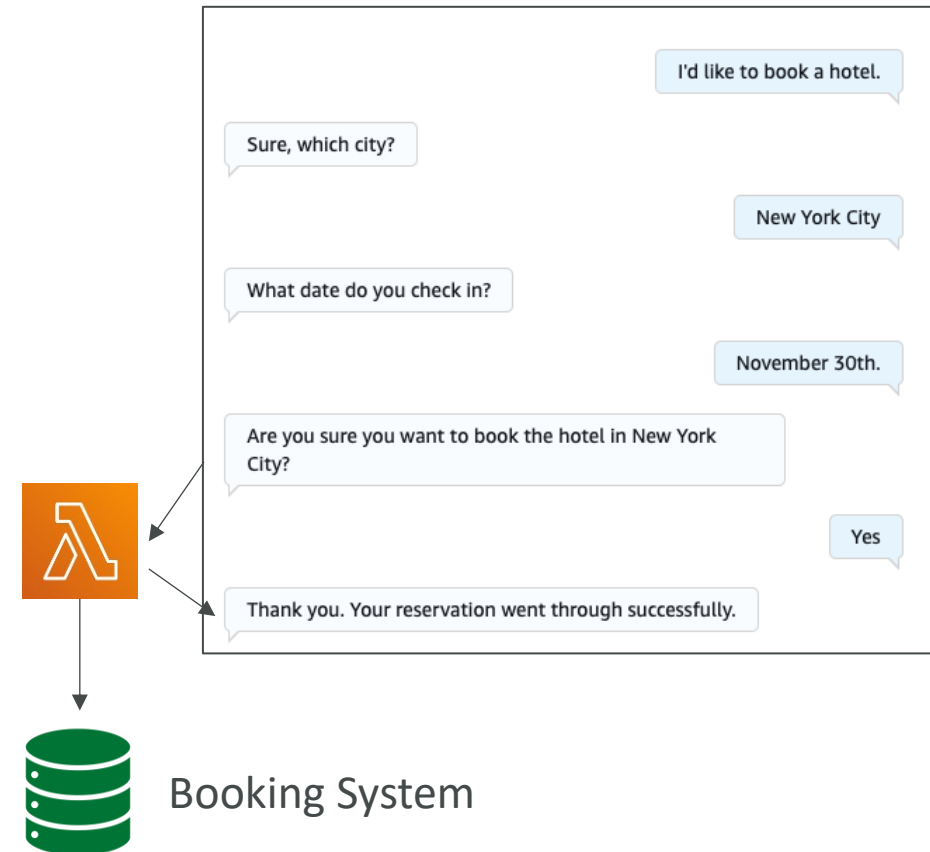
Content Moderation API – Diagram



Amazon Lex



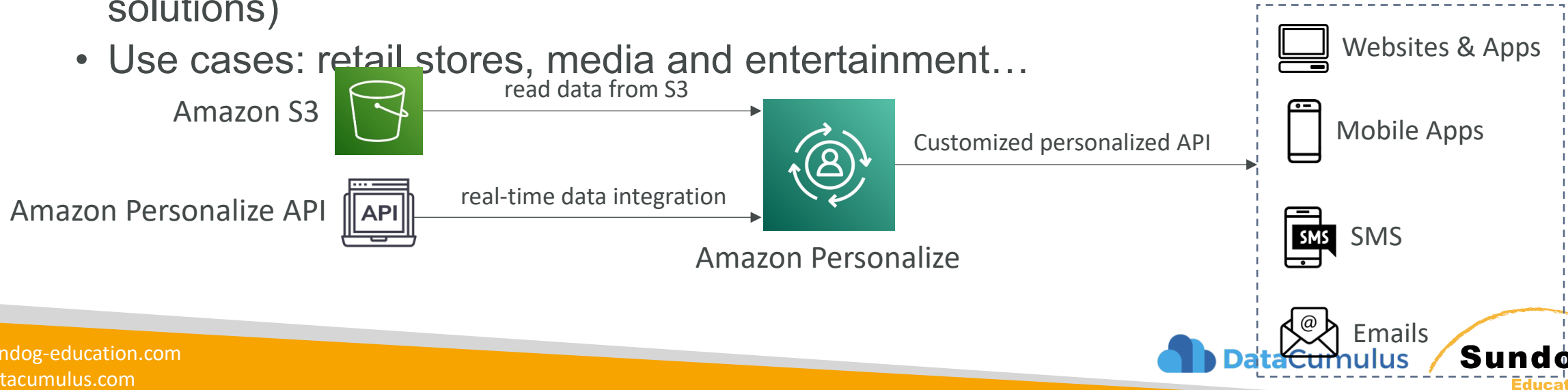
- Build chatbots quickly for your applications using voice and text
- Example: a chatbot that allows your customers to order pizzas or book a hotel
- Supports multiple languages
- Integration with AWS Lambda, Connect, Comprehend
- The bot automatically understands the user intent to invoke the correct Lambda function to “fulfill the intent”
- The bot will ask for “Slots” (input parameters) if necessary



Amazon Personalize



- Fully managed ML-service to build apps with real-time personalized recommendations
- Example: personalized product recommendations/re-ranking, customized direct marketing
 - Example: User bought gardening tools, provide recommendations on the next one to buy
- Same technology used by Amazon.com
- Integrates into existing websites, applications, SMS, email marketing systems, ...
- Implement in days, not months (you don't need to build, train, and deploy ML solutions)
- Use cases: retail stores, media and entertainment...



Amazon Personalize – Recipes

- Algorithms that are prepared for specific use cases
- You must provide the training configuration on top of the recipe
- Example recipes:
 - Recommending items for users (**USER_PERSONALIZATION** recipes)
 - User-Personalization-v2
 - Ranking items for a user (**PERSONALIZED_RANKING** recipes)
 - Personalized-Ranking-v2
 - Recommending trending or popular items (**POPULAR_ITEMS** recipes)
 - Trending-Now, Popularity-Count
 - Recommending similar items (**RELATED_ITEMS** recipes)
 - Similar-Items
 - Recommending the next best action (**PERSONALIZED_ACTIONS** recipes)
 - Next-Best-Action
 - Getting user segments (**USER_SEGMENTATION** recipes)
 - Item-Affinity
- **NOTE:** recipes and personalize are for recommendations

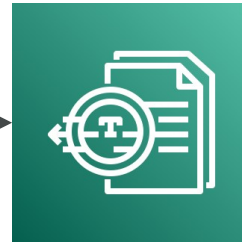
Amazon Textract



- Automatically extracts text, handwriting, and data from any scanned documents using AI and ML



analyze



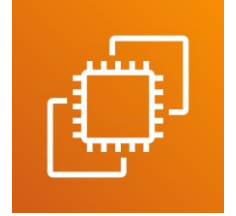
Amazon Textract

result

```
{
  "Document ID": "123456789-005",
  "Name": "",
  "SEX": "F",
  "DOB": "23.05.1997",
  ...
}
```

- Extract data from forms and tables
- Read and process any type of document (PDFs, images, ...)
- Use cases:
 - Financial Services (e.g., invoices, financial reports)
 - Healthcare (e.g., medical records, insurance claims)
 - Public Sector (e.g., tax forms, ID documents, passports)

Amazon EC2



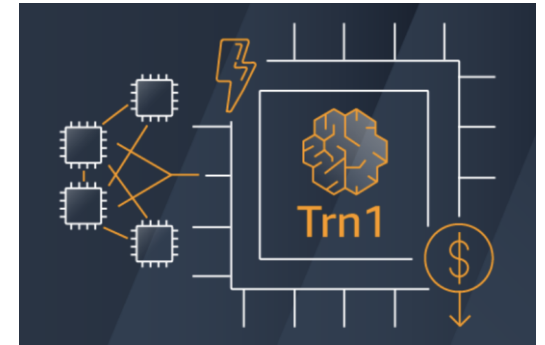
- EC2 is one of the most popular of AWS' offerings
- EC2 = Elastic Compute Cloud = Infrastructure as a Service
- It mainly consists in the capability of :
 - Renting virtual machines (EC2)
 - Storing data on virtual drives (EBS)
 - Distributing load across machines (ELB)
 - Scaling the services using an auto-scaling group (ASG)
- Knowing EC2 is fundamental to understand how the Cloud works

EC2 sizing & configuration options

- Operating System (**OS**): Linux, Windows or Mac OS
- How much compute power & cores (**CPU**)
- How much random-access memory (**RAM**)
- How much storage space:
 - Network-attached (**EBS & EFS**)
 - hardware (**EC2 Instance Store**)
- Network card: speed of the card, Public IP address
- Firewall rules: **security group**
- Bootstrap script (configure at first launch): EC2 User Data

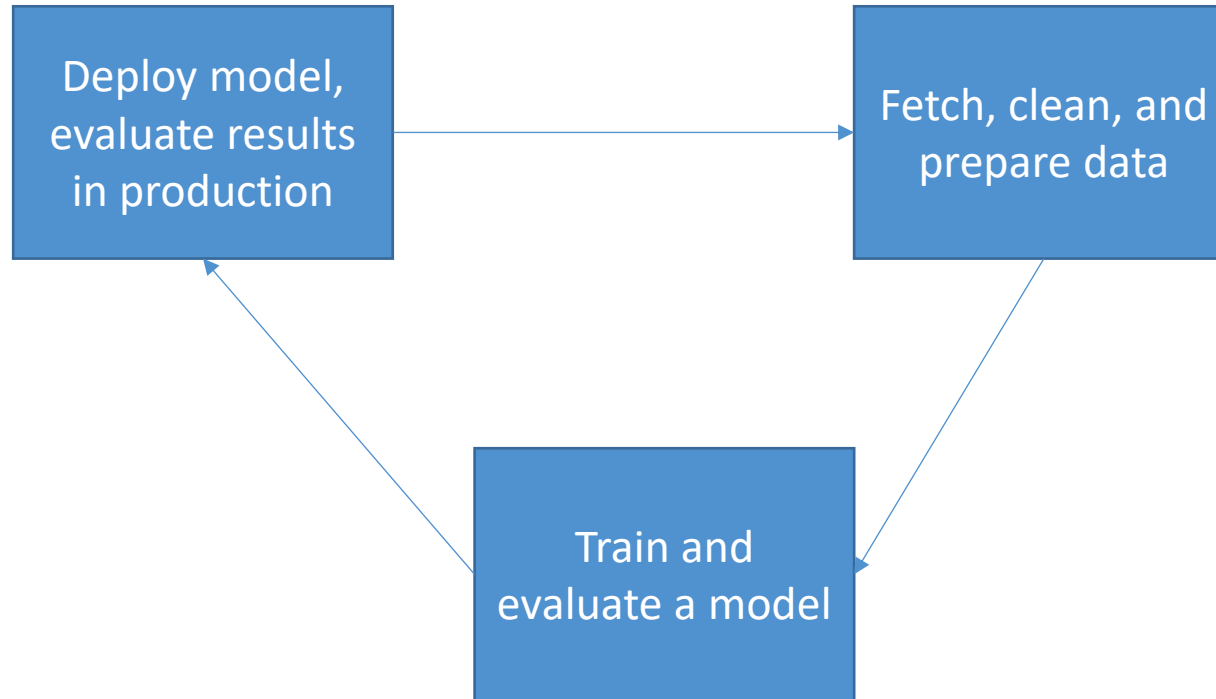
Amazon's Hardware for AI

- GPU-based EC2 Instances (P3, P4, P5..., G3...G6...)
- **AWS Trainium**
 - ML chip built to perform Deep Learning on 100B+ parameter models
 - Trn1 instance has for example 16 Trainium Accelerators
 - 50% cost reduction when training a model
- **AWS Inferentia**
 - ML chip built to deliver inference at high performance and low cost
 - Inf1, Inf2 instances are powered by AWS Inferentia
 - Up to 4x throughput and 70% cost reduction

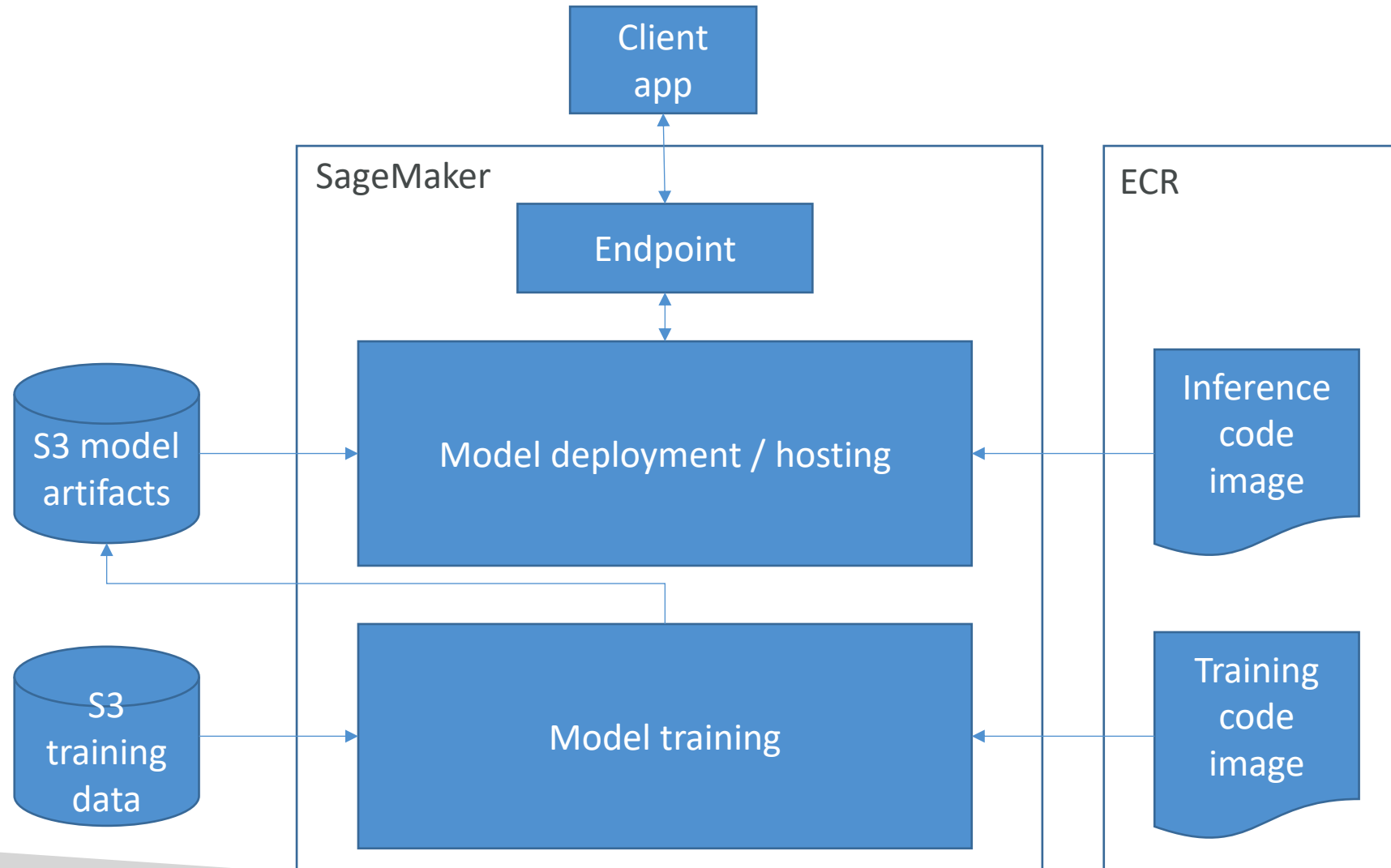


Amazon SageMaker AI

SageMaker AI is built to handle the entire machine learning workflow.

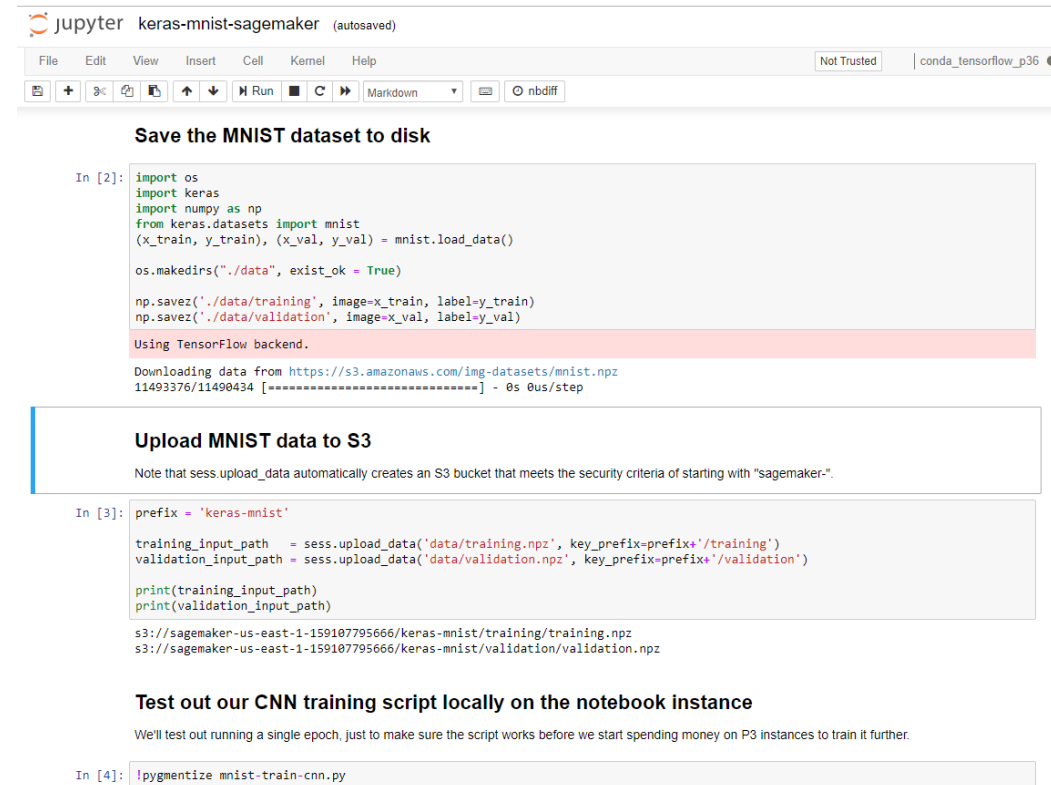


SageMaker AI Training & Deployment



SageMaker Notebooks can direct the process

- Notebook Instances on EC2 are spun up from the console
 - S3 data access
 - Scikit_learn, Spark, Tensorflow
 - Wide variety of built-in models
 - Ability to spin up training instances
 - Ability to deploy trained models for making predictions at scale



The screenshot shows a Jupyter Notebook interface with the title 'keras-mnist-sagemaker (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations and execution. The notebook content is divided into sections:

- Save the MNIST dataset to disk**: This section contains a code cell (In [2]:) that imports necessary libraries (os, keras, numpy, mnist) and saves the MNIST dataset to local files. The code includes comments and a status bar indicating the use of the TensorFlow backend and the download progress of the dataset from S3.
- Upload MNIST data to S3**: This section contains a code cell (In [3]:) that uses the SageMaker session's upload_data method to upload the training and validation data to an S3 bucket. The code also prints the input paths for both datasets.
- Test out our CNN training script locally on the notebook instance**: This section contains a code cell (In [4]:) that runs the command '!pygmentize mnist-train-cnn.py' to test the training script locally.

So can the SageMaker AI console

The screenshot displays the Amazon SageMaker AI console interface. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information (Frank Kane, N. Virginia, Support). The left sidebar shows the 'Amazon SageMaker' menu with options like Dashboard, Search, Ground Truth, Notebook, Training, and Inference. The 'Training' section is expanded, showing 'Training jobs' as the selected option. The main content area, titled 'Training jobs', features a search bar and a table of training jobs. The table has columns for Name, Creation time, Duration, and Status. All listed jobs are in a 'Completed' status. At the bottom of the console, there is a footer with 'Feedback', 'English (US)', and copyright information.

Name	Creation time	Duration	Status
sagemaker-tensorflow-190926-1513-010-6f1bf0d5	Sep 26, 2019 15:30 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-009-53c7332e	Sep 26, 2019 15:30 UTC	3 minutes	Completed
sagemaker-tensorflow-190926-1513-008-a53612ca	Sep 26, 2019 15:26 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-007-b06a4d5b	Sep 26, 2019 15:26 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-006-167e1e3c	Sep 26, 2019 15:22 UTC	3 minutes	Completed
sagemaker-tensorflow-190926-1513-005-fd89504c	Sep 26, 2019 15:21 UTC	5 minutes	Completed
sagemaker-tensorflow-190926-1513-004-4a31fc9c	Sep 26, 2019 15:17 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-003-187d6ee7	Sep 26, 2019 15:17 UTC	5 minutes	Completed
sagemaker-tensorflow-190926-1513-002-0fd0ad85	Sep 26, 2019 15:14 UTC	4 minutes	Completed
sagemaker-tensorflow-190926-1513-001-7f7cf798	Sep 26, 2019 15:14 UTC	3 minutes	Completed

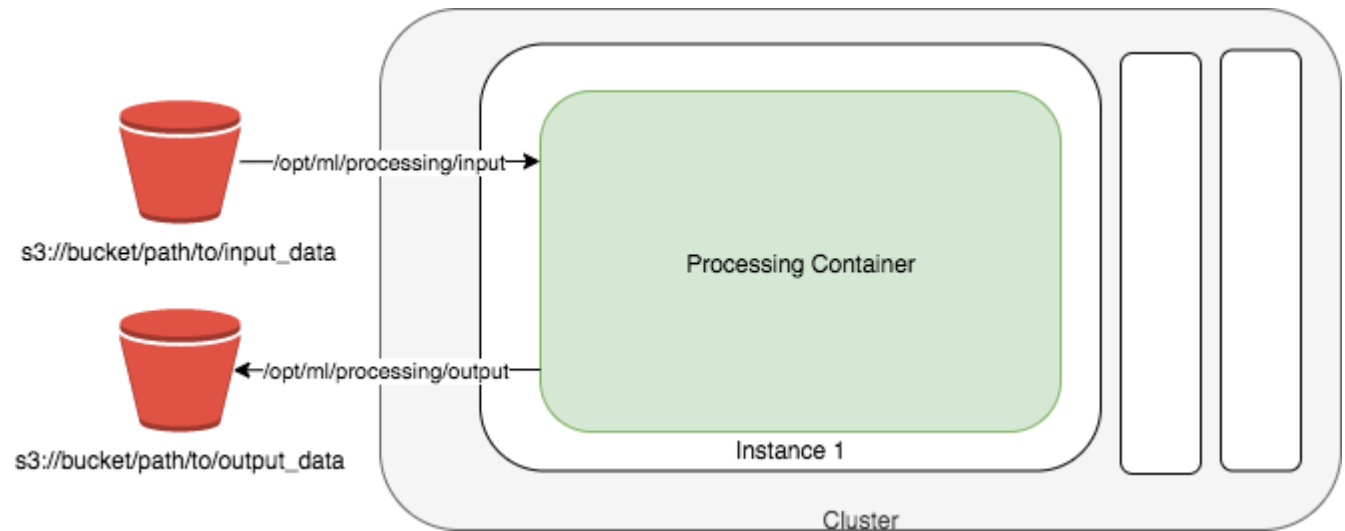
Data prep on SageMaker AI

- Data usually comes from S3
 - Ideal format varies with algorithm – often it is RecordIO / Protobuf
- Can also ingest from Athena, EMR, Redshift, and Amazon Keyspaces DB
- Apache Spark integrates with SageMaker
 - More on this later...
- Scikit_learn, numpy, pandas all at your disposal within a notebook



SageMaker Processing

- Processing jobs
 - Copy data from S3
 - Spin up a processing container
 - SageMaker built-in or user provided
 - Output processed data to S3



Training on SageMaker AI

- Create a training job
 - URL of S3 bucket with training data
 - ML compute resources
 - URL of S3 bucket for output
 - ECR path to training code
- Training options
 - Built-in training algorithms
 - Spark MLlib
 - Custom Python Tensorflow / MXNet code
 - PyTorch, Scikit-Learn, RLEstimator
 - XGBoost, Hugging Face, Chainer
 - Your own Docker image
 - Algorithm purchased from AWS marketplace



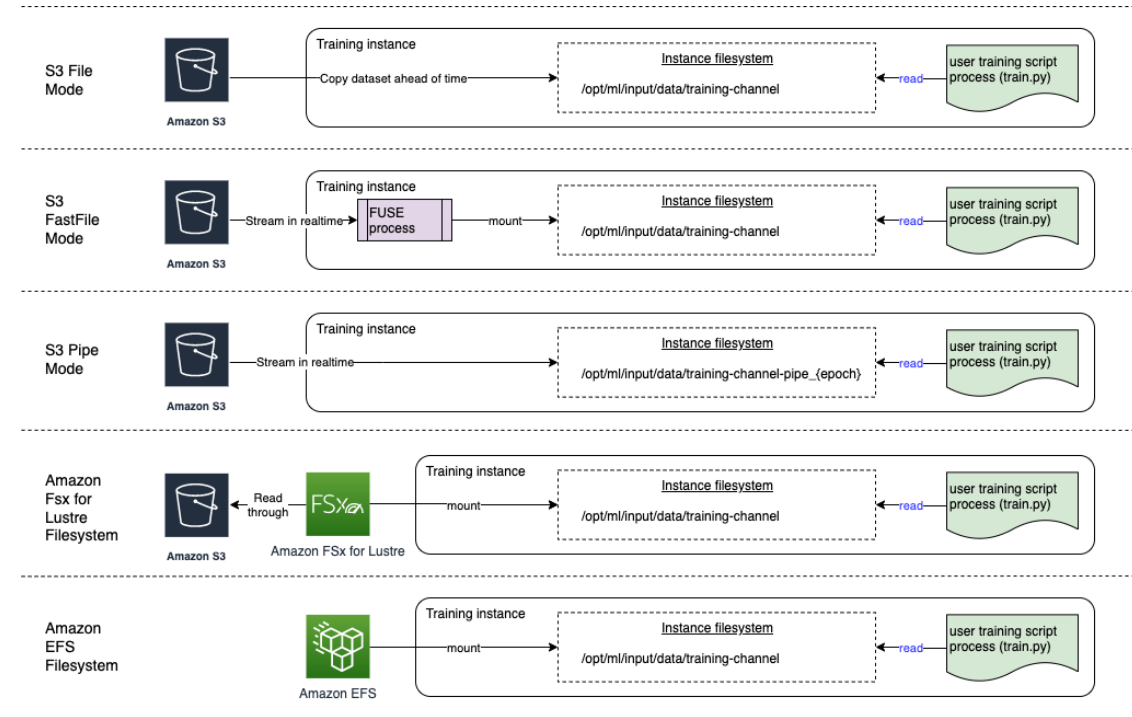
Deploying Trained Models

- Save your trained model to S3
- Can deploy two ways:
 - Persistent endpoint for making individual predictions on demand
 - SageMaker Batch Transform to get predictions for an entire dataset
- Lots of cool options
 - Inference Pipelines for more complex processing
 - SageMaker Neo for deploying to edge devices
 - Elastic Inference for accelerating deep learning models
 - Automatic scaling (increase # of endpoints as needed)
 - Shadow Testing evaluates new models against currently deployed model to catch errors



SageMaker AI Input Modes

- S3 File Mode
 - Default; **copies** training data from S3 to local directory in Docker container
- S3 Fast File Mode
 - Akin to “pipe mode”; training can begin without waiting to download data
 - Can do random access, but works best with sequential access
- Pipe Mode
 - Streams data directly from S3
 - Mainly replaced by Fast File
- Amazon S3 Express One Zone
 - High-performance storage class in one AZ
 - Works with file, fast file, and pipe modes
- Amazon FSx for Lustre
 - Scales to 100's of GB of throughput and millions of IOPS with low latency
 - Single AZ, requires VPC
- Amazon EFS
 - Requires data to be in EFS already, requires VPC



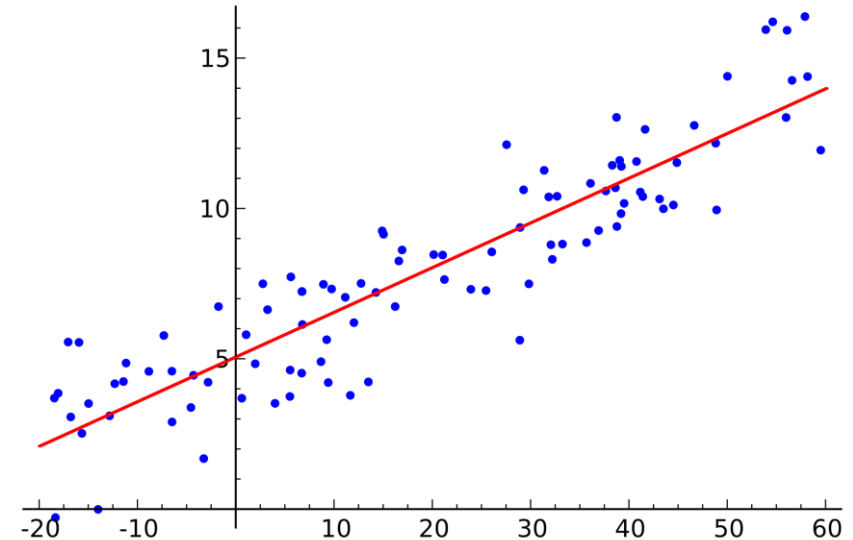
<https://docs.aws.amazon.com/sagemaker/latest/dg/model-access-training-data.html>

SageMaker AI's Built-In Algorithms

Linear Learner

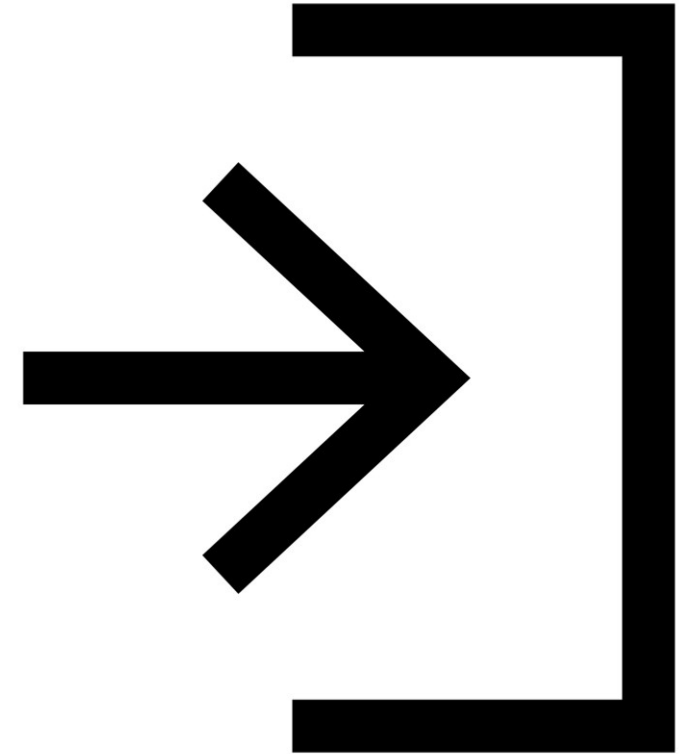
Linear Learner: What's it for?

- Linear regression
 - Fit a line to your training data
 - Predications based on that line
- Can handle both regression (numeric) predictions and classification predictions
 - For classification, a linear threshold function is used.
 - Can do binary or multi-class



Linear Learner: What training input does it expect?

- RecordIO-wrapped protobuf
 - Float32 data only!
- CSV
 - First column assumed to be the label
- File or Pipe mode both supported



Linear Learner: How is it used?

- Preprocessing
 - Training data must be *normalized* (so all features are weighted the same)
 - Linear Learner can do this for you automatically
 - Input data should be *shuffled*
- Training
 - Uses stochastic gradient descent
 - Choose an optimization algorithm (Adam, AdaGrad, SGD, etc)
 - Multiple models are optimized in parallel
 - Tune L1, L2 regularization
- Validation
 - Most optimal model is selected



Linear Learner: Important Hyperparameters

- `Balance_multiclass_weights`
 - Gives each class equal importance in loss functions
- `Learning_rate`, `mini_batch_size`
- `L1`
 - Regularization
- `Wd`
 - Weight decay (L2 regularization)
- `target_precision`
 - Use with `binary_classifier_model_selection_criteria` set to `recall_at_target_precision`
 - Holds precision at this value while maximizing recall
- `target_recall`
 - Use with `binary_classifier_model_selection_criteria` set to `precision_at_target_recall`
 - Holds recall at this value while maximizing precision



Linear Learner: Instance Types

- Training
 - Single or multi-machine CPU or GPU
 - Multi-GPU does not help



Example

jupyter linear_learner_mnist (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted | conda_python3

Run nbdiff

Data inspection

Once the dataset is imported, it's typical as part of the machine learning process to inspect the data, understand the distributions, and determine what type(s) of preprocessing might be needed. You can perform those tasks right here in the notebook. As an example, let's go ahead and look at one of the digits that is part of the dataset.

In [3]:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (2,10)

def show_digit(img, caption='', subplot=None):
    if subplot==None:
        _,(subplot)=plt.subplots(1,1)
    imgr=img.reshape((28,28))
    subplot.axis('off')
    subplot.imshow(imgr, cmap='gray')
    plt.title(caption)

show_digit(train_set[0][30], 'This is a {}'.format(train_set[1][30]))
```

This is a 3



Data conversion

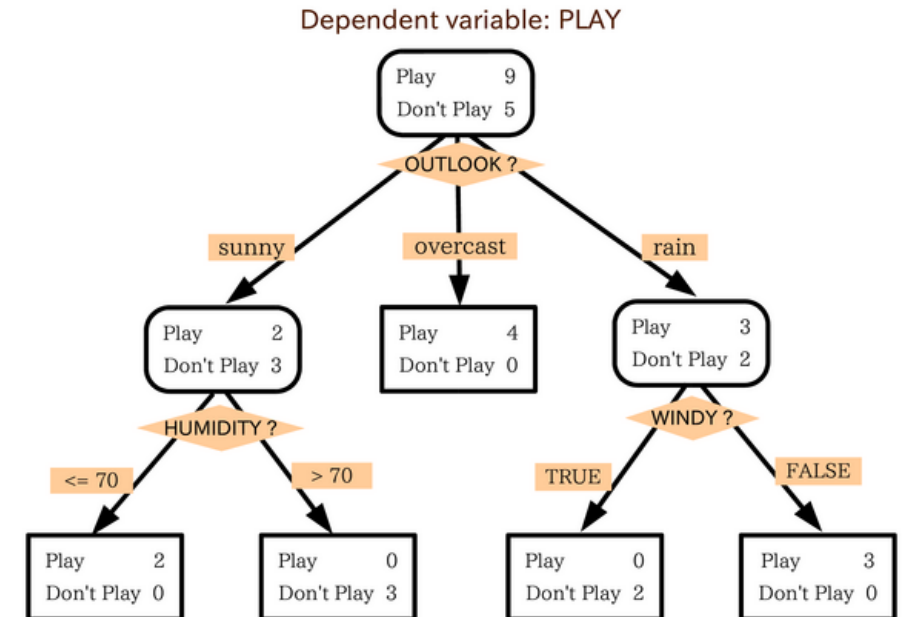
Since algorithms have particular input and output requirements, converting the dataset is also part of the process that a data scientist goes through prior to initiating training. In this particular case, the Amazon SageMaker implementation of Linear Learner takes recordIO-wrapped protobuf, where the data we have today is a pickle-sized numpy array on disk.

Most of the conversion effort is handled by the Amazon SageMaker Python SDK, imported as `sagemaker` below.

XGBoost

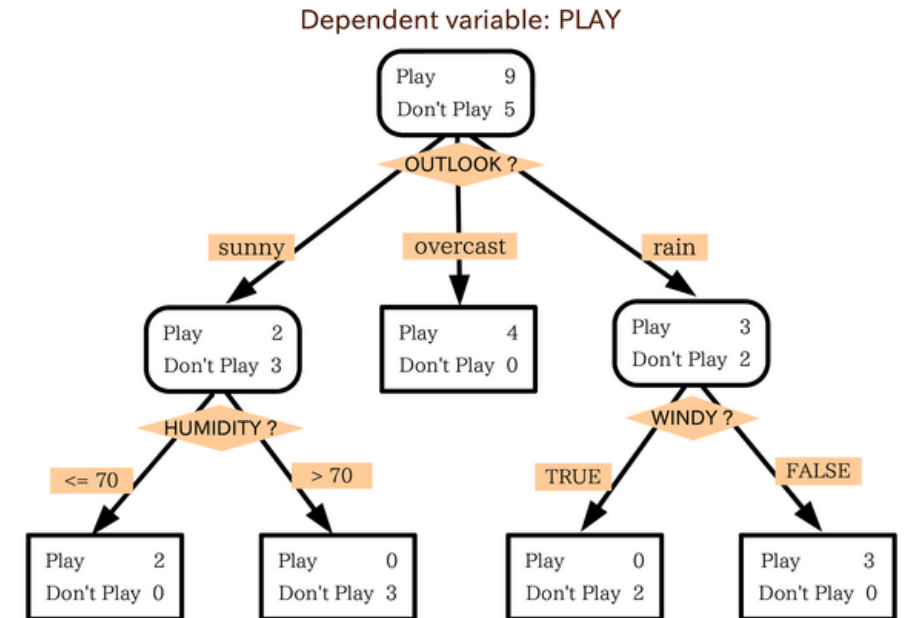
XGBoost: What's it for?

- eXtreme Gradient Boosting
 - Boosted group of decision trees
 - New trees made to correct the errors of previous trees
 - Uses gradient descent to minimize loss as new trees are added
- It's been winning a lot of Kaggle competitions
 - And it's fast, too
- Can be used for classification
- And also for regression
 - Using regression trees



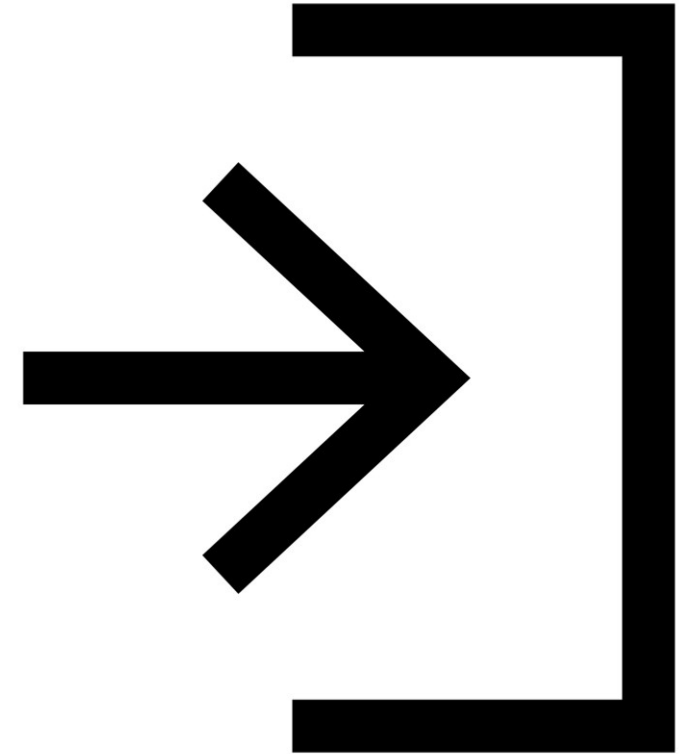
XGBoost: What's it for?

- eXtreme Gradient Boosting
 - Boosted group of decision trees
 - New trees made to correct the errors of previous trees
 - Uses gradient descent to minimize loss as new trees are added
- It's been winning a lot of Kaggle competitions
 - And it's fast, too
- Can be used for classification
- And also for regression
 - Using regression trees



XGBoost: What training input does it expect?

- XGBoost is weird, since it's not made for SageMaker. It's just open source XGBoost
- So, it takes CSV or libsvm input.
- AWS recently extended it to accept recordIO-protobuf and Parquet as well.



XGBoost: How is it used?

- Models are serialized/deserialized with Pickle
- Can use as a framework within notebooks
 - Sagemaker.xgboost
- Or as a built-in SageMaker algorithm



XGBoost: Important Hyperparameters

- There are a lot of them. A few:
- Subsample
 - Prevents overfitting
- Eta
 - Step size shrinkage, prevents overfitting
- Gamma
 - Minimum loss reduction to create a partition; larger = more conservative
- Alpha
 - L1 regularization term; larger = more conservative
- Lambda
 - L2 regularization term; larger = more conservative



XGBoost: Important Hyperparameters

- **eval_metric**
 - Optimize on AUC, error, rmse...
 - For example, if you care about false positives more than accuracy, you might use AUC here
- **scale_pos_weight**
 - Adjusts balance of positive and negative weights
 - Helpful for unbalanced classes
 - Might set to $\frac{\text{sum(negative cases)}}{\text{sum(positive cases)}}$
- **max_depth**
 - Max depth of the tree
 - Too high and you may overfit



XGBoost: Instance Types

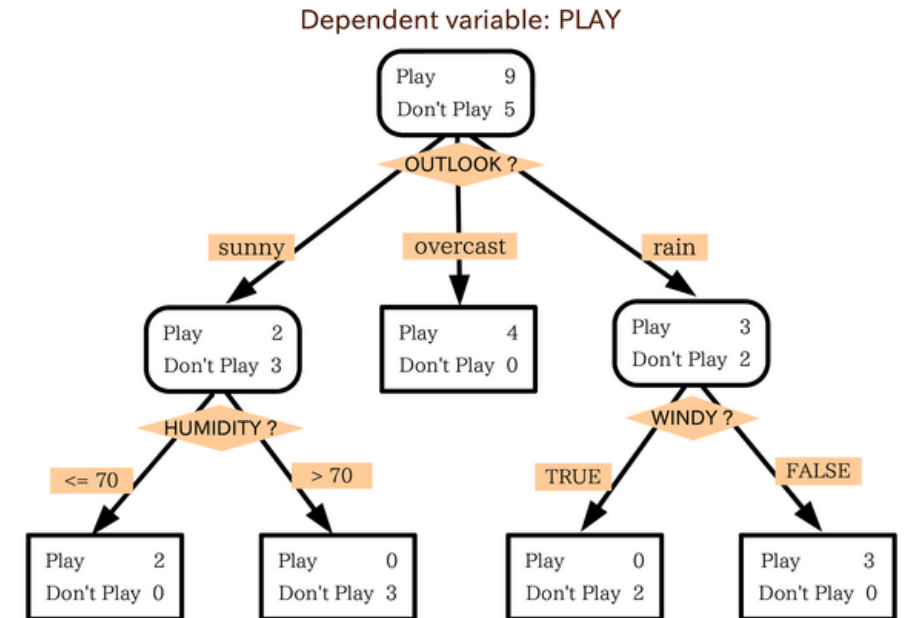
- Is memory-bound, not compute-bound
- So, **M5** is a good choice
- As of XGBoost 1.2, single-instance GPU training is available
 - For **example P2, P3**
 - Must set `tree_method` hyperparameter to `gpu_hist`
 - Trains more quickly and can be more cost effective.
- XGBoost 1.2-2
 - P2, P3, G4dn, G5
- XGBoost 1.5+: Distributed GPU training now available
 - Must set `use_dask_gpu_training` to `true`
 - Set `distribution` to `fully_replicated` in `TrainingInput`
 - Only works with `csv` or `parquet` input



LightGBM

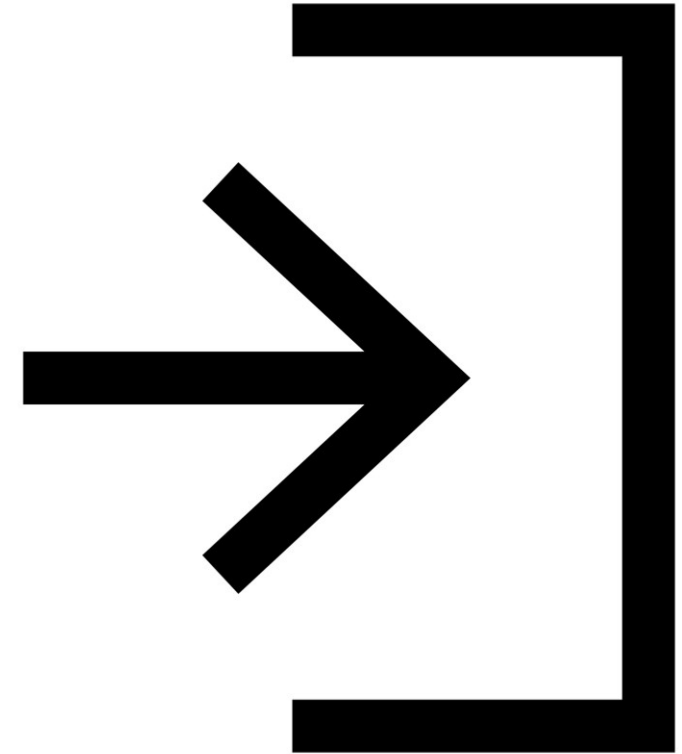
LightGBM: What's it for?

- Gradient Boosting Decision Tree
 - Kinda like XGBoost
 - Even more like CatBoost
 - Extended with Gradient-based One-Side Sampling and Exclusive Feature Bundling
- Predicts target variables with an ensemble of estimates from simpler models
- Classification, regression, or ranking



LightGBM: What training input does it expect?

- Requires txt/csv for training and for inference.
- Training and optional validation channels may be provided



LightGBM: Important Hyperparameters

- There are a lot of them. A few:
- Learning_rate
- Num_leaves
 - Max leaves per tree
- Feature_fraction
 - Subset of features per tree
- Bagging_fraction
 - Similar, but randomly sampled
- Bagging_freq
 - How often bagging is done
- Max_depth
- Min_data_in_leaf
 - Minimum amount of data in one leaf; can address overfitting



LightGBM: Instance Types

- Single or multi-instance CPU training
 - Define instances with `instance_count`
- Memory-bound algorithm
 - Choose general purpose over compute-optimized
 - M5, not C5
 - Be sure you have enough memory



Seq2Seq

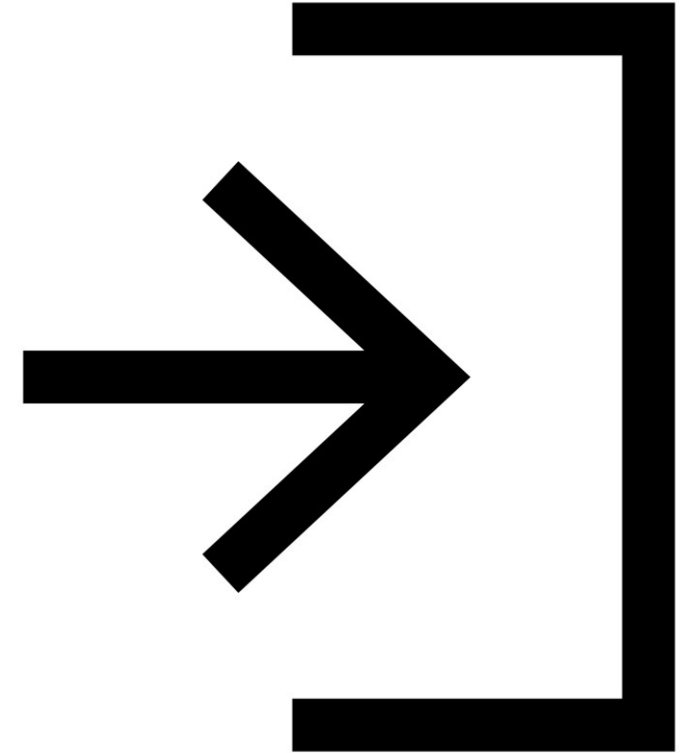
Seq2Seq: What's it for?

- Input is a sequence of tokens, output is a sequence of tokens
- Machine Translation
- Text summarization
- Speech to text
- Implemented with RNN's and CNN's with attention



Seq2Seq: What training input does it expect?

- RecordIO-Protobuf
 - Tokens must be integers (this is unusual, since most algorithms want floating point data.)
- Start with tokenized text files
- Convert to protobuf using sample code
 - Packs into integer tensors with vocabulary files
 - A lot like the TF/IDF lab we did earlier.
- Must provide training data, validation data, and vocabulary files.



Seq2Seq: How is it used?

- Training for machine translation can take days, even on SageMaker
- Pre-trained models are available
 - See the example notebook
- Public training datasets are available for specific translation tasks



Seq2Seq: Important Hyperparameters

- Batch_size
- Optimizer_type (adam, sgd, rmsprop)
- Learning_rate
- Num_layers_encoder
- Num_layers_decoder
- Can optimize on:
 - Accuracy
 - Vs. provided validation dataset
 - BLEU score
 - Compares against multiple reference translations
 - Perplexity
 - Cross-entropy



Seq2Seq: Instance Types

- Can only use GPU instance types (P3 for example)
- Can only use a single machine for training
 - But can use multi-GPU's on one machine



DeepAR

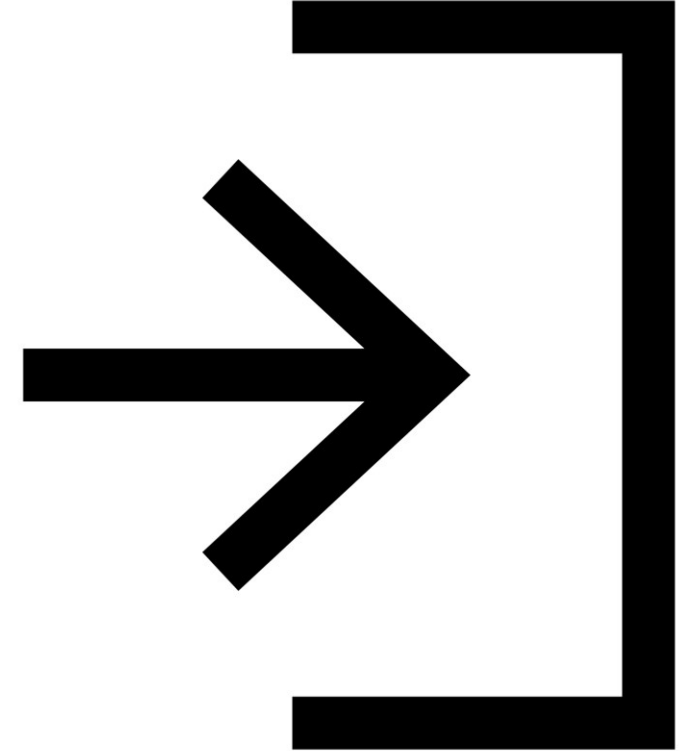
DeepAR: What's it for?

- Forecasting one-dimensional time series data
- Uses RNN's
- Allows you to train the same model over several related time series
- Finds frequencies and seasonality



DeepAR: What training input does it expect?

- JSON lines format
 - Gzip or Parquet
- Each record must contain:
 - Start: the starting time stamp
 - Target: the time series values
- Each record can contain:
 - Dynamic_feat: dynamic features (such as, was a promotion applied to a product in a time series of product purchases)
 - Cat: categorical features



```
{"start": "2009-11-01 00:00:00", "target": [4.3, "NaN", 5.1, ...], "cat": [0, 1], "dynamic_feat": [[1.1, 1.2, 0.5, ...]]}  
{"start": "2012-01-30 00:00:00", "target": [1.0, -5.0, ...], "cat": [2, 3], "dynamic_feat": [[1.1, 2.05, ...]]}  
{"start": "1999-01-30 00:00:00", "target": [2.0, 1.0], "cat": [1, 4], "dynamic_feat": [[1.3, 0.4]]}
```

DeepAR: How is it used?

- Always include entire time series for training, testing, and inference
- Use entire dataset as training set, remove last time points for testing. Evaluate on withheld values.
- Don't use very large values for prediction length (> 400)
- Train on many time series and not just one when possible



DeepAR: Important Hyperparameters

- Context_length
 - Number of time points the model sees before making a prediction
 - Can be smaller than seasonalities; the model will lag one year anyhow.
- Epochs
- mini_batch_size
- Learning_rate
- Num_cells



DeepAR: Instance Types

- Can use CPU or GPU
- Single or multi machine
- Start with CPU (ml.c4.2xlarge, ml.c4.4xlarge)
- Move up to GPU if necessary
 - Only helps with larger models
 - Or with large mini-batch sizes (>512)
- CPU-only for inference
- May need larger instances for tuning



BlazingText

BlazingText: What's it for?

- Text classification
 - Predict labels for a sentence
 - Useful in web searches, information retrieval
 - Supervised
- Word2vec
 - Creates a vector representation of words
 - Semantically similar words are represented by vectors close to each other
 - This is called a *word embedding*
 - It is useful for NLP, but is not an NLP algorithm in itself!
 - Used in machine translation, sentiment analysis
 - Remember it only works on individual words, not sentences or documents



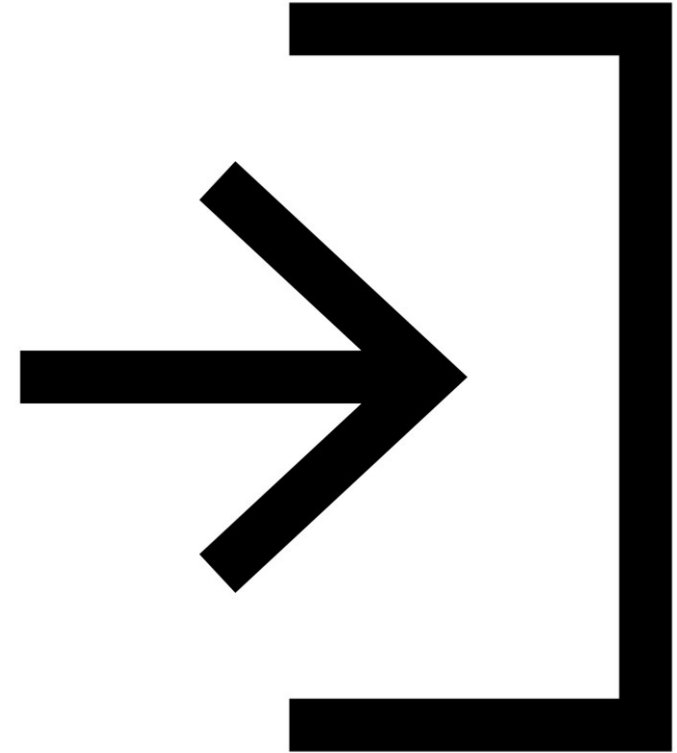
BlazingText: What training input does it expect?

- For supervised mode (text classification):
 - One sentence per line
 - First “word” in the sentence is the string `__label__` followed by the label
- Also, “augmented manifest text format”
- Word2vec just wants a text file with one training sentence per line.

`__label__4` linux ready for prime time , intel says , despite all the linux hype , the open-source movement has yet to make a huge splash in the desktop market . that may be about to change , thanks to chipmaking giant intel corp .

`__label__2` bowled by the slower one again , kolkata , november 14 the past caught up with sourav ganguly as the indian skippers return to international cricket was short lived .

```
{"source":"linux ready for prime time , intel says , despite all the linux hype", "label":1}  
{"source":"bowled by the slower one again , kolkata , november 14 the past caught up with  
sourav ganguly", "label":2}
```



BlazingText: How is it used?

- Word2vec has multiple modes
 - Cbow (Continuous Bag of Words)
 - Skip-gram
 - Batch skip-gram
 - Distributed computation over many CPU nodes



BlazingText: Important Hyperparameters

- Word2vec:
 - Mode (batch_skipgram, skipgram, cbow)
 - Learning_rate
 - Window_size
 - Vector_dim
 - Negative_samples
- Text classification:
 - Epochs
 - Learning_rate
 - Word_ngrams
 - Vector_dim



BlazingText: Instance Types

- For cbow and skipgram, recommend a single ml.p3.2xlarge
 - Any single CPU or single GPU instance will work
- For batch_skipgram, can use single or multiple CPU instances
- For text classification, C5 recommended if less than 2GB training data. For larger data sets, use a single GPU instance (ml.p2.xlarge or ml.p3.2xlarge)



Object2Vec

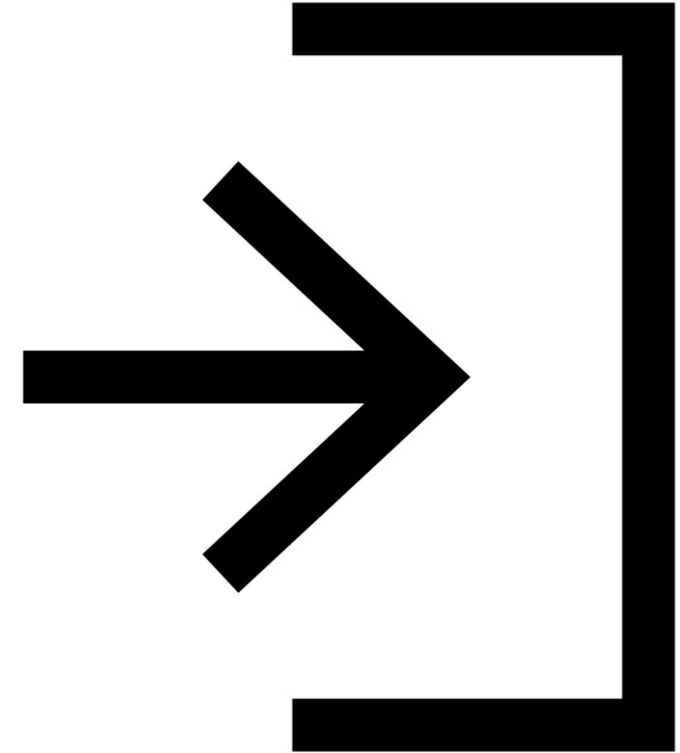
Object2Vec: What's it for?

- Remember word2vec from Blazing Text? It's like that, but arbitrary objects
- It creates low-dimensional dense embeddings of high-dimensional objects
- It is basically word2vec, generalized to handle things other than words.
- Compute nearest neighbors of objects
- Visualize clusters
- Genre prediction
- Recommendations (similar items or users)



Object2Vec: What training input does it expect?

- Data must be tokenized into integers
- Training data consists of pairs of tokens and/or sequences of tokens
 - Sentence – sentence
 - Labels-sequence (genre to description?)
 - Customer-customer
 - Product-product
 - User-item



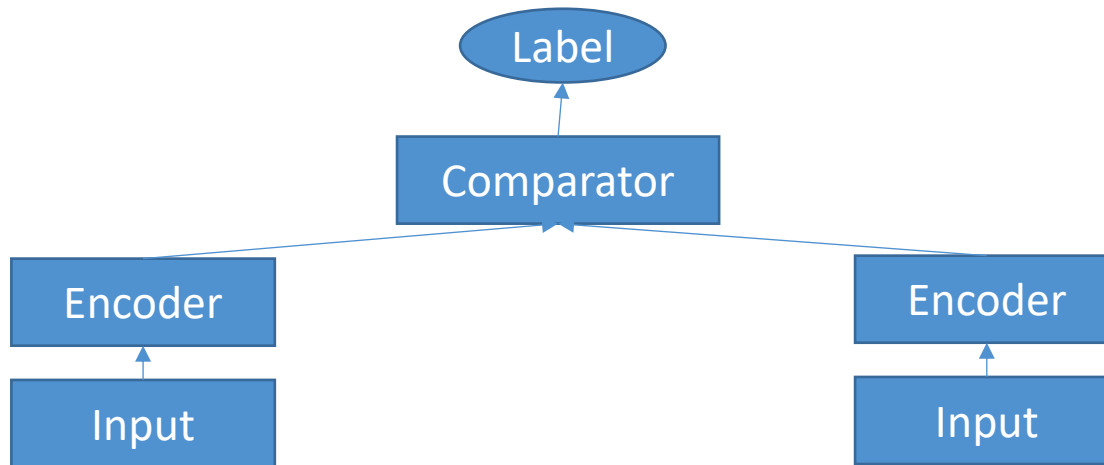
```
{"label": 0, "in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15, 69, 821, 4], "in1": [16, 21, 13, 45, 14, 9, 80, 59, 164, 4]}
```

```
{"label": 1, "in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9, 107, 4], "in1": [22, 32, 13, 25, 1016, 573, 3252, 4]}
```

```
{"label": 1, "in0": [774, 14, 21, 206], "in1": [21, 366, 125]}
```


Object2Vec: How is it used?

- Process data into JSON Lines and shuffle it
- Train with two input channels, two encoders, and a comparator
- Encoder choices:
 - Average-pooled embeddings
 - CNN's
 - Bidirectional LSTM
- Comparator is followed by a feed-forward neural network



Object2Vec: Important Hyperparameters

- The usual deep learning ones...
 - Dropout, early stopping, epochs, learning rate, batch size, layers, activation function, optimizer, weight decay
- Enc1_network, enc2_network
 - Choose hcnn, bilstm, pooled_embedding



Object2Vec: Instance Types

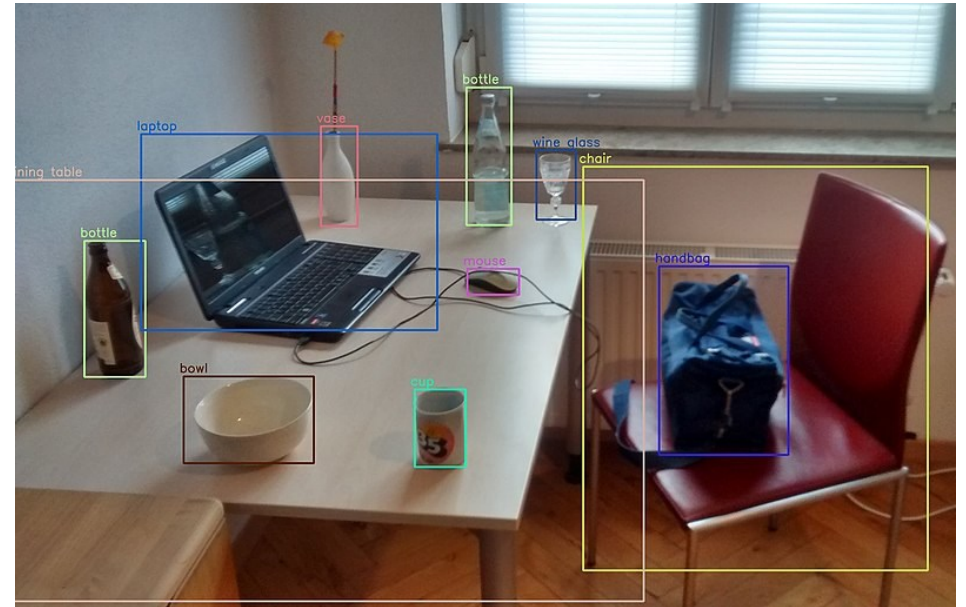
- Can only train on a single machine (CPU or GPU, multi-GPU OK)
 - ml.m5.2xlarge
 - ml.p2.xlarge
 - If needed, go up to ml.m5.4xlarge or ml.m5.12xlarge
 - GPU options: P2, P3, G4dn, G5
- Inference: use ml.p3.2xlarge
 - Use `INFERENCE_PREFERRED_MODE` environment variable to optimize for encoder embeddings rather than classification or regression.



Object Detection

Object Detection: What's it for?

- Identify all objects in an image with bounding boxes
- Detects and classifies objects with a single deep neural network
- Classes are accompanied by confidence scores
- Can train from scratch, or use pre-trained models based on ImageNet



(MTheiler) [CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>)]

Object Detection: How is it used?

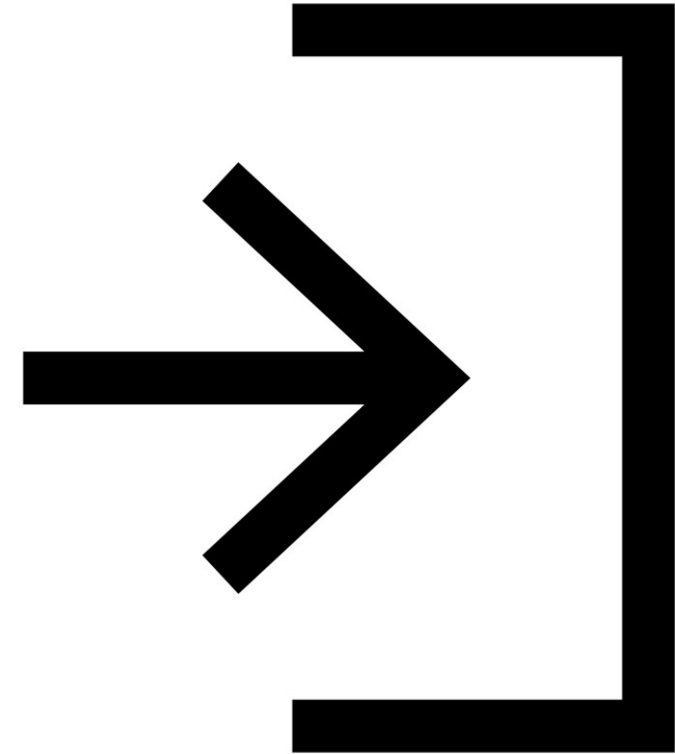
- Two variants: MXNet and Tensorflow
- Takes an image as input, outputs all instances of objects in the image with categories and confidence scores
- MXNet
 - Uses a CNN with the Single Shot multibox Detector (SSD) algorithm
 - The base CNN can be VGG-16 or ResNet-50
 - Transfer learning mode / incremental training
 - Use a pre-trained model for the base network weights instead of random initial weights
 - Uses flip, rescale, and jitter internally to avoid overfitting
- Tensorflow
 - Uses ResNet, EfficientNet, MobileNet models from the TensorFlow Model Garden



Object Detection: What training input does it expect?

- MXNet: RecordIO or image format (jpg or png)
- With image format, supply a JSON file for annotation data for each image

```
"file": "your_image_directory/sample_image1.jpg",  
"image_size": [  
  {  
    "width": 500,  
    "height": 400,  
    "depth": 3  
  }  
],  
"annotations": [  
  {  
    "class_id": 0,  
    "left": 111,  
    "top": 134,  
    "width": 61,  
    "height": 128  
  },  
],  
"categories": [  
  {  
    "class_id": 0,  
    "name": "dog"  
  },  
],  
}
```



Object Detection: Important Hyperparameters

- Mini_batch_size
- Learning_rate
- Optimizer
 - Sgd, adam, rmsprop, adadelata



Object Detection: Instance Types

- Use GPU instances for training (multi-GPU and multi-machine OK)
 - ml.p2.xlarge, ml.p2.16xlarge, ml.p3.2xlarge, ml.p3.16xlarge, G4dn, G5
- Use CPU or GPU for inference
 - M5, P2, P3, G4dn all OK



Image Classification

Image Classification: What's it for?

- Assign one or more labels to an image
- Doesn't tell you where objects are, just what objects are in the image



Image Classification: How is it used?

- Separate algorithms for MXNet and Tensorflow
- MXNet:
 - Full training mode
 - Network initialized with random weights
 - Transfer learning mode
 - Initialized with pre-trained weights
 - The top fully-connected layer is initialized with random weights
 - Network is fine-tuned with new training data
 - Default image size is 3-channel 224x224 (ImageNet's dataset)
- Tensorflow: Uses various Tensorflow Hub models (MobileNet, Inception, ResNet, EfficientNet)
 - Top classification layer is available for fine tuning or further training



Image Classification: Important Hyperparameters

- The usual suspects for deep learning
 - Batch size, learning rate, optimizer
- Optimizer-specific parameters
 - Weight decay, beta 1, beta 2, eps, gamma
 - Slightly different between MXNet and Tensorflow versions



Image Classification: Instance Types

- GPU instances for training (ml.p2, p3, g4dn, g5) Multi-GPU and multi-machine OK.
- CPU or GPU for inference (m5, p2, p3, g4dn, g5)



Semantic Segmentation

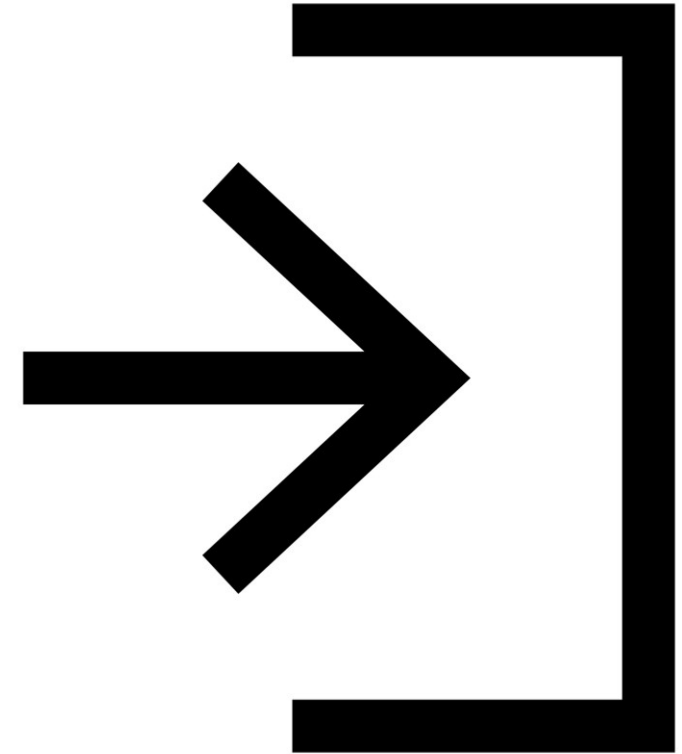
Semantic Segmentation: What's it for?

- Pixel-level object classification
- Different from image classification – that assigns labels to whole images
- Different from object detection – that assigns labels to bounding boxes
- Useful for self-driving vehicles, medical imaging diagnostics, robot sensing
- Produces a *segmentation mask*



Semantic Segmentation: What training input does it expect?

- JPG Images and PNG annotations
- For both training and validation
- Label maps to describe annotations
- Augmented manifest image format supported for Pipe mode.
- JPG images accepted for inference



Semantic Segmentation: How is it used?

- Built on MXNet Gluon and Gluon CV
- Choice of 3 algorithms:
 - Fully-Convolutional Network (FCN)
 - Pyramid Scene Parsing (PSP)
 - DeepLabV3
- Choice of backbones:
 - ResNet50
 - ResNet101
 - Both trained on ImageNet
- Incremental training, or training from scratch, supported too



Semantic Segmentation: Important Hyperparameters

- Epochs, learning rate, batch size, optimizer, etc
- Algorithm
- Backbone



Image Classification: Instance Types

- GPU instances for training (ml.p2, p3, g4dn, g5) Multi-GPU and multi-machine OK.
- CPU or GPU for inference (m5, p2, p3, g4dn, g5)



Random Cut Forest

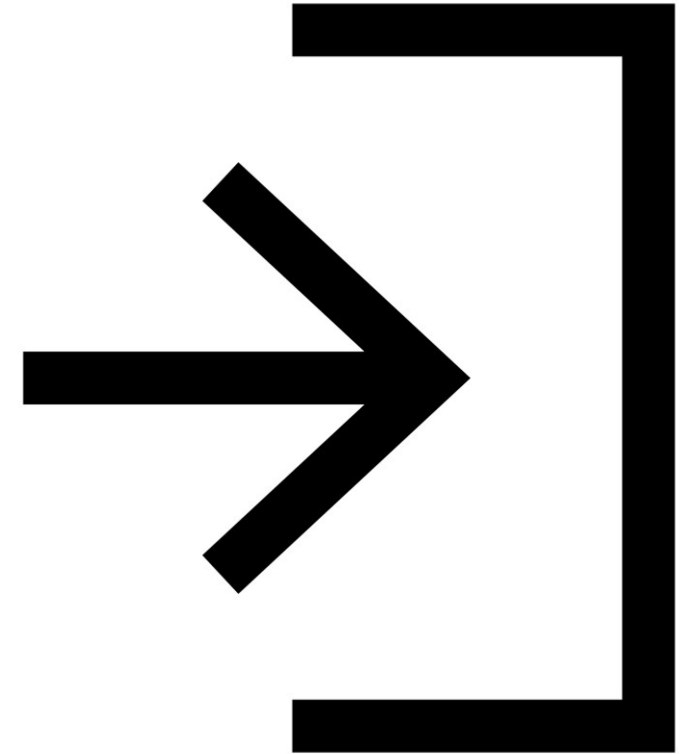
Random Cut Forest: What's it for?

- Anomaly detection
- Unsupervised
- Detect unexpected spikes in time series data
- Breaks in periodicity
- Unclassifiable data points
- Assigns an anomaly score to each data point
- Based on an algorithm developed by Amazon that they seem to be very proud of!



Random Cut Forest: What training input does it expect?

- RecordIO-protobuf or CSV
- Can use File or Pipe mode on either
- Optional test channel for computing accuracy, precision, recall, and F1 on labeled data (anomaly or not)



Random Cut Forest: How is it used?

- Creates a forest of trees where each tree is a partition of the training data; looks at expected change in complexity of the tree as a result of adding a point into it
- Data is sampled randomly
- Then trained
- RCF shows up in Kinesis Analytics as well; it can work on streaming data too.



Random Cut Forest: Important Hyperparameters

- Num_trees
 - Increasing reduces noise
- Num_samples_per_tree
 - Should be chosen such that $1/\text{num_samples_per_tree}$ approximates the ratio of anomalous to normal data



Random Cut Forest: Instance Types

- Does not take advantage of GPUs
- Use M4, C4, or C5 for training
- ml.c5.xl for inference



Neural Topic Model

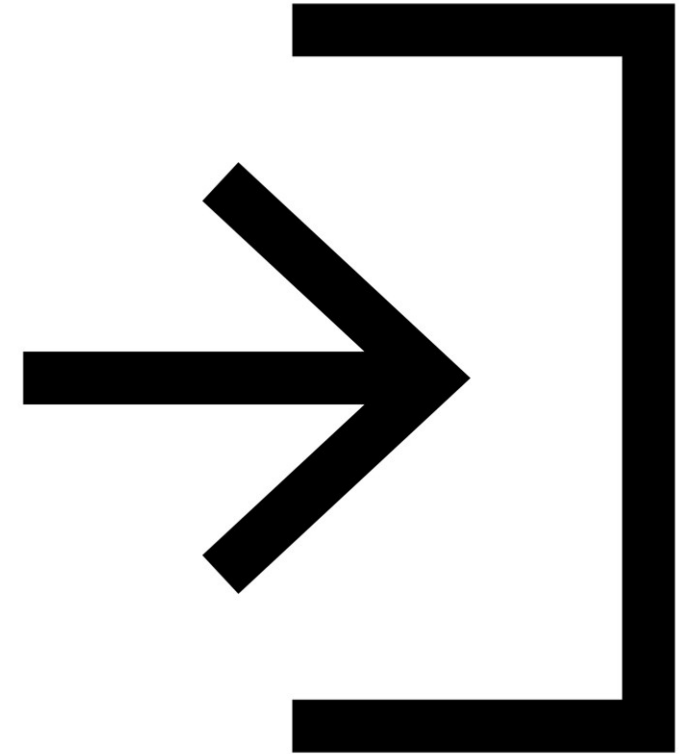
Neural Topic Model: What's it for?

- Organize documents into topics
- Classify or summarize documents based on topics
- It's not just TF/IDF
 - “bike”, “car”, “train”, “mileage”, and “speed” might classify a document as “transportation” for example (although it wouldn't know to call it that)
- Unsupervised
 - Algorithm is “Neural Variational Inference”



Neural Topic Model: What training input does it expect?

- Four data channels
 - “train” is required
 - “validation”, “test”, and “auxiliary” optional
- recordIO-protobuf or CSV
- Words must be tokenized into integers
 - Every document must contain a count for every word in the vocabulary in CSV
 - The “auxiliary” channel is for the vocabulary
- File or pipe mode



Neural Topic Model: How is it used?

- You define how many topics you want
- These topics are a latent representation based on top ranking words
- One of two topic modeling algorithms in SageMaker – you can try them both!



Neural Topic Model: Important Hyperparameters

- Lowering mini_batch_size and learning_rate can reduce validation loss
 - At expense of training time
- Num_topics



Neural Topic Model: Instance Types

- GPU or CPU
 - GPU recommended for training
 - CPU OK for inference
 - CPU is cheaper



LDA

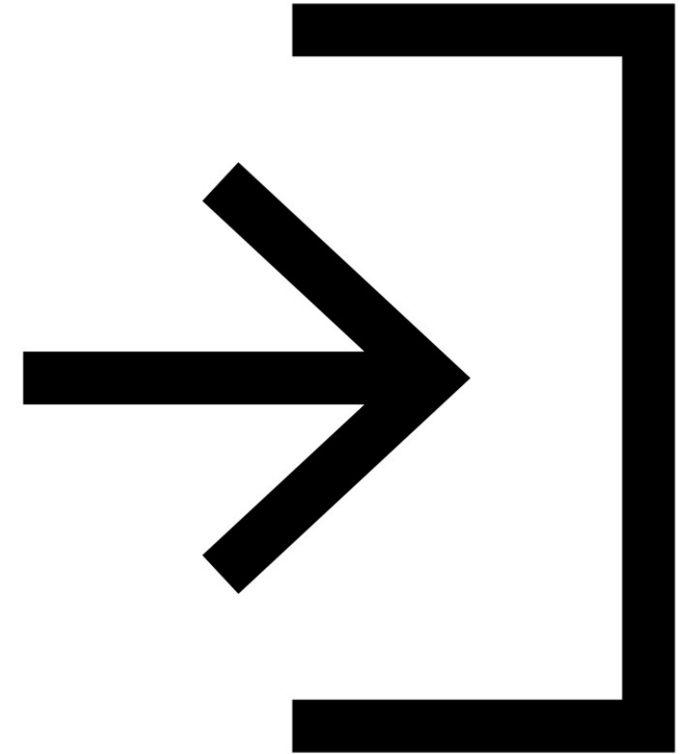
LDA: What's it for?

- Latent Dirichlet Allocation
- Another topic modeling algorithm
 - Not deep learning
- Unsupervised
 - The topics themselves are unlabeled; they are just groupings of documents with a shared subset of words
- Can be used for things other than words
 - Cluster customers based on purchases
 - Harmonic analysis in music



LDA: What training input does it expect?

- Train channel, optional test channel
- recordIO-protobuf or CSV
- Each document has counts for every word in vocabulary (in CSV format)
- Pipe mode only supported with recordIO



LDA: How is it used?

- Unsupervised; generates however many topics you specify
- Optional test channel can be used for scoring results
 - Per-word log likelihood
- Functionally similar to NTM, but CPU-based
 - Therefore maybe cheaper / more efficient



LDA: Important Hyperparameters

- Num_topics
- Alpha0
 - Initial guess for concentration parameter
 - Smaller values generate sparse topic mixtures
 - Larger values (>1.0) produce uniform mixtures



LDA: Instance Types

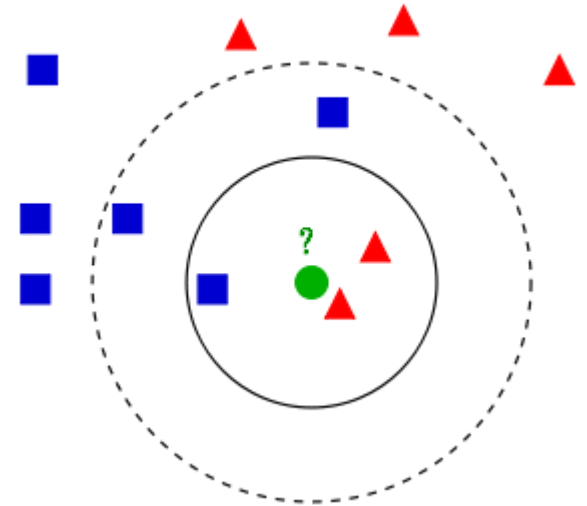
- Single-instance CPU training



KNN

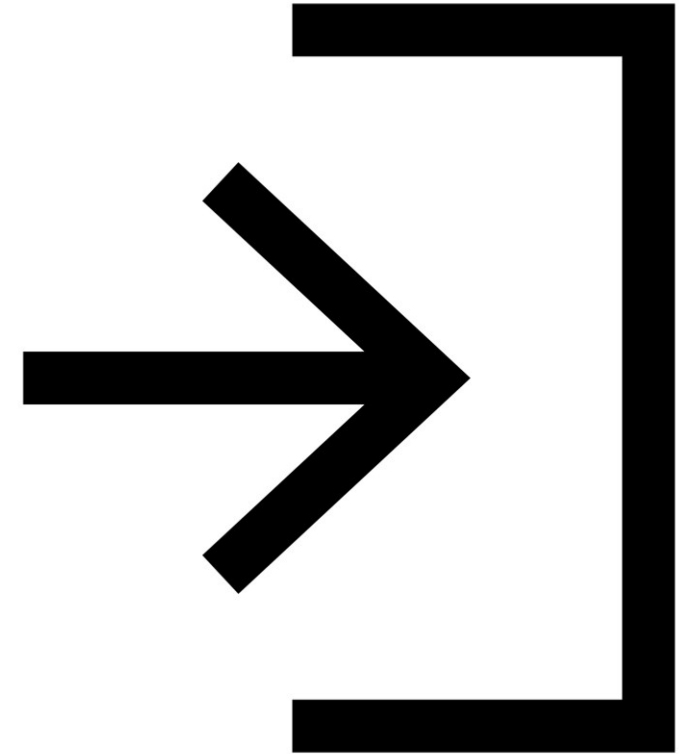
KNN: What's it for?

- K-Nearest-Neighbors
- Simple classification or regression algorithm
- Classification
 - Find the K closest points to a sample point and return the most frequent label
- Regression
 - Find the K closest points to a sample point and return the average value



KNN: What training input does it expect?

- Train channel contains your data
- Test channel emits accuracy or MSE
- recordIO-protobuf or CSV training
 - First column is label
- File or pipe mode on either



KNN: How is it used?

- Data is first sampled
- SageMaker includes a dimensionality reduction stage
 - Avoid sparse data (“curse of dimensionality”)
 - At cost of noise / accuracy
 - “sign” or “fjlt” methods
- Build an index for looking up neighbors
- Serialize the model
- Query the model for a given K



KNN: Important Hyperparameters

- K!
- Sample_size



KNN: Instance Types

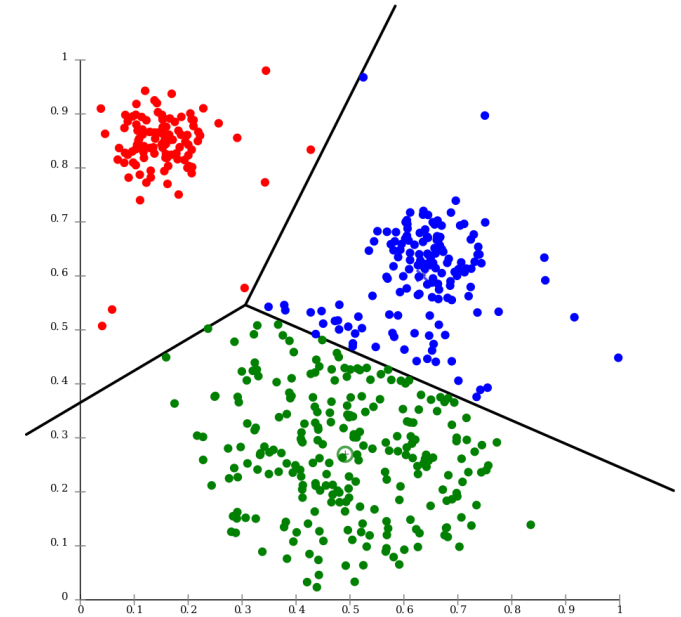
- Training on CPU or GPU
 - Ml.m5.2xlarge
 - Ml.p2.xlarge
- Inference
 - CPU for lower latency
 - GPU for higher throughput on large batches



K-Means

K-Means: What's it for?

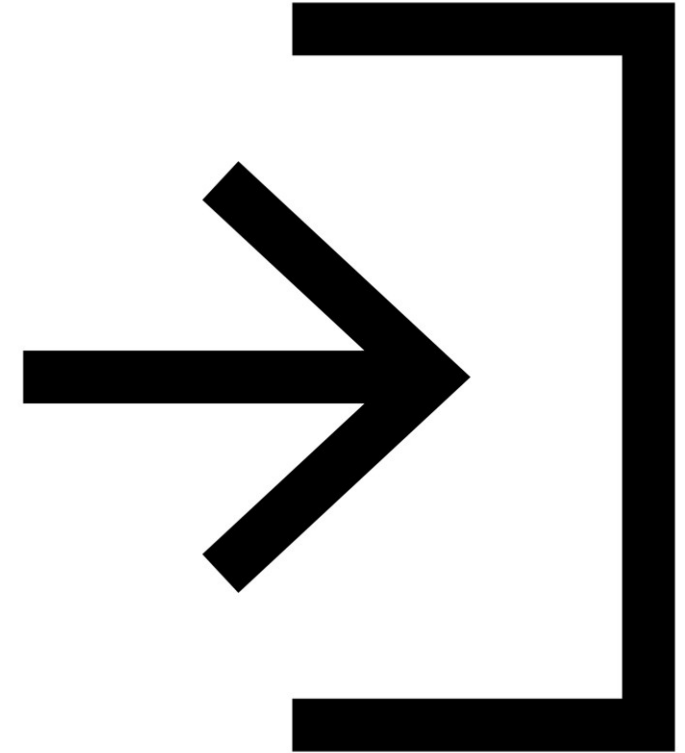
- Unsupervised clustering
- Divide data into K groups, where members of a group are as similar as possible to each other
 - You define what “similar” means
 - Measured by Euclidean distance
- Web-scale K-Means clustering



Chire [CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/>)]

K-Means: What training input does it expect?

- Train channel, optional test
 - Train ShardedByS3Key, test FullyReplicate
- recordIO-protobuf or CSV
- File or Pipe on either



K-Means: How is it used?

- Every observation mapped to n-dimensional space (n = number of features)
- Works to optimize the center of K clusters
 - “extra cluster centers” may be specified to improve accuracy (which end up getting reduced to k)
 - $K = k \times x$
- Algorithm:
 - Determine initial cluster centers
 - Random or k-means++ approach
 - K-means++ tries to make initial clusters far apart
 - Iterate over training data and calculate cluster centers
 - Reduce clusters from K to k
 - Using Lloyd’s method with kmeans++



K-Means: Important Hyperparameters

- K!
 - Choosing K is tricky
 - Plot within-cluster sum of squares as function of K
 - Use “elbow method”
 - Basically optimize for tightness of clusters
- Mini_batch_size
- Extra_center_factor
- Init_method



K-Means: Instance Types

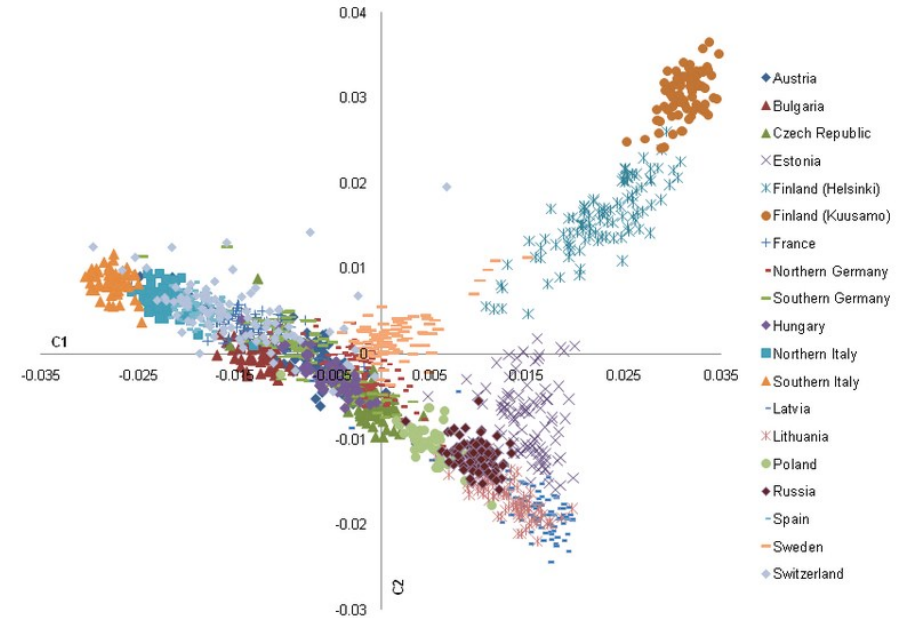
- CPU or GPU, but CPU recommended
 - Only one GPU per instance used on GPU
 - So use ml.g4dn.xlarge if you're going to use GPU
 - p2, p3, g4dn, and g4 supported



PCA

PCA: What's it for?

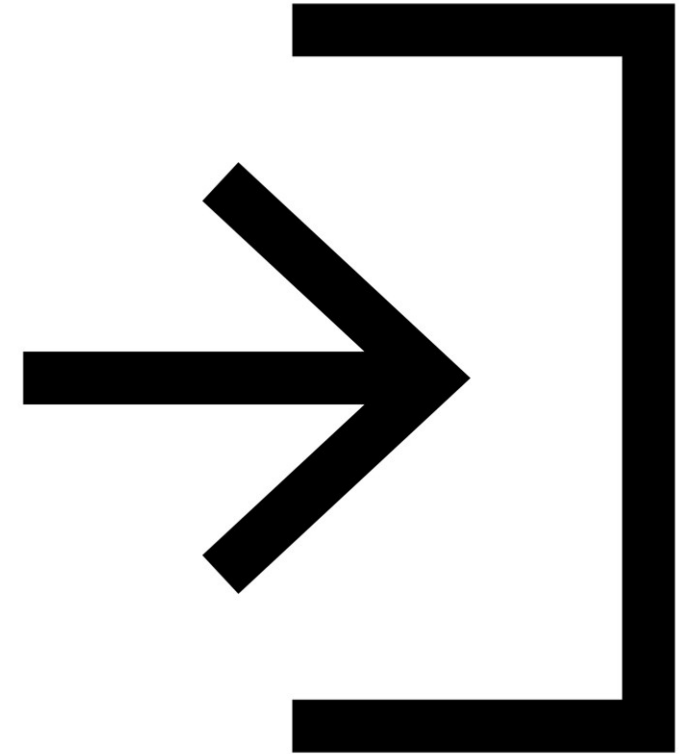
- Principal Component Analysis
- Dimensionality reduction
 - Project higher-dimensional data (lots of features) into lower-dimensional (like a 2D plot) while minimizing loss of information
 - The reduced dimensions are called components
 - First component has largest possible variability
 - Second component has the next largest...
- Unsupervised



Nelis M, Esko T, Mañgi R, Zimprich F, Zimprich A, et al. (2009) [CC BY 2.5 (<https://creativecommons.org/licenses/by/2.5>)]

PCA: What training input does it expect?

- recordIO-protobuf or CSV
- File or Pipe on either



PCA: How is it used?

- Covariance matrix is created, then singular value decomposition (SVD)
- Two modes
 - Regular
 - For sparse data and moderate number of observations and features
 - Randomized
 - For large number of observations and features
 - Uses approximation algorithm



PCA: Important Hyperparameters

- Algorithm_mode
- Subtract_mean
 - Unbias data



PCA: Instance Types

- GPU or CPU
 - It depends “on the specifics of the input data”



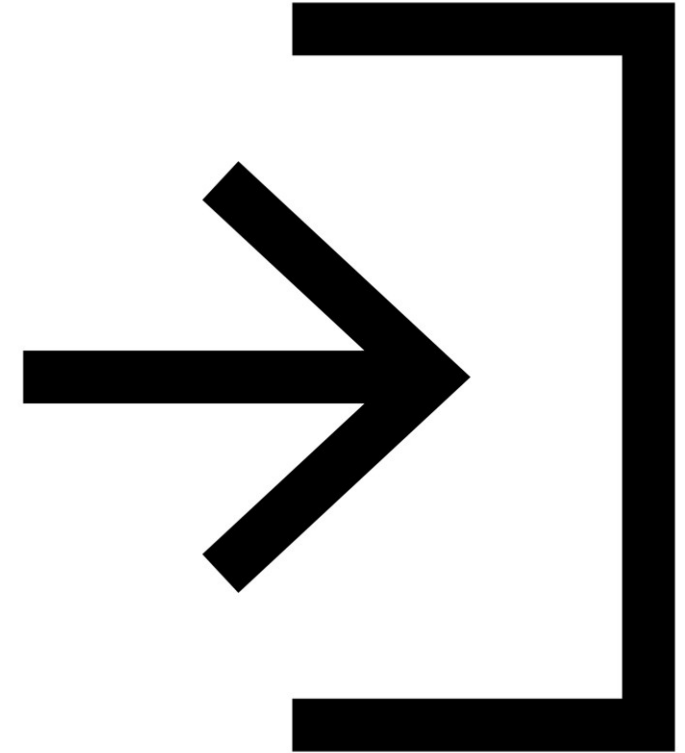
Factorization Machines

Factorization Machines: What's it for?

- Dealing with sparse data
 - Click prediction
 - Item recommendations
 - Since an individual user doesn't interact with most pages / products the data is sparse
- Supervised
 - Classification or regression
- Limited to pair-wise interactions
 - User -> item for example

Factorization Machines: What training input does it expect?

- recordIO-protobuf with Float32
 - Sparse data means CSV isn't practical



Factorization Machines: How is it used?

- Finds factors we can use to predict a classification (click or not? Purchase or not?) or value (predicted rating?) given a matrix representing some pair of things (users & items?)
- Usually used in the context of recommender systems



Factorization Machines: Important Hyperparameters

- Initialization methods for bias, factors, and linear terms
 - Uniform, normal, or constant
 - Can tune properties of each method



Factorization Machines: Instance Types

- CPU or GPU
 - CPU recommended
 - GPU only works with dense data



IP Insights

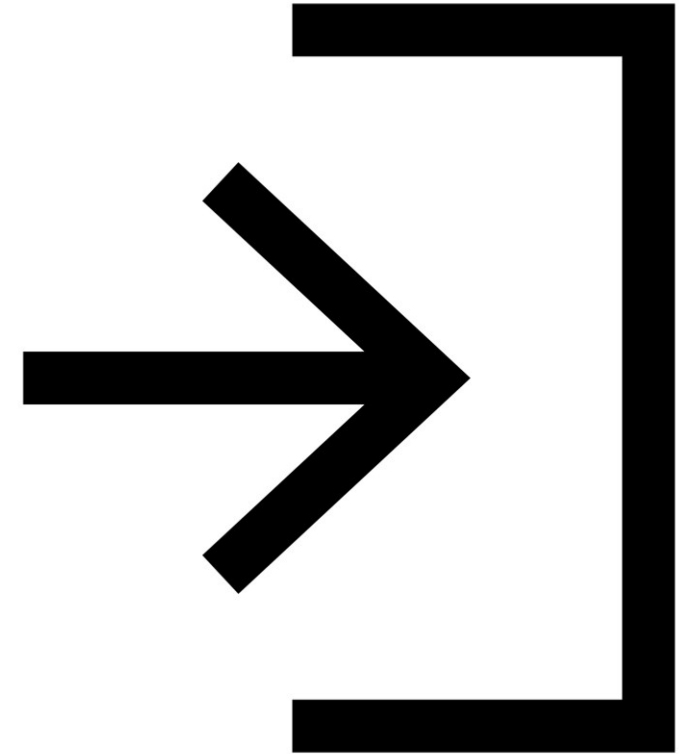
IP Insights: What's it for?

- Unsupervised learning of IP address usage patterns
- Identifies suspicious behavior from IP addresses
 - Identify logins from anomalous IP's
 - Identify accounts creating resources from anomalous IP's



IP Insights: What training input does it expect?

- User names, account ID's can be fed in directly; no need to pre-process
- Training channel, optional validation (computes AUC score)
- CSV only
 - Entity, IP



IP Insights: How is it used?

- Uses a neural network to learn latent vector representations of entities and IP addresses.
- Entities are hashed and embedded
 - Need sufficiently large hash size
- Automatically generates negative samples during training by randomly pairing entities and IP's



IP Insights: Important Hyperparameters

- Num_entity_vectors
 - Hash size
 - Set to twice the number of unique entity identifiers
- Vector_dim
 - Size of embedding vectors
 - Scales model size
 - Too large results in overfitting
- Epochs, learning rate, batch size, etc.



IP Insights: Instance Types

- CPU or GPU
 - GPU recommended
 - Ml.p3.2xlarge or higher
 - Can use multiple GPU's
 - Size of CPU instance depends on `vector_dim` and `num_entity_vectors`



Deep Learning 101

And AWS Best Practices

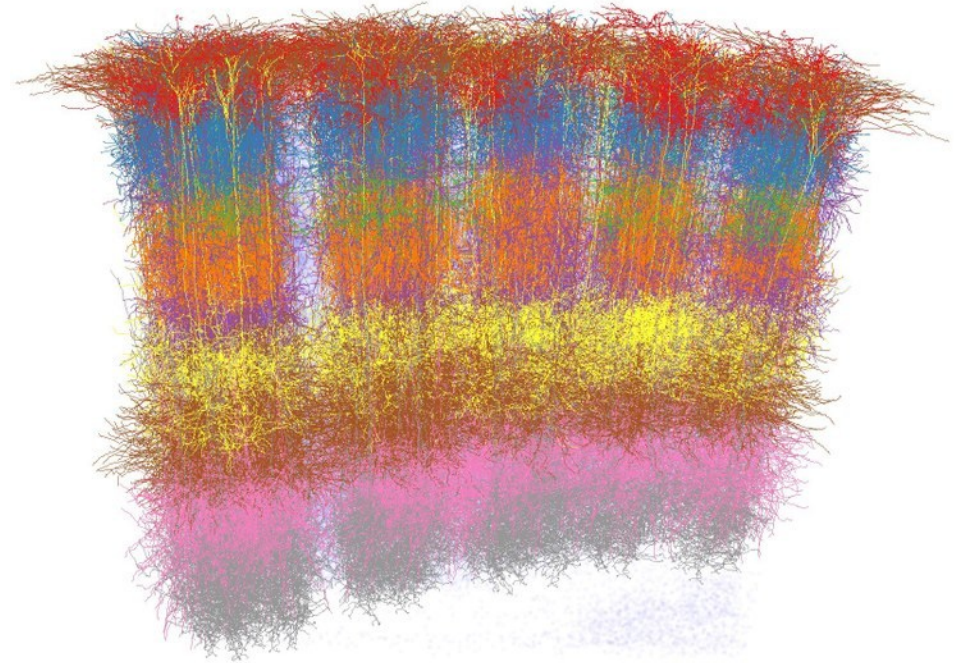
The biological inspiration

- Neurons in your cerebral cortex are connected via axons
- A neuron “fires” to the neurons it’s connected to, when enough of its input signals are activated.
- Very simple at the individual neuron level – but layers of neurons connected in this way can yield learning behavior.
- Billions of neurons, each with thousands of connections, yields a mind



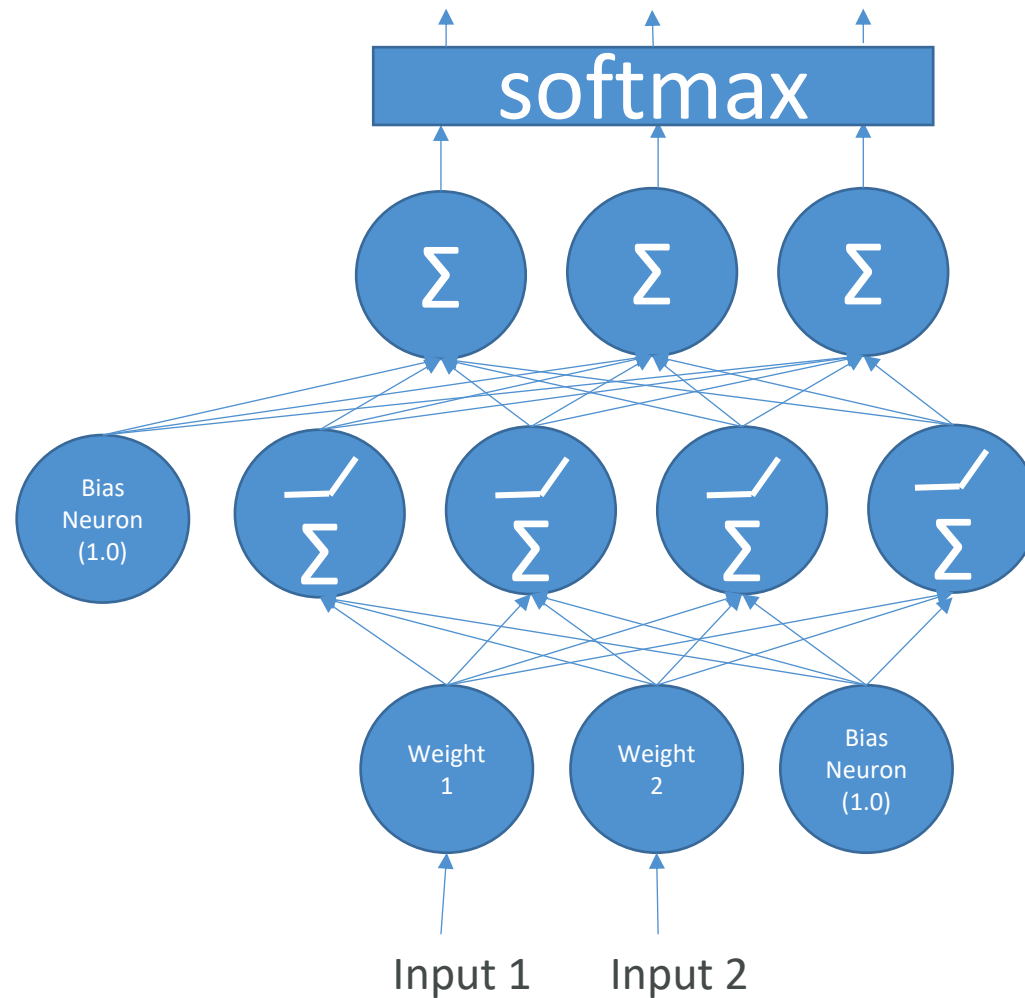
Cortical columns

- Neurons in your cortex seem to be arranged into many stacks, or “columns” that process information in parallel
- “mini-columns” of around 100 neurons are organized into larger “hyper-columns”. There are 100 million mini-columns in your cortex
- This is coincidentally similar to how GPU’s work...



(credit: Marcel Oberlaender et al.)

Deep Neural Networks



Deep Learning Frameworks

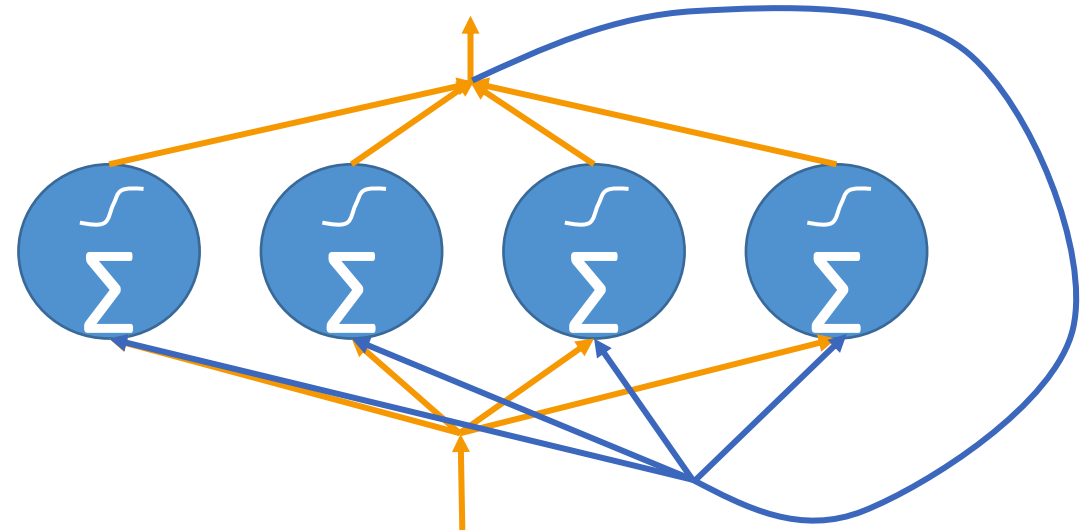
- Tensorflow / Keras
- MXNet

```
model = Sequential()

model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9,
          nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd, metrics=['accuracy'])
```

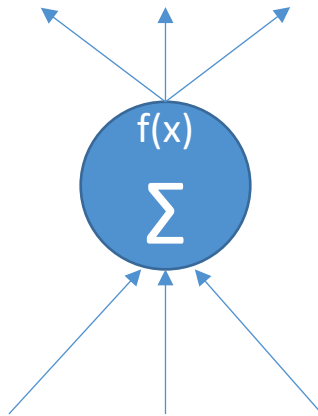
Types of Neural Networks

- Feedforward Neural Network
- Convolutional Neural Networks (CNN)
 - Image classification (is there a stop sign in this image?)
- Recurrent Neural Networks (RNNs)
 - Deals with sequences in time (predict stock prices, understand words in a sentence, translation, etc)
 - **LSTM**, GRU



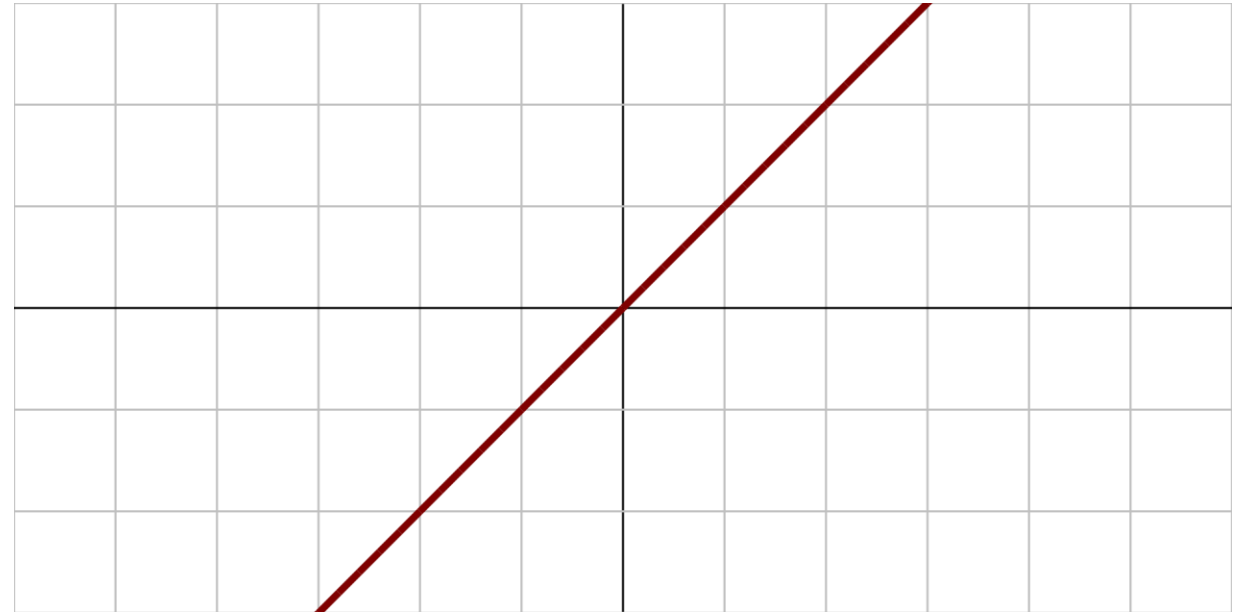
Activation Functions

- Define the output of a node / neuron given its input signals



Linear activation function

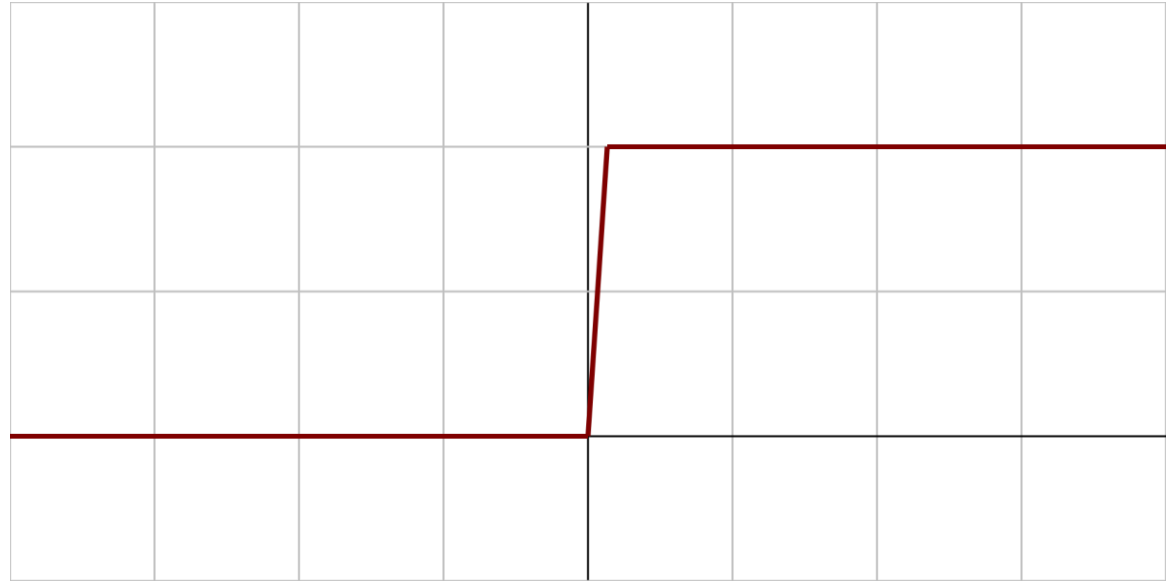
- It doesn't really *do* anything
- Can't do backpropagation



By Laughsinthestocks - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=44920411>

Binary step function

- It's on or off
- Can't handle multiple classification – it's binary after all
- Vertical slopes don't work well with calculus!



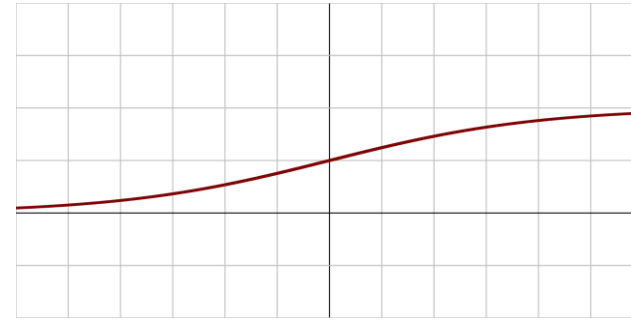
By Laughsinthestocks - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=44920435>

Instead we need non-linear activation functions

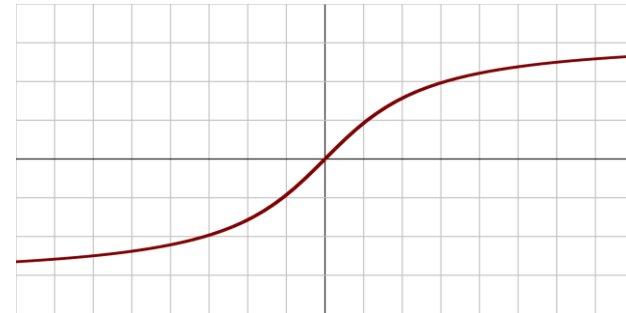
- These can create complex mappings between inputs and outputs
- Allow backpropagation (because they have a useful derivative)
- Allow for multiple layers (linear functions degenerate to a single layer)

Sigmoid / Logistic / TanH

- Nice & smooth
- Scales everything from 0-1 (Sigmoid / Logistic) or -1 to 1 (tanh / hyperbolic tangent)
- But: changes slowly for high or low values
 - The “Vanishing Gradient” problem
- Computationally expensive
- Tanh generally preferred over sigmoid



Sigmoid AKA Logistic

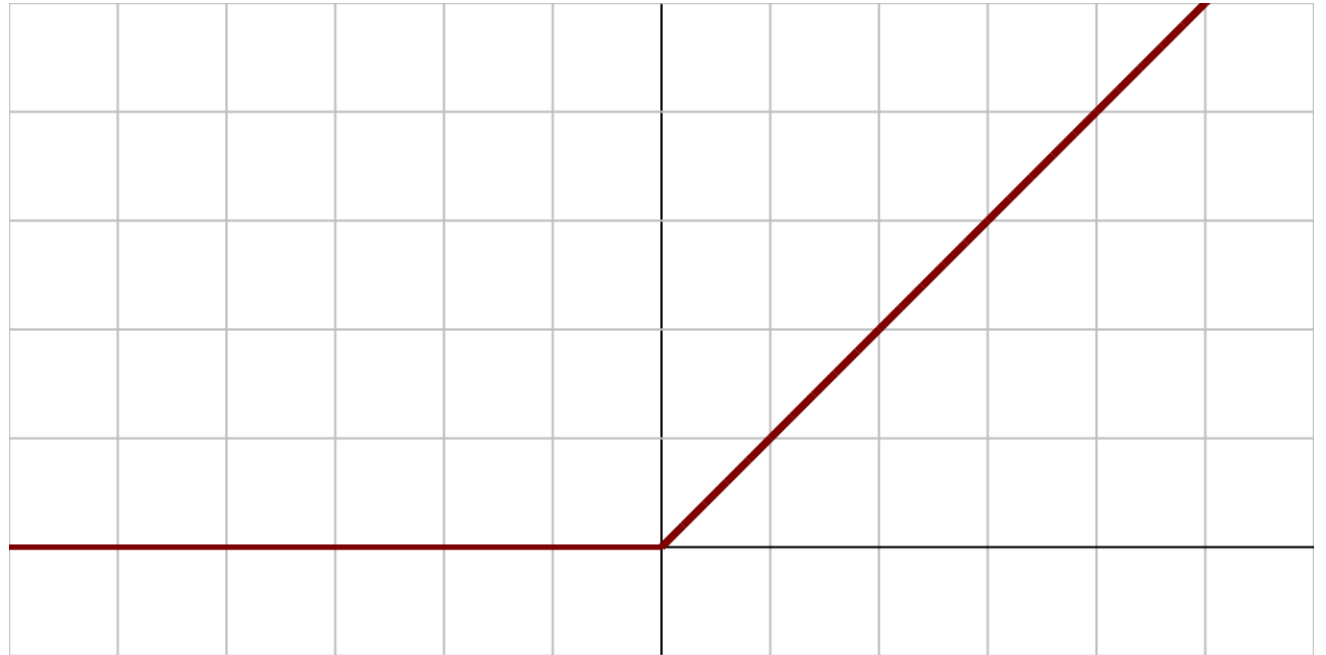


TanH AKA Hyperbolic Tangent

By Laughsinthestocks - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=44920533>

Rectified Linear Unit (ReLU)

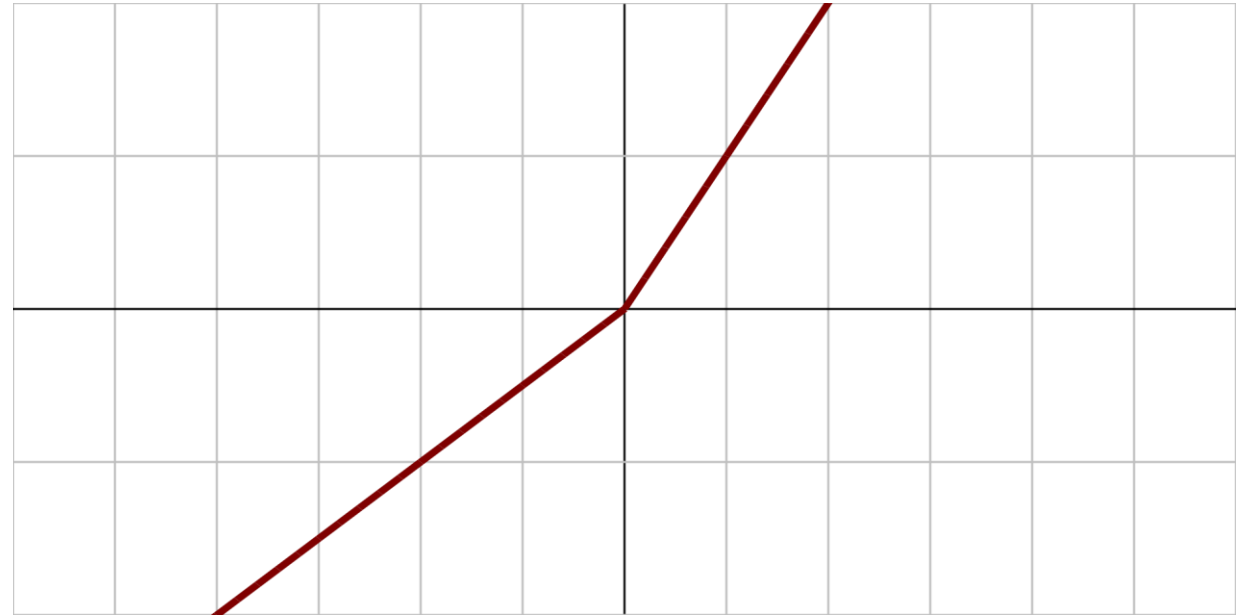
- Now we're talking
- Very popular choice
- Easy & fast to compute
- But, when inputs are zero or negative, we have a linear function and all of its problems
 - The “Dying ReLU problem”



By Laughsinthestocks - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=44920600>

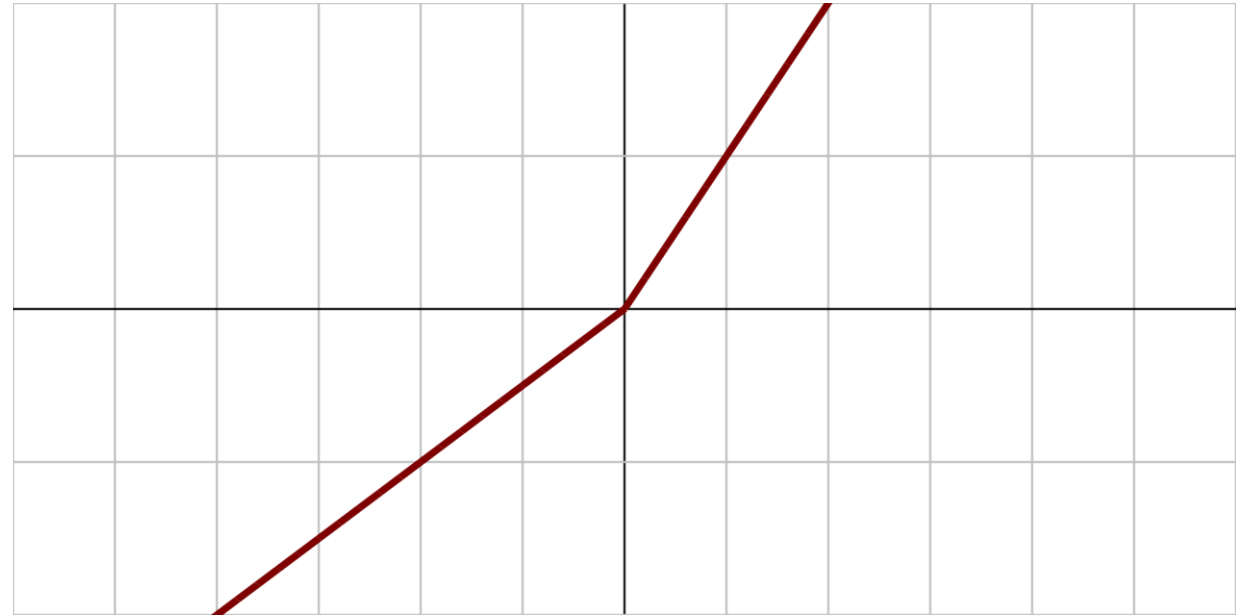
Leaky ReLU

- Solves “dying ReLU” by introducing a negative slope below 0 (usually not as steep as this)



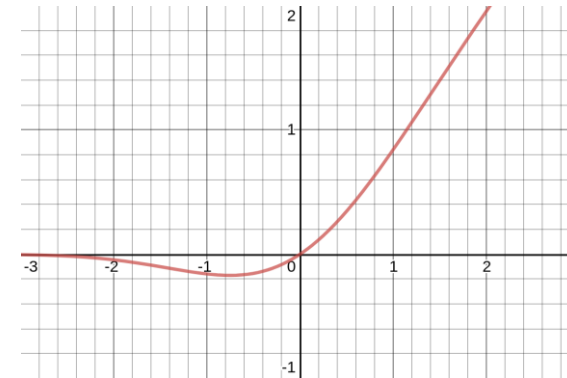
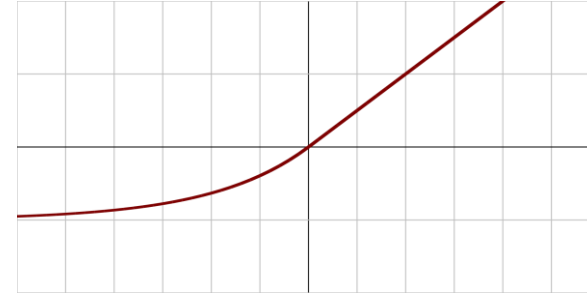
Parametric ReLU (PReLU)

- ReLU, but the slope in the negative part is learned via backpropagation
- Complicated and YMMV



Other ReLU variants

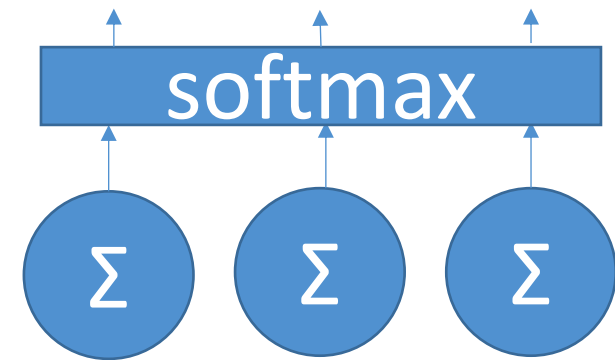
- Exponential Linear Unit (ELU)
- Swish
 - From Google, performs really well
 - But it's from Google, not Amazon...
 - Mostly a benefit with very deep networks (40+ layers)
- Maxout
 - Outputs the max of the inputs
 - Technically ReLU is a special case of maxout
 - But doubles parameters that need to be trained, not often practical.



By Ringdongling - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=85402414>

Softmax

- Used on the final output layer of a multi-class classification problem
- Basically converts outputs to probabilities of each classification
- Can't produce more than one label for something (sigmoid can)
- Don't worry about the actual function for the exam, just know what it's used for.



Choosing an activation function

- For multiple classification, use softmax on the output layer
- RNN's do well with Tanh
- For everything else
 - Start with ReLU
 - If you need to do better, try Leaky ReLU
 - Last resort: PReLU, Maxout
 - Swish for really deep networks

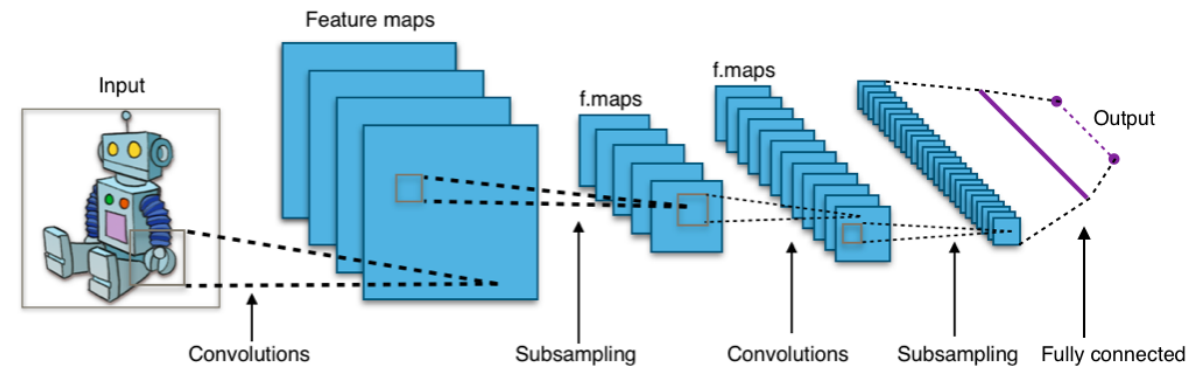
CNN's: what are they for?

- When you have data that doesn't neatly align into columns
 - Images that you want to find features within
 - Machine translation
 - Sentence classification
 - Sentiment analysis
- They can find features that aren't in a specific spot
 - Like a stop sign in a picture
 - Or words within a sentence
- They are “feature-location invariant”



CNN's: how do they work?

- Inspired by the biology of the visual cortex
 - Local receptive fields are groups of neurons that only respond to a part of what your eyes see (subsampling)
 - They overlap each other to cover the entire visual field (convolutions)
 - They feed into higher layers that identify increasingly complex images
 - Some receptive fields identify horizontal lines, lines at different angles, etc. (filters)
 - These would feed into a layer that identifies shapes
 - Which might feed into a layer that identifies objects
- For color images, extra layers for red, green, and blue



How do we “know” that’s a stop sign?

- Individual local receptive fields scan the image looking for edges, and pick up the edges of the stop sign in a layer
- Those edges in turn get picked up by a higher level convolution that identifies the stop sign’s shape (and letters, too)
- This shape then gets matched against your pattern of what a stop sign looks like, also using the strong red signal coming from your red layers
- That information keeps getting processed upward until your foot hits the brake!
- A CNN works the same way



CNN's with Keras / Tensorflow

- Source data must be of appropriate dimensions
 - ie width x length x color channels
- Conv2D layer type does the actual convolution on a 2D image
 - Conv1D and Conv3D also available – doesn't have to be image data
- MaxPooling2D layers can be used to reduce a 2D layer down by taking the maximum value in a given block
- Flatten layers will convert the 2D layer to a 1D layer for passing into a flat hidden layer of neurons
- Typical usage:
 - Conv2D -> MaxPooling2D -> Dropout -> Flatten -> Dense -> Dropout -> Softmax

CNN's are hard

- Very resource-intensive (CPU, GPU, and RAM)
- Lots of hyperparameters
 - Kernel sizes, many layers with different numbers of units, amount of pooling... in addition to the usual stuff like number of layers, choice of optimizer
- Getting the training data is often the hardest part! (As well as storing and accessing it)



Specialized CNN architectures

- Defines specific arrangement of layers, padding, and hyperparameters
- LeNet-5
 - Good for handwriting recognition
- AlexNet
 - Image classification, deeper than LeNet
- GoogLeNet
 - Even deeper, but with better performance
 - Introduces *inception modules* (groups of convolution layers)
- ResNet (Residual Network)
 - Even deeper – maintains performance via *skip connections*.

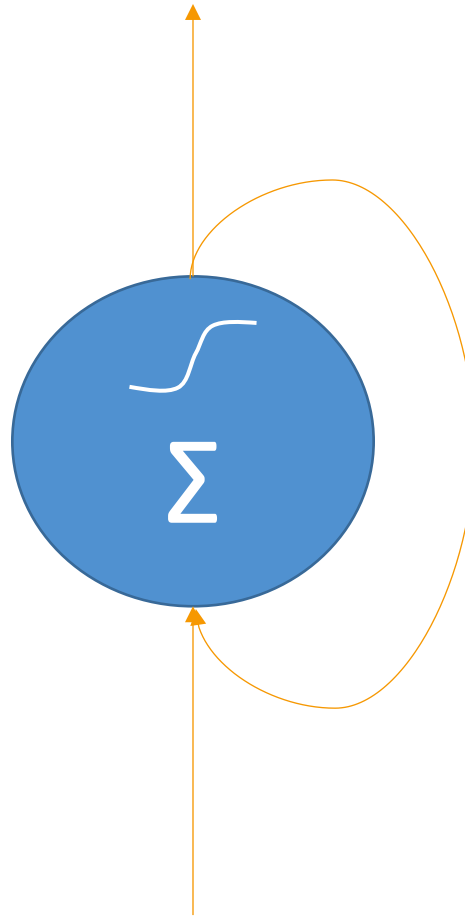
Recurrent Neural Networks

RNN's: what are they for?

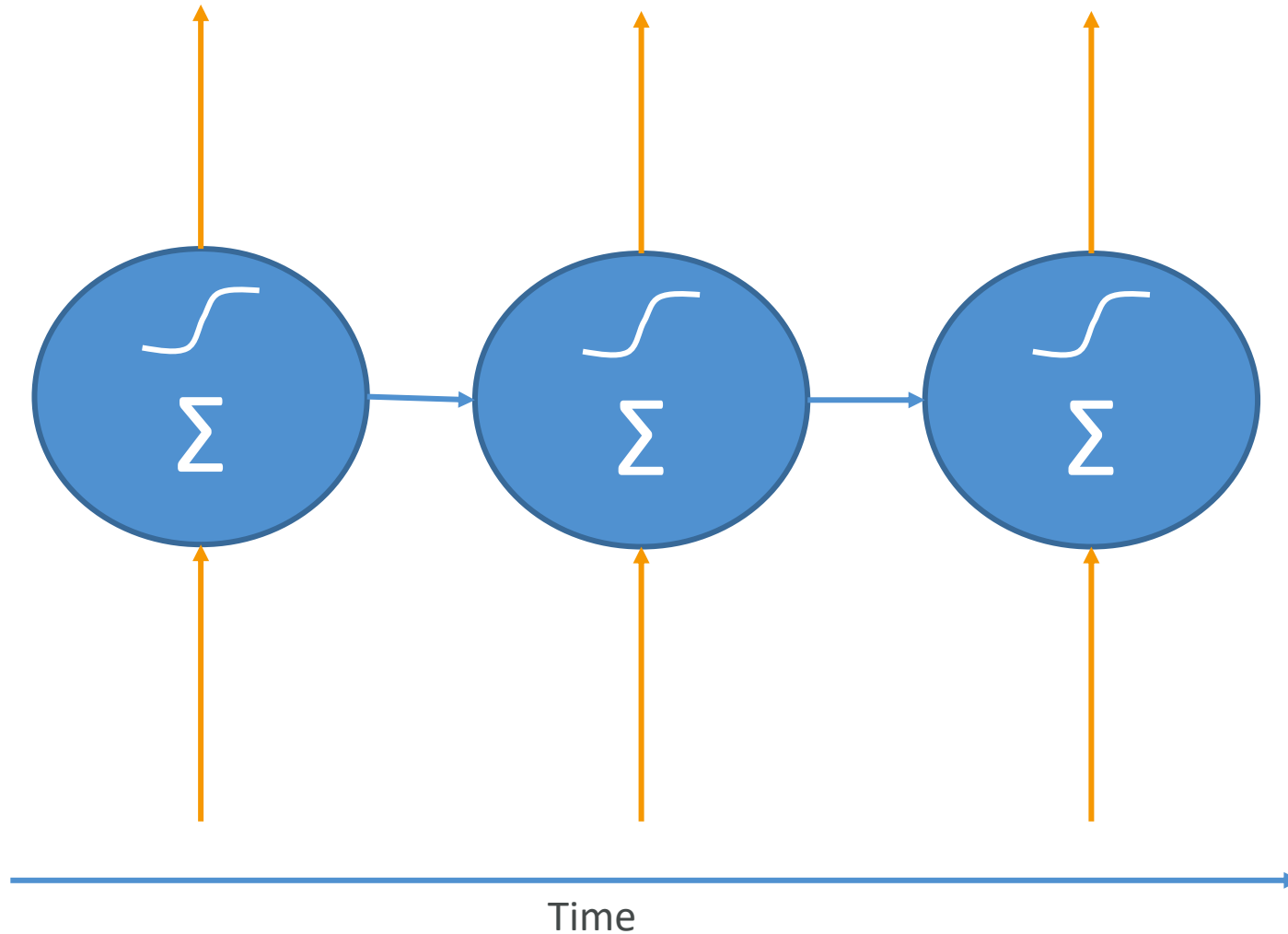
- Time-series data
 - When you want to predict future behavior based on past behavior
 - Web logs, sensor logs, stock trades
 - Where to drive your self-driving car based on past trajectories
- Data that consists of sequences of arbitrary length
 - Machine translation
 - Image captions
 - Machine-generated music



A recurrent neuron

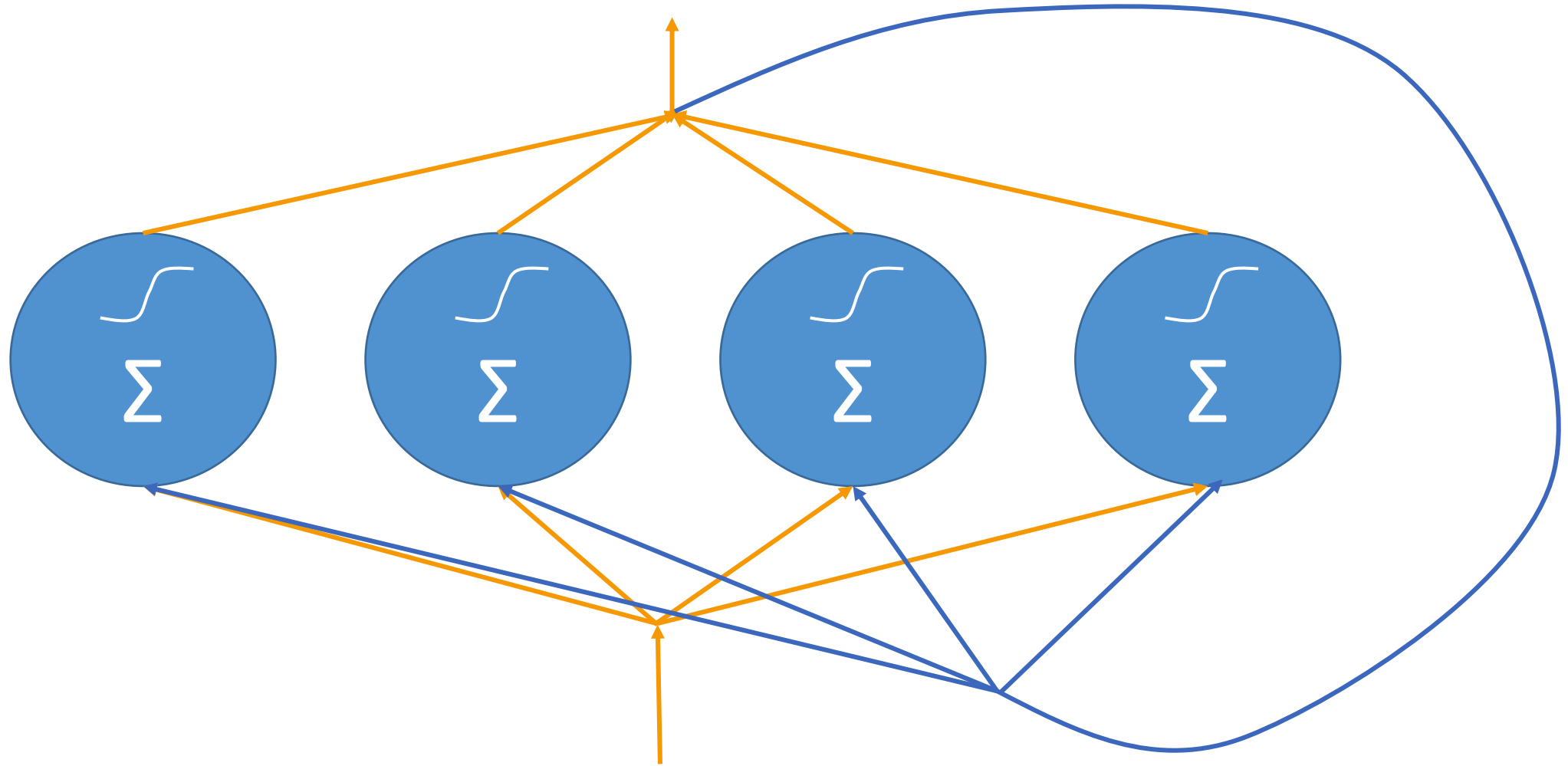


Another way to look at it



A “Memory Cell”

A layer of recurrent neurons



RNN topologies

- Sequence to sequence
 - i.e., predict stock prices based on series of historical data
- Sequence to vector
 - i.e., words in a sentence to sentiment
- Vector to sequence
 - i.e., create captions from an image
- Encoder -> Decoder
 - Sequence -> vector -> sequence
 - i.e., machine translation

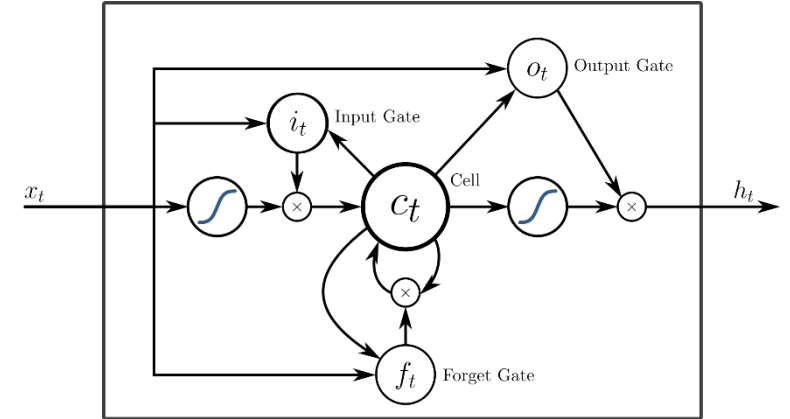


Training rnn's

- Backpropagation through time
 - Just like backpropagation on MLP's, but applied to each time step.
- All those time steps add up fast
 - Ends up looking like a really, really deep neural network.
 - Can limit backpropagation to a limited number of time steps (truncated backpropagation through time)

Training rnn's

- State from earlier time steps get diluted over time
 - This can be a problem, for example when learning sentence structures
- LSTM Cell
 - Long Short-Term Memory Cell
 - Maintains separate short-term and long-term states
- GRU Cell
 - Gated Recurrent Unit
 - Simplified LSTM Cell that performs about as well



Training rnn's

- It's really hard
 - Very sensitive to topologies, choice of hyperparameters
 - Very resource intensive
 - A wrong choice can lead to a RNN that doesn't converge at all.



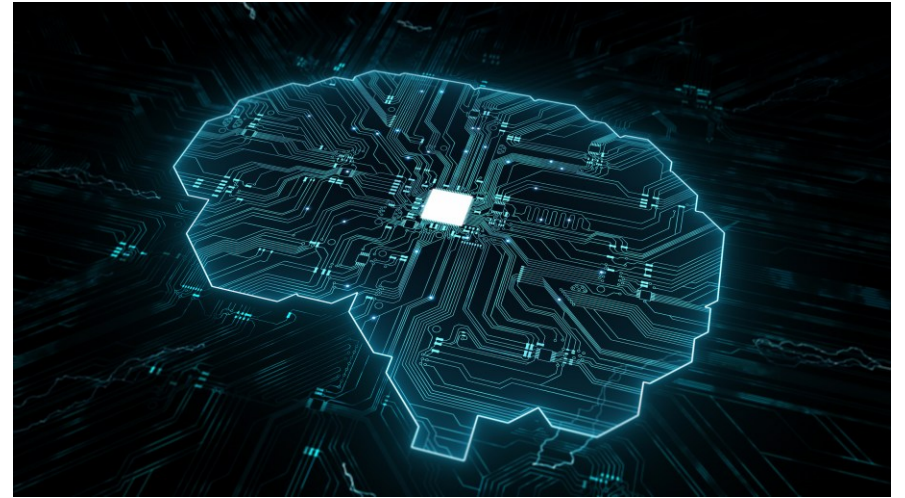
Modern Natural Language Processing

- **Transformer** deep learning architectures are what's hot
 - Adopts mechanism of “self-attention”
 - Weighs significance of each part of the input data
 - Processes sequential data (like words, like an RNN), but processes entire input all at once.
 - The attention mechanism provides context, so no need to process one word at a time.
 - BERT, RoBERTa, T5, GPT-2 etc., DistilBERT
 - DistilBERT: uses knowledge distillation to reduce model size by 40%
- BERT: Bi-directional Encoder Representations from Transformers
- GPT: Generative Pre-trained Transformer



Transfer Learning

- NLP models (and others) are too big and complex to build from scratch and re-train every time
 - The latest may have hundreds of billions of parameters!
- Model zoos such as **Hugging Face** offer pre-trained models to start from
 - Integrated with Sagemaker via Hugging Face Deep Learning Containers
- You can fine-tune these models for your own use cases



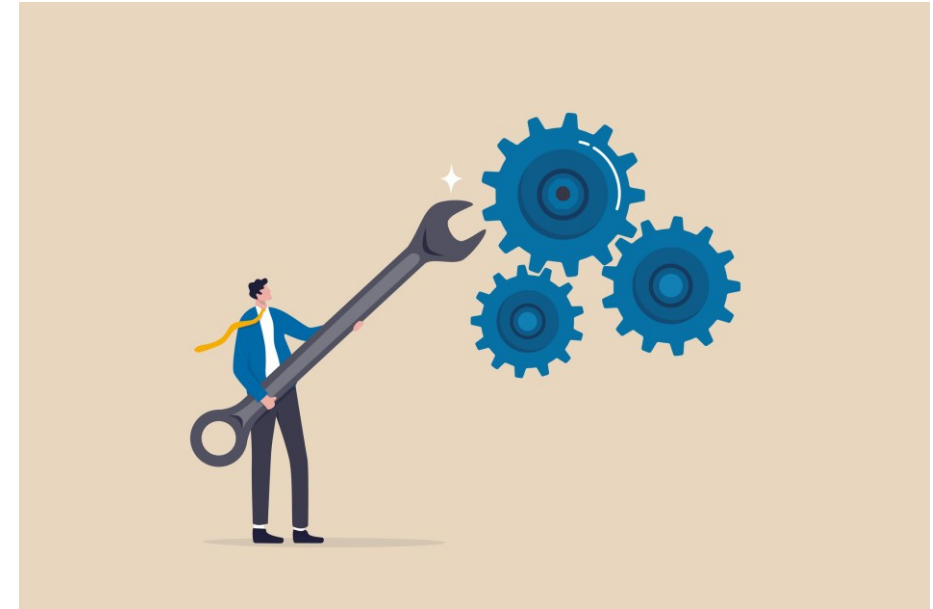
Transfer Learning: BERT example

- Hugging Face offers a Deep Learning Container (DLC) for BERT
- It's pre-trained on BookCorpus and Wikipedia
- You can **fine-tune** BERT (or DistilBERT etc) with your own additional training data through **transfer learning**
 - Tokenize your own training data to be of the same format
 - Just start training it further with your data, with a low learning rate.



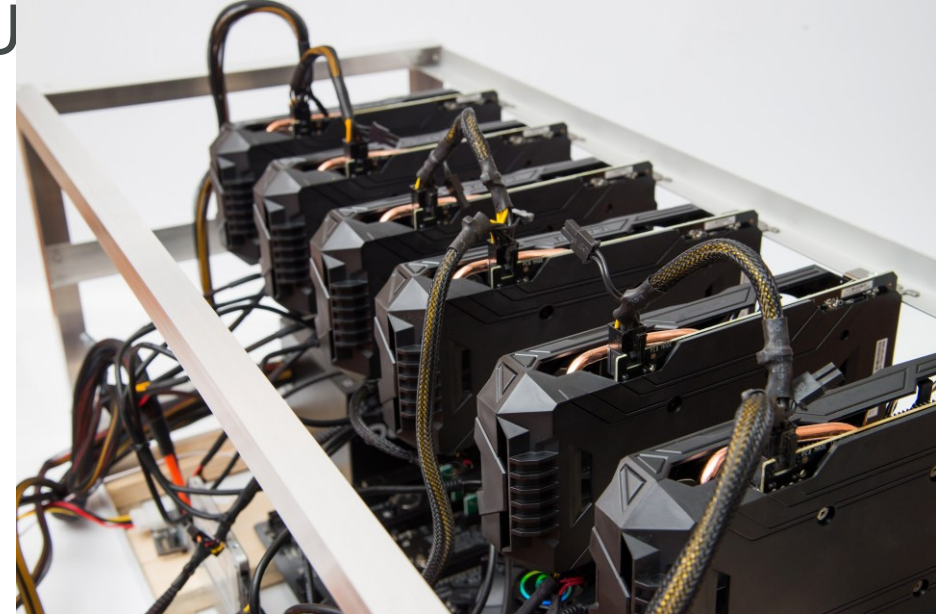
Transfer Learning approaches

- Continue training a pre-trained model (**fine-tuning**)
 - Use for fine-tuning a model that has way more training data than you'll ever have
 - Use a low learning rate to ensure you are just incrementally improving the model
- Add new trainable layers to the top of a frozen model
 - Learns to turn old features into predictions on new data
 - Can do both: add new layers, then fine tune as well
- Retrain from scratch
 - If you have large amounts of training data, and it's fundamentally different from what the model was pre-trained with
 - And you have the computing capacity for it!
- Use it as-is
 - When the model's training data is what you want already



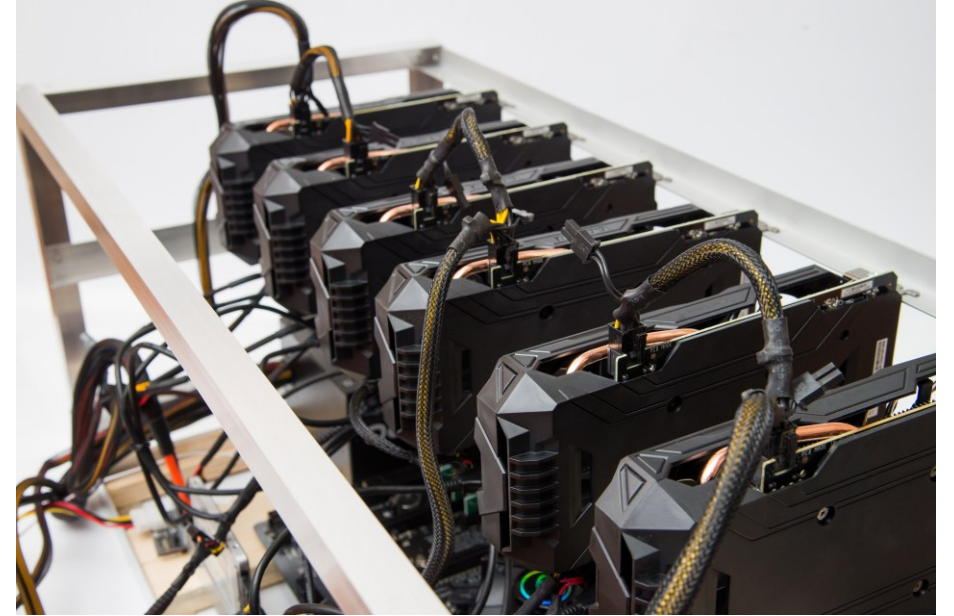
Deep Learning on EC2 / EMR

- EMR supports Apache MXNet and GPU instance types
- Appropriate instance types for deep learning:
 - P3: 8 Tesla V100 GPU's
 - P2: 16 K80 GPU's
 - G3: 4 M60 GPU's (all Nvidia chips)
 - G5g: AWS Graviton 2 processors / Nvidia T4G Tensor Core GPU's
 - Not (yet) available in EMR
 - Also used for Android game streaming
 - P4d – A100 “UltraClusters” for supercomputing
- Deep Learning AMI's



Deep Learning on EC2 / EMR

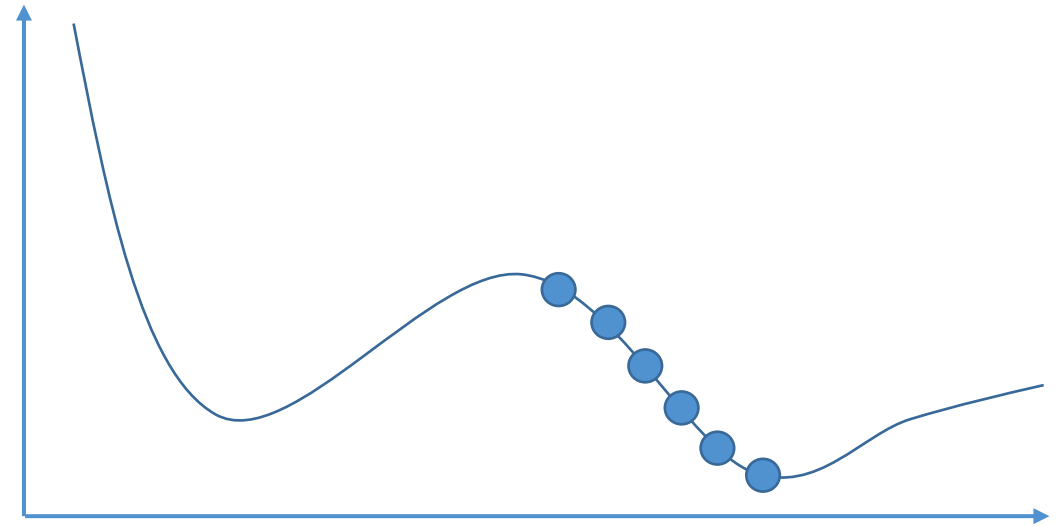
- Trn1 instances
 - “Powered by Trainium”
 - Optimized for training (50% savings)
 - 800 Gbps of Elastic Fabric Adapter (EFA) networking for fast clusters
- Trn1n instances
 - Even more bandwidth (1600 Gbps)
- Inf2 instances
 - “Powered by AWS Inferentia2”
 - Optimized for inference



Tuning Neural Networks

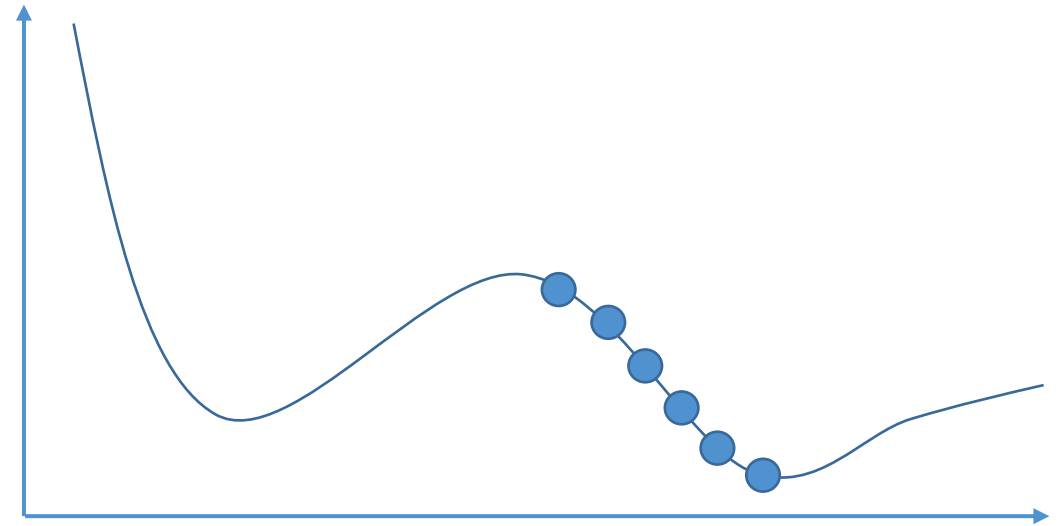
Learning Rate

- Neural networks are trained by gradient descent (or similar means)
- We start at some random point, and sample different solutions (weights) seeking to minimize some cost function, over many *epochs*
- How far apart these samples are is the *learning rate*



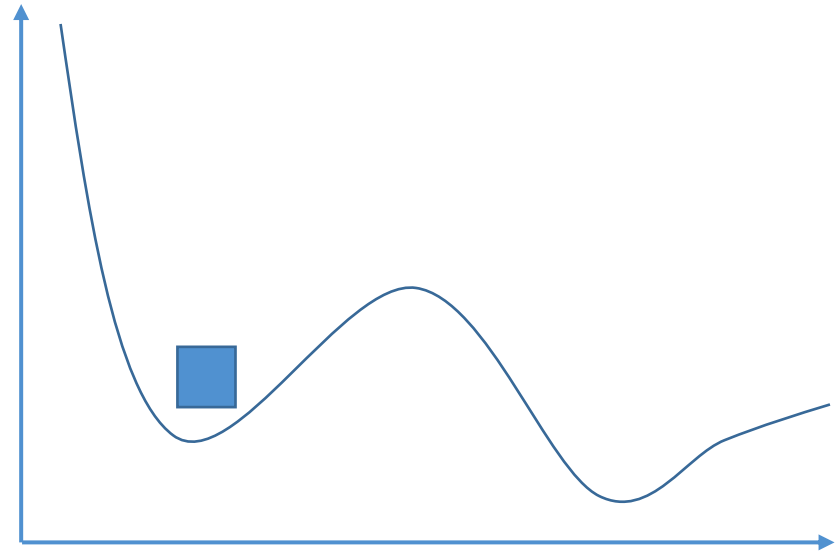
Effect of learning rate

- Too high a learning rate means you might overshoot the optimal solution!
- Too small a learning rate will take too long to find the optimal solution
- Learning rate is an example of a *hyperparameter*



Batch Size

- How many training samples are used within each batch of each epoch
- Somewhat counter-intuitively:
 - Smaller batch sizes can work their way out of “local minima” more easily
 - Batch sizes that are too large can end up getting stuck in the wrong solution
 - Random shuffling at each epoch can make this look like very inconsistent results from run to run



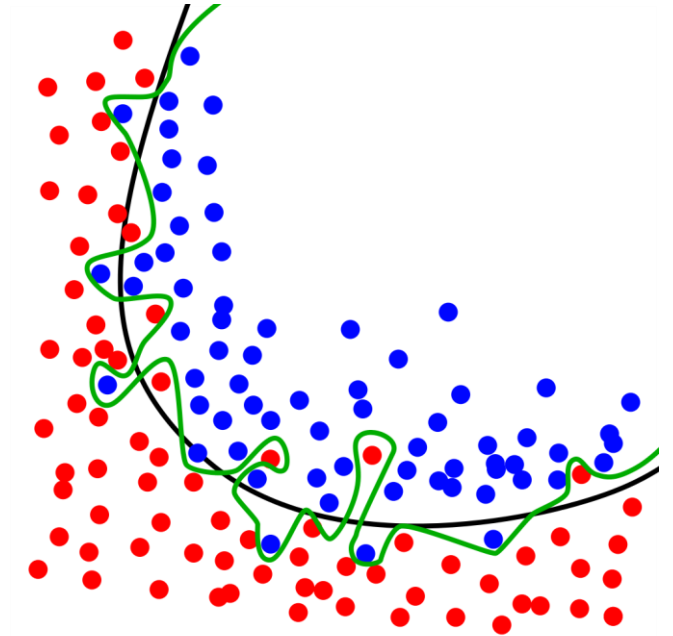
To Recap (this is important!)

- Small batch sizes tend to not get stuck in local minima
- Large batch sizes can converge on the wrong solution at random
- Large learning rates can overshoot the correct solution
- Small learning rates increase training time

Neural Network Regularization Techniques

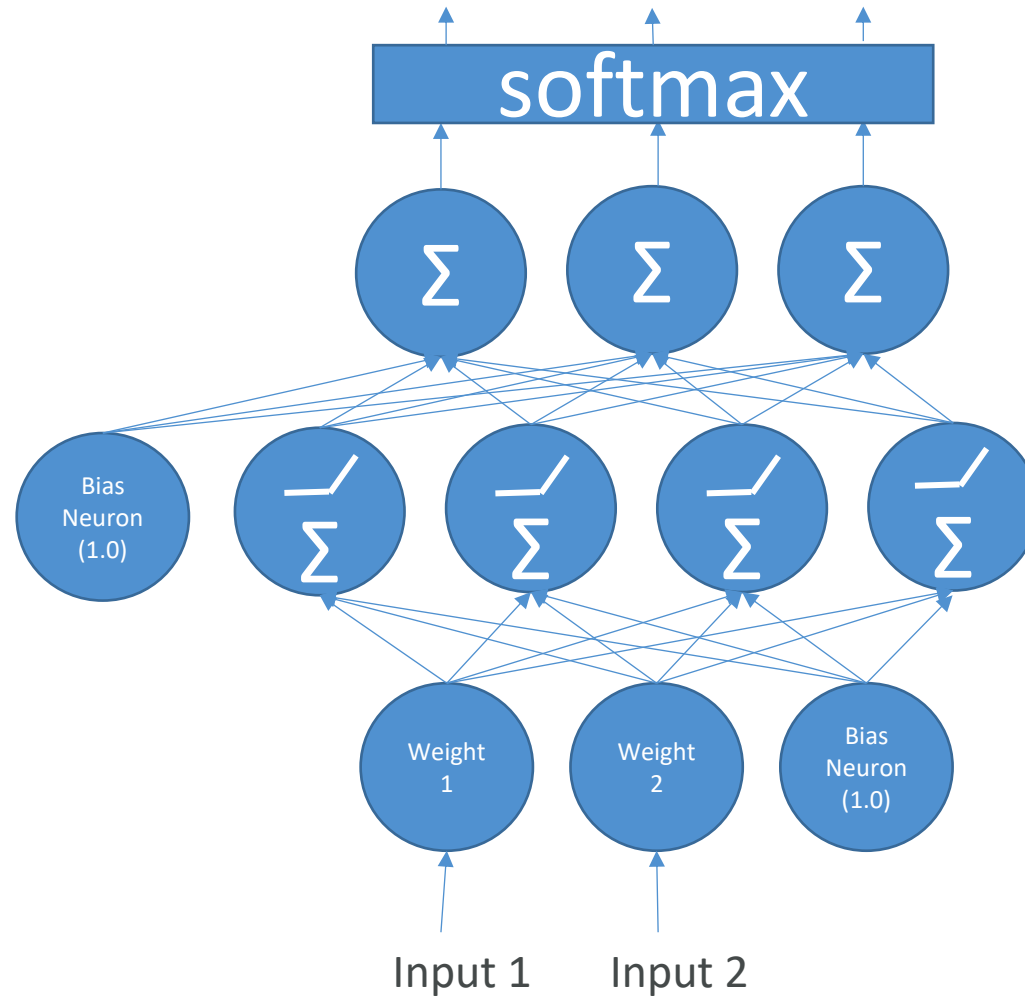
What is regularization?

- Preventing *overfitting*
 - Models that are good at making predictions on the data they were trained on, but not on new data it hasn't seen before
 - Overfitted models have learned patterns in the training data that don't generalize to the real world
 - Often seen as high accuracy on training data set, but lower accuracy on test or evaluation data set.
 - When training and evaluating a model, we use *training*, *evaluation*, and *testing* data sets.
- Regularization techniques are intended to prevent overfitting.

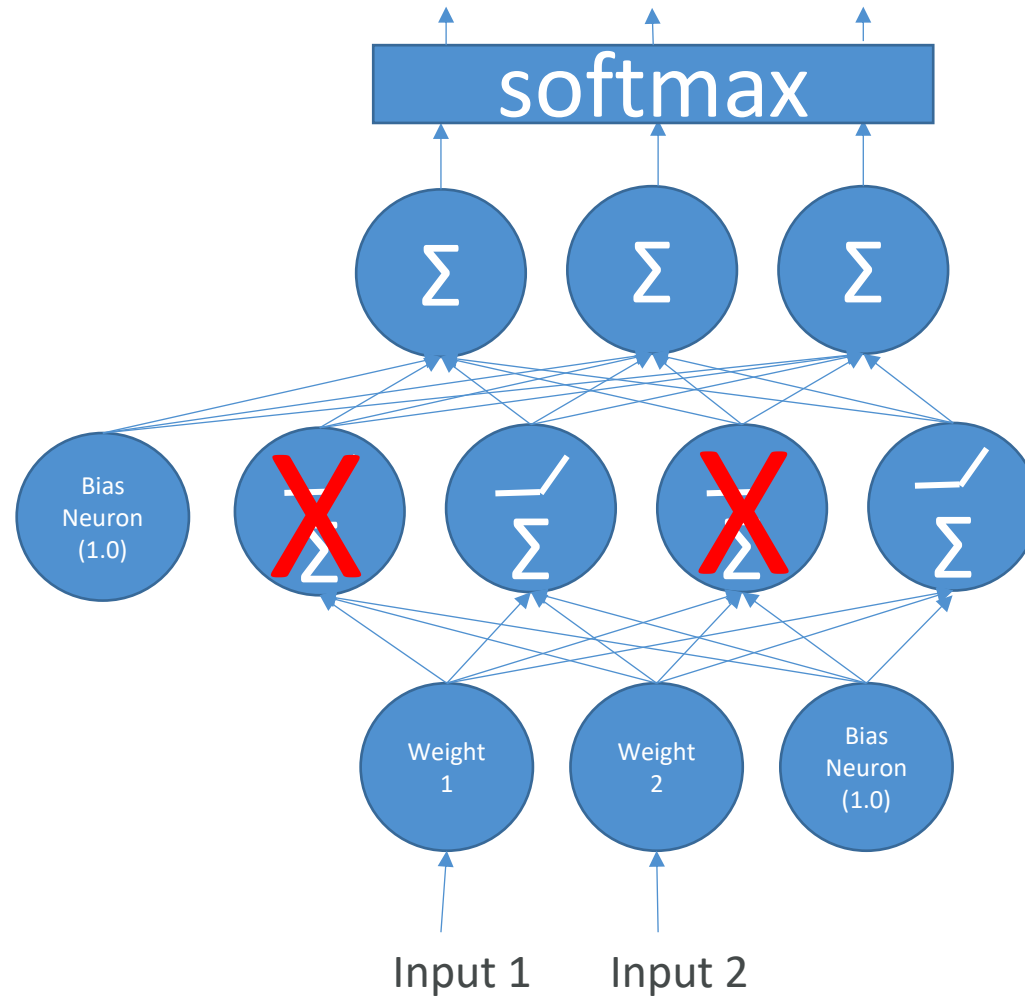


Chabacano [CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>)]

Too many layers? Too many neurons?



Dropout



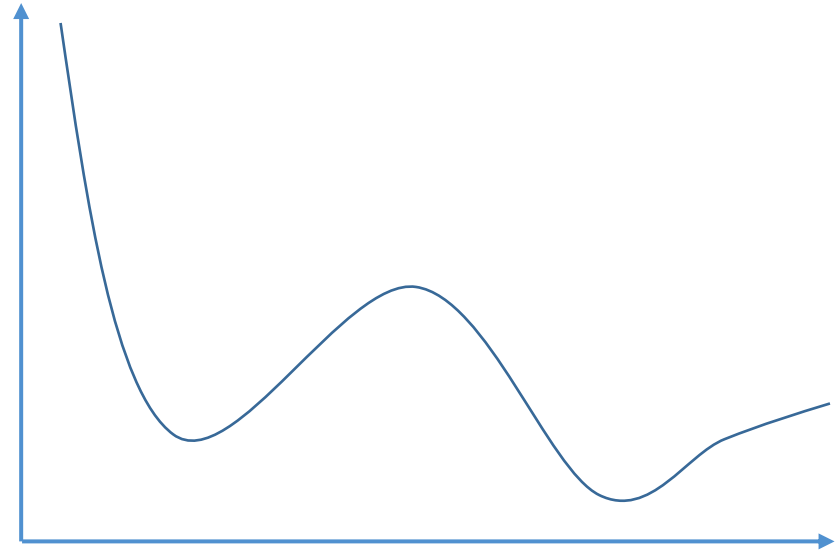
Early Stopping

```
Epoch 1/10
- 4s - loss: 0.2406 - acc: 0.9302 - val_loss: 0.1437 - val_acc: 0.9557
Epoch 2/10
- 2s - loss: 0.0971 - acc: 0.9712 - val_loss: 0.0900 - val_acc: 0.9725
Epoch 3/10
- 2s - loss: 0.0653 - acc: 0.9803 - val_loss: 0.0725 - val_acc: 0.9786
Epoch 4/10
- 2s - loss: 0.0471 - acc: 0.9860 - val_loss: 0.0689 - val_acc: 0.9795
Epoch 5/10
- 2s - loss: 0.0367 - acc: 0.9890 - val_loss: 0.0675 - val_acc: 0.9808
Epoch 6/10
- 2s - loss: 0.0266 - acc: 0.9919 - val_loss: 0.0680 - val_acc: 0.9796
Epoch 7/10
- 2s - loss: 0.0208 - acc: 0.9937 - val_loss: 0.0678 - val_acc: 0.9811
Epoch 8/10
- 2s - loss: 0.0157 - acc: 0.9953 - val_loss: 0.0719 - val_acc: 0.9810
Epoch 9/10
- 2s - loss: 0.0130 - acc: 0.9960 - val_loss: 0.0707 - val_acc: 0.9825
Epoch 10/10
- 2s - loss: 0.0097 - acc: 0.9972 - val_loss: 0.0807 - val_acc: 0.9805
```

Grief with Gradients

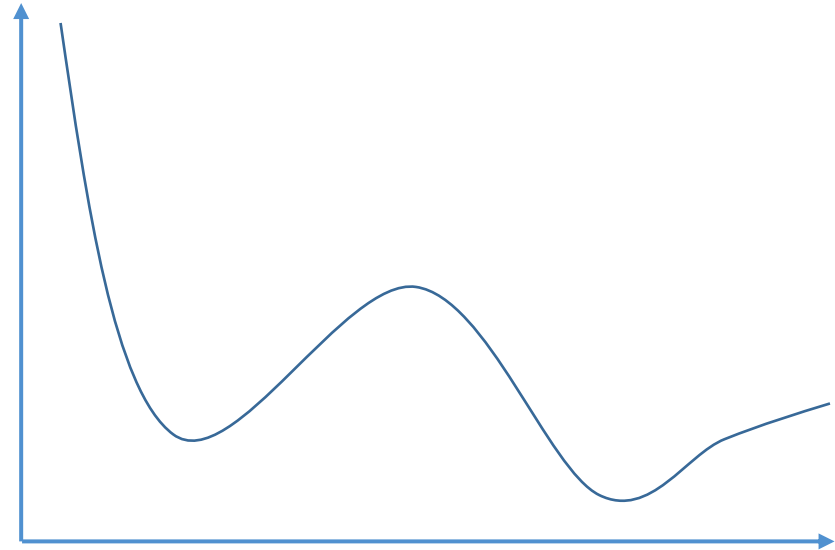
The Vanishing Gradient Problem

- When the slope of the learning curve approaches zero, things can get stuck
- We end up working with very small numbers that slow down training, or even introduce numerical errors
- Becomes a problem with deeper networks and RNN's as these “vanishing gradients” propagate to deeper layers
- Opposite problem: “exploding gradients”



Fixing the Vanishing Gradient Problem

- Multi-level hierarchy
 - Break up levels into their own sub-networks trained individually
- Long short-term memory (LSTM)
- Residual Networks
 - i.e., ResNet
 - Ensemble of shorter networks
- Better choice of activation function
 - ReLU is a good choice



Gradient Checking

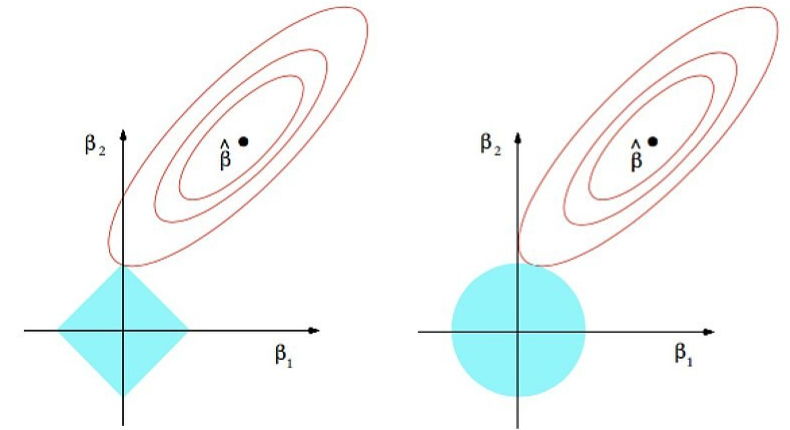
- A debugging technique
- Numerically check the derivatives computed during training
- Useful for validating code of neural network training
 - But you're probably not going to be writing this code...



L1 and L2 Regularization

L1 and L2 Regularization

- Preventing overfitting in ML in general
- A regularization term is added as weights are learned
- L1 term is the sum of the weights
 - $\lambda \sum_{i=1}^k |w_i|$
- L2 term is the sum of the square of the weights
 - $\lambda \sum_{i=1}^k w_i^2$
- Same idea can be applied to loss functions



Xiaoli C. [CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>)]

What's the difference?

- L1: sum of weights
 - Performs *feature selection* – entire features go to 0
 - Computationally inefficient
 - Sparse output
- L2: sum of square of weights
 - All features remain considered, just weighted
 - Computationally efficient
 - Dense output

Why would you want L1?

- Feature selection can reduce dimensionality
 - Out of 100 features, maybe only 10 end up with non-zero coefficients!
 - The resulting sparsity can make up for its computational inefficiency
- But, if you think all of your features are important, L2 is probably a better choice.

Confusion Matrix

Sometimes accuracy doesn't tell the whole story

- A test for a rare disease can be 99.9% accurate by just guessing “no” all the time
- We need to understand true positives and true negative, as well as false positives and false negatives.
- A confusion matrix shows this.



Binary confusion matrix

	Actual YES	Actual NO
Predicted YES	TRUE POSITIVES	FALSE POSITIVES
Predicted NO	FALSE NEGATIVES	TRUE NEGATIVE

Image has cat?

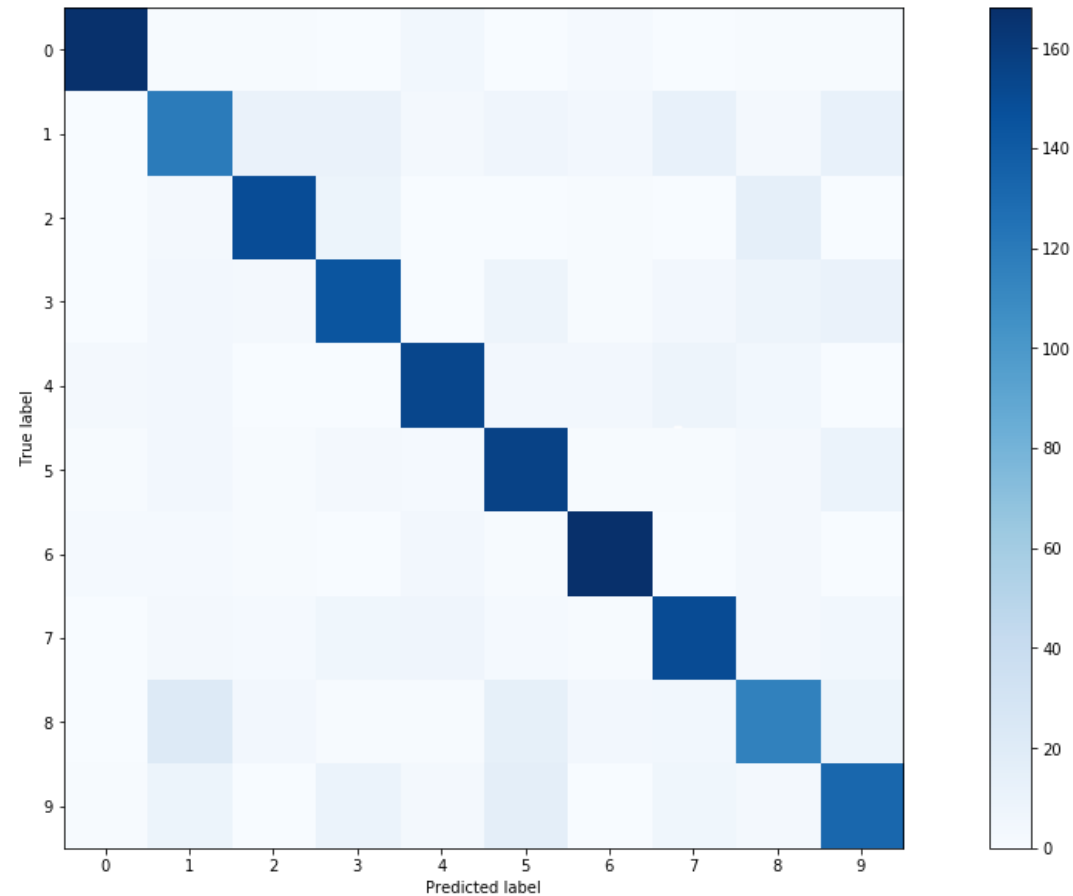
	Actual cat	Actual not cat
Predicted cat	50	5
Predicted not cat	10	100



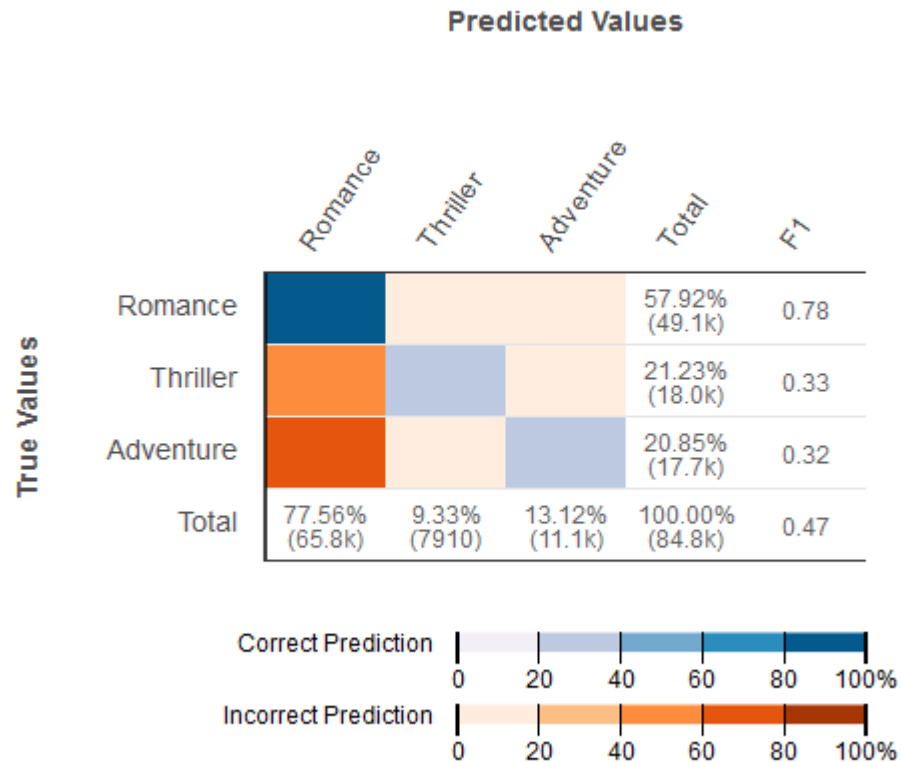
Another format

	Predicted NO	Predicted YES	
Actual NO	50	5	55
Actual YES	10	100	110
	60	105	

Multi-class confusion matrix + heat map



Example from the AWS docs



- # of correct and incorrect predictions per class (infer from colors of each cell)
- F1 scores per class
- True class frequencies: the “total” column
- Predicted class frequencies: the “total” row

Measuring your Models

Remember our friend the confusion matrix

	Actual YES	Actual NO
Predicted YES	TRUE POSITIVES	FALSE POSITIVES
Predicted NO	FALSE NEGATIVES	TRUE NEGATIVE

Recall

- $$\frac{TRUE POSITIVES}{TRUE POSITIVES + FALSE NEGATIVES}$$
- AKA Sensitivity, True Positive rate, Completeness
- Percent of positives rightly predicted
- Good choice of metric when you care a lot about false negatives
 - i.e., fraud detection

Recall example

	Actual fraud	Actual not fraud
Predicted fraud	5	20
Predicted not fraud	10	100

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{Recall} = 5 / (5 + 10) = 5 / 15 = 1 / 3 = 33\%$$

Precision

- $$\frac{TRUE POSITIVES}{TRUE POSITIVES + FALSE POSITIVES}$$
- AKA Correct Positives
- Percent of relevant results
- Good choice of metric when you care a lot about false positives
 - i.e., medical screening, drug testing

Precision example

	Actual fraud	Actual not fraud
Predicted fraud	5	20
Predicted not fraud	10	100

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

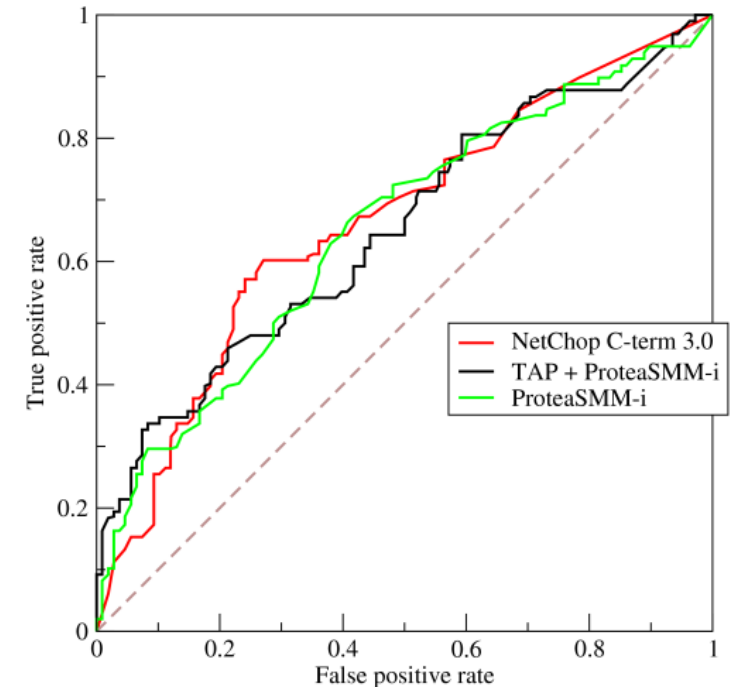
$$\text{Precision} = 5 / (5 + 20) = 5 / 25 = 1 / 5 = 20\%$$

Other metrics

- Specificity = $\frac{TN}{TN+FP}$ = “True negative rate”
- F1 Score
 - $\frac{2TP}{2TP+FP+FN}$
 - $2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
 - Harmonic mean of precision and sensitivity
 - When you care about precision AND recall
- RMSE
 - Root mean squared error, exactly what it sounds like
 - Accuracy measurement
 - Only cares about right & wrong answers

ROC Curve

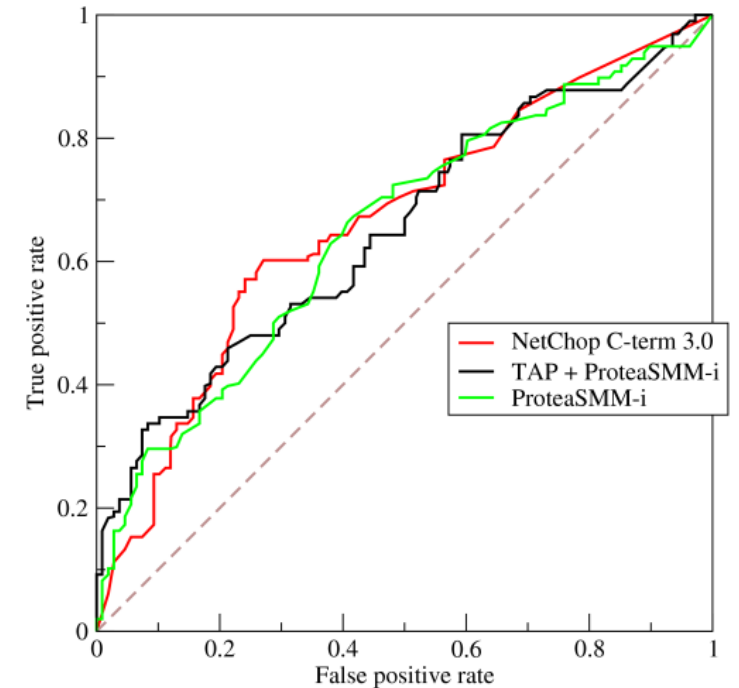
- Receiver Operating Characteristic Curve
- Plot of true positive rate (recall) vs. false positive rate at various threshold settings.
- Points above the diagonal represent good classification (better than random)
- Ideal curve would just be a point in the upper-left corner
- The more it's “bent” toward the upper-left, the better



BOR at the English language Wikipedia [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0/>)]

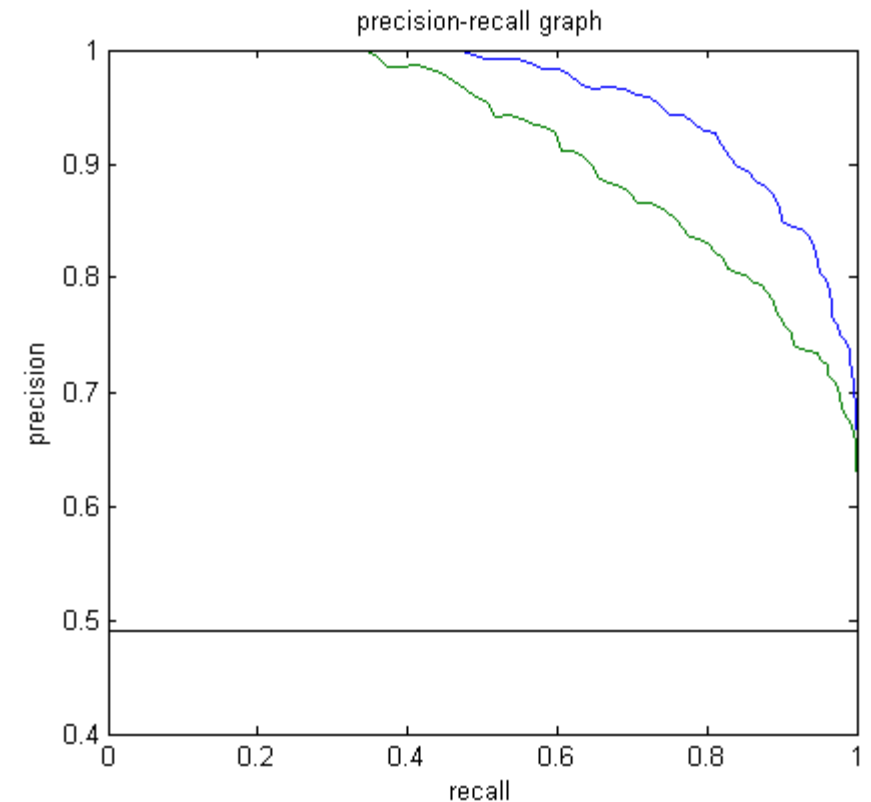
AUC

- The area under the ROC curve is... wait for it..
- Area Under the Curve (AUC)
- Equal to probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one
- ROC AUC of 0.5 is a useless classifier, 1.0 is perfect
- Commonly used metric for comparing classifiers



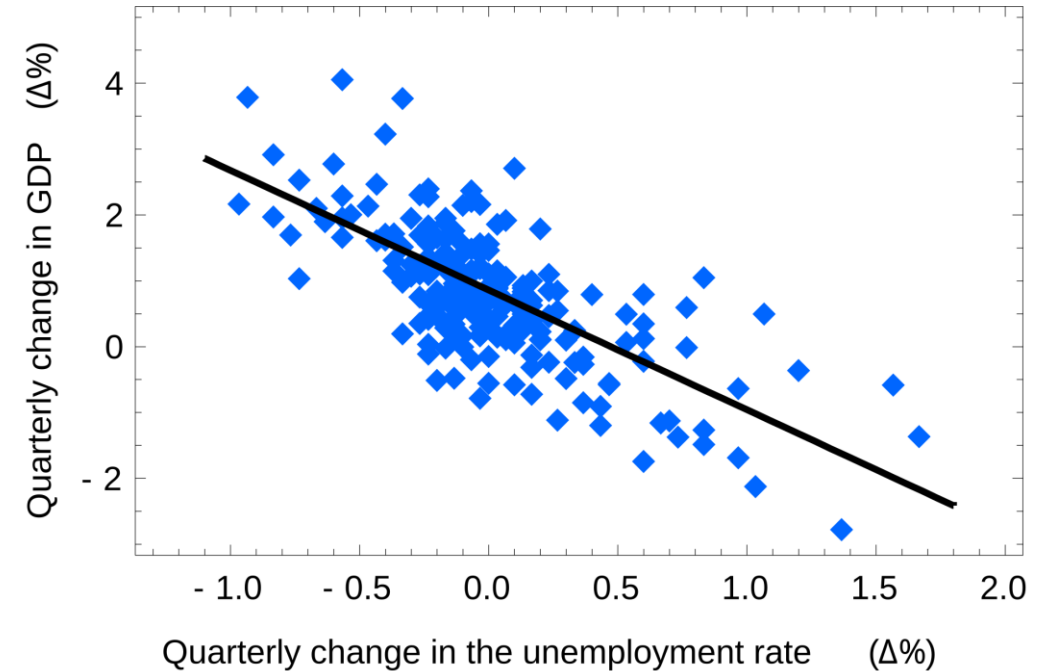
P-R Curve

- Precision / Recall curve
- Good = higher area under curve
- Similar to ROC curve
 - But better suited for information retrieval problems
 - ROC can result in very small values if you are searching large number of documents for a tiny number that are relevant



Other Types of Metrics

- We have been talking about measuring CLASSIFICATION problems so far
 - Accuracy, precision, recall, F1, etc.
- What if you are predicting values (numbers) and not discrete classifications?
 - R^2 (R-squared) – Square of the correlation coefficient between observed outcomes and predicted
 - Measuring error between actual and predicted values:
 - **RMSE** (Root Mean-Squared Error)
 - **MAE** (Mean Absolute Error)

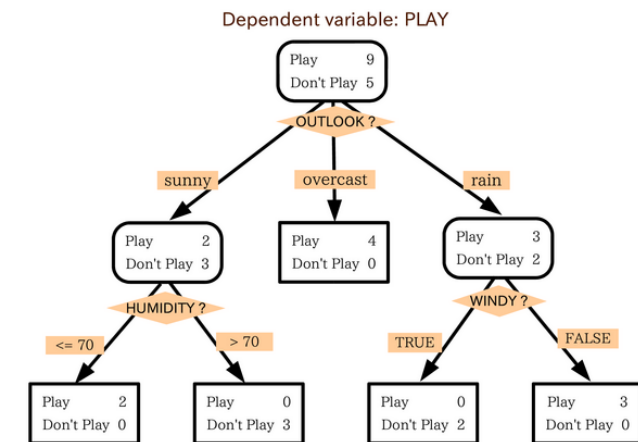
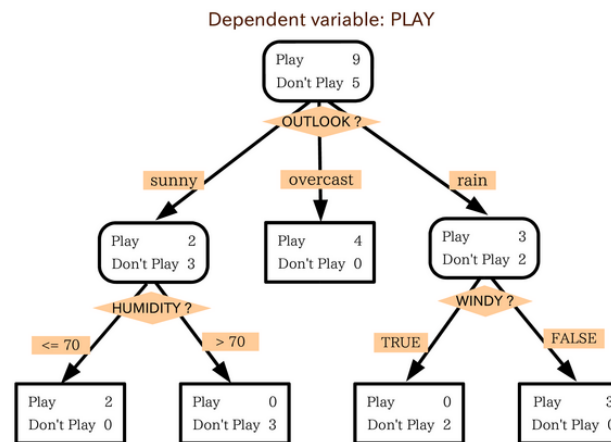
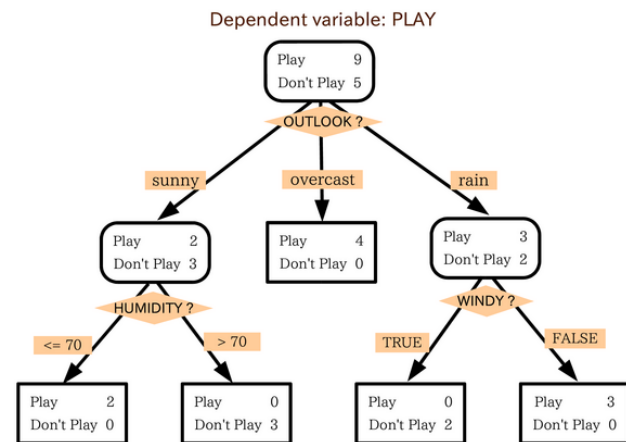


Ensemble Learning

Bagging & Boosting

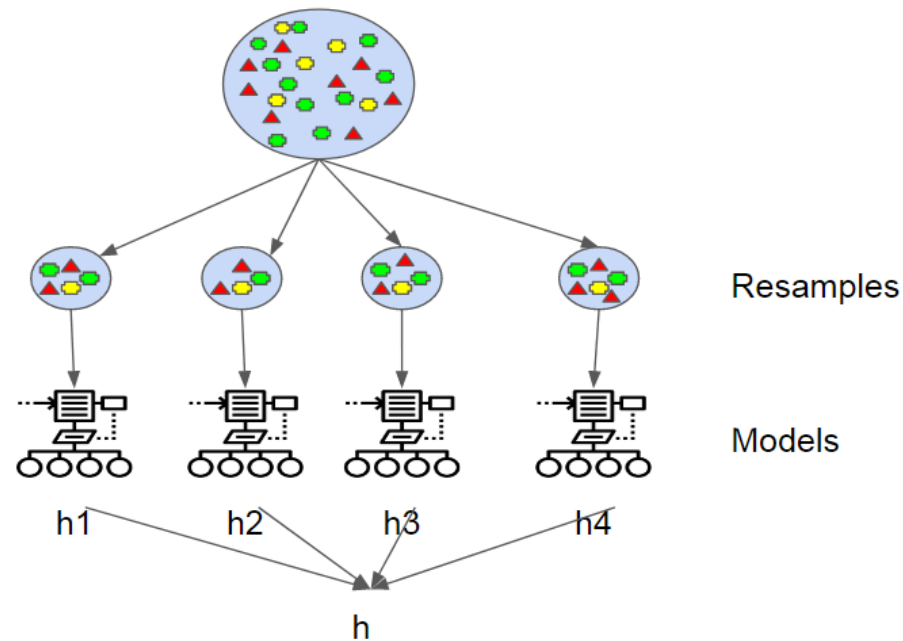
Ensemble methods

- Common example: random forest
 - Decision trees are prone to overfitting
 - So, make lots of decision trees and let them all vote on the result
 - This is a random forest
 - How do they differ?



Bagging

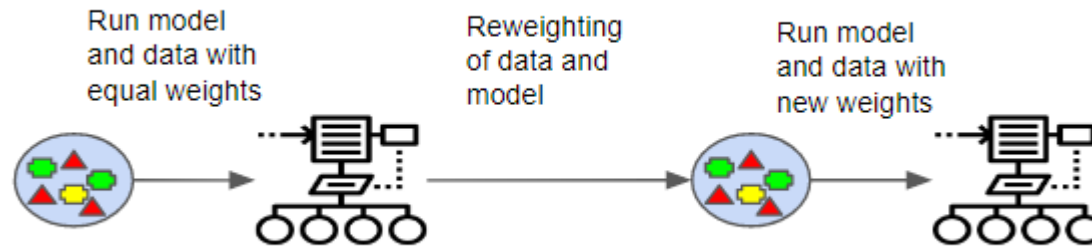
- Generate N new training sets by **random sampling with replacement**
- Each resampled model can be trained in parallel



SeattleDataGuy [CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>)]

Boosting

- Observations are weighted
- Some will take part in new training sets more often
- Training is sequential; each classifier takes into account the previous one's success.



SeattleDataGuy [CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0>)]

Bagging vs. Boosting

- XGBoost is the latest hotness
- Boosting generally yields better accuracy
- But bagging avoids overfitting
- Bagging is easier to parallelize
- So, depends on your goal

Automatic Model Tuning

With SageMaker

Hyperparameter tuning

- How do you know the best values of learning rate, batch size, depth, etc?
- Often you have to experiment
- Problem blows up quickly when you have many different hyperparameters; need to try every combination of every possible value somehow, train a model, and evaluate it every time



Automatic Model Tuning

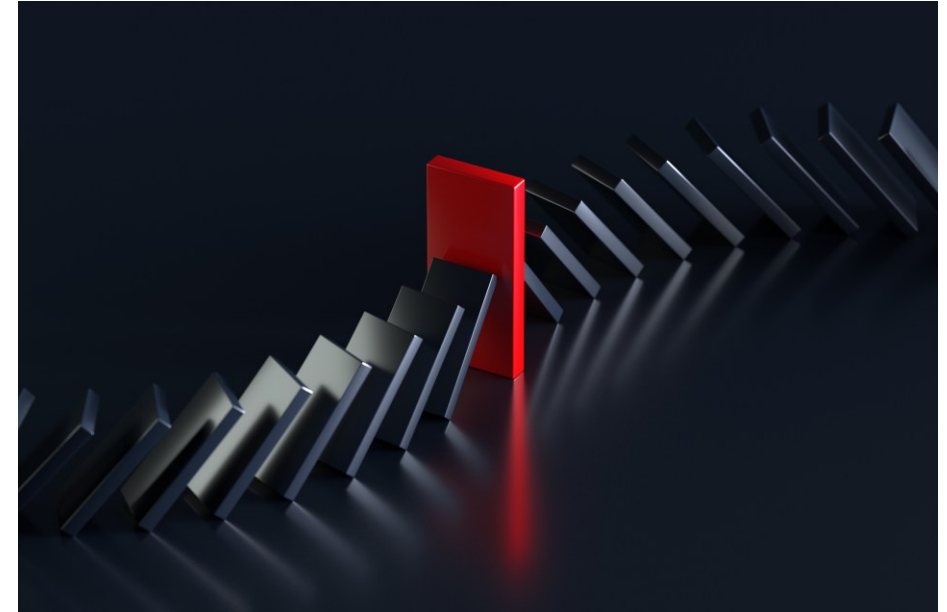
- Define the hyperparameters you care about and the ranges you want to try, and the metrics you are optimizing for
- SageMaker spins up a “HyperParameter Tuning Job” that trains as many combinations as you’ll allow
 - Training instances are spun up as needed, potentially a lot of them
- The set of hyperparameters producing the best results can then be deployed as a model
- **It learns as it goes**, so it doesn’t have to try every possible combination

Automatic Model Tuning: Best Practices

- Don't optimize too many hyperparameters at once
- Limit your ranges to as small a range as possible
- Use logarithmic scales when appropriate
- Don't run too many training jobs concurrently
 - This limits how well the process can learn as it goes
- Make sure training jobs running on multiple instances report the correct objective metric in the end

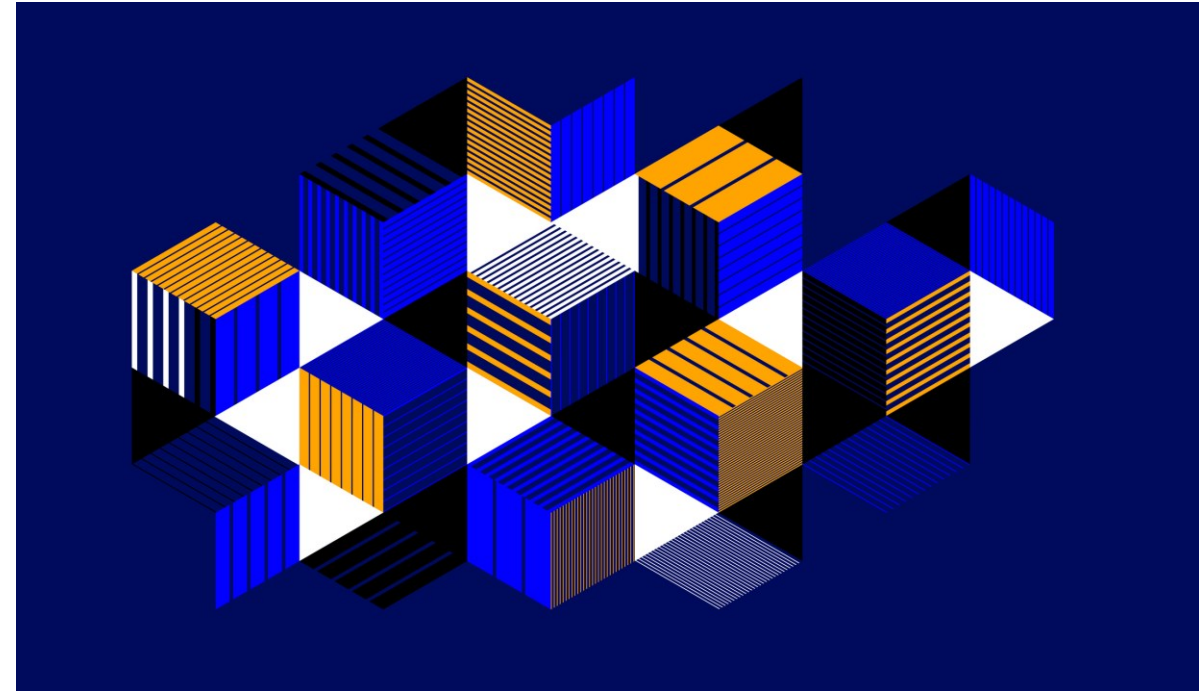
Hyperparameter Tuning in AMT

- Early Stopping
 - Stop training in a tuning job early if it is not improving the objective significantly
 - Reduces computing time, avoids overfitting
 - Set “early stopping” to “Auto”
 - Depends on algorithms that emit objective metric after each epoch
- Warm Start
 - Uses one or more previous tuning jobs as a starting point
 - Informs which hyperparameter combinations to search next
 - Can be a way to start where you left off from a stopped hyperparameter job
 - Two types: IDENTICAL_DATA_AND_ALGORITHM, TRANSFER_LEARNING
- Resource Limits
 - There are default limits for number of parallel tuning jobs, number of hyperparameters, number of training jobs per tuning job, etc.
 - Increasing these requires requesting a quota increase from AWS support



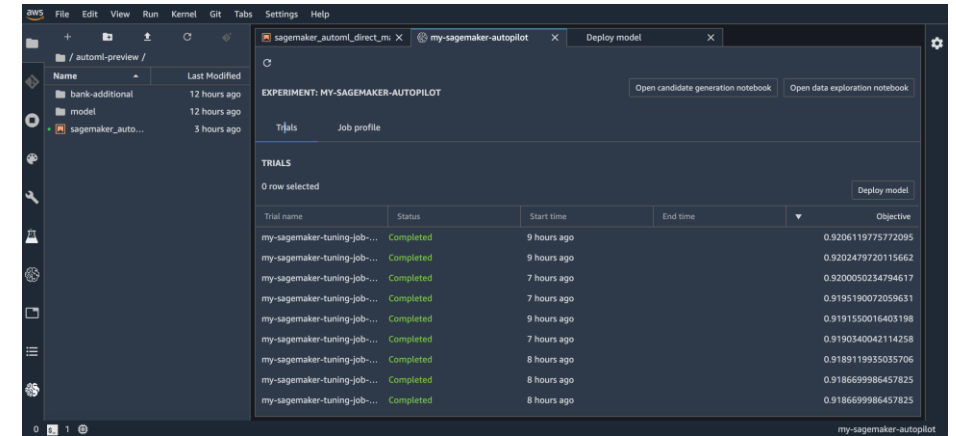
Hyperparameter Tuning Approaches

- Grid search
 - Limited to categorical parameters
 - Brute force; tries every possible combination (!)
- Random search
 - Chooses a random combination of hyperparameter values on each job
 - No dependence on prior runs, so they can all run in parallel
- Bayesian optimization
 - Treats tuning as a regression problem
 - Learns from each run to converge on optimal values
- Hyperband
 - Appropriate for algorithms that publish results iteratively (like training a neural network over several epochs)
 - Dynamically allocates resources, early stopping, parallel
 - Much faster than random search or Bayesian



SageMaker Autopilot

- Automates:
 - Algorithm selection
 - Data preprocessing
 - Model tuning
 - All infrastructure
- It does all the trial & error for you
- More broadly this is called AutoML



The screenshot shows the SageMaker Autopilot interface. On the left is a file explorer with folders like 'bank-additional', 'model', and 'sagemaker_auto...'. The main panel displays 'EXPERIMENT: MY-SAGEMAKER-AUTOPILOT' with buttons for 'Open candidate generation notebook' and 'Open data exploration notebook'. Below this is a 'TRIALS' section with a 'Deploy model' button. A table lists several completed trials with their names, statuses, start/end times, and objective values.

Trial name	Status	Start time	End time	Objective
my-sagemaker-tuning-job-...	Completed	9 hours ago		0.9206119775772095
my-sagemaker-tuning-job-...	Completed	9 hours ago		0.9202479720115662
my-sagemaker-tuning-job-...	Completed	7 hours ago		0.920050234794617
my-sagemaker-tuning-job-...	Completed	7 hours ago		0.9195190072059631
my-sagemaker-tuning-job-...	Completed	9 hours ago		0.9191550016403198
my-sagemaker-tuning-job-...	Completed	7 hours ago		0.9190340042114258
my-sagemaker-tuning-job-...	Completed	8 hours ago		0.9189119935035706
my-sagemaker-tuning-job-...	Completed	8 hours ago		0.9186699986457825
my-sagemaker-tuning-job-...	Completed	8 hours ago		0.9186699986457825

SageMaker Autopilot workflow

- Load data from S3 for training
- Select your target column for prediction
- Automatic model creation
- Model notebook is available for visibility & control
- Model leaderboard
 - Ranked list of recommended models
 - You can pick one
- Deploy & monitor the model, refine via notebook if needed

Do you want to run a complete experiment?

- ☒ Yes
- ☐ No, run a pilot to create a notebook with candidate definitions

SageMaker Autopilot

- Can add in human guidance
- With or without code in SageMaker Studio or AWS SDK's
- Problem types:
 - Binary classification
 - Multiclass classification
 - Regression
- Algorithm Types:
 - Linear Learner
 - XGBoost
 - Deep Learning (MLP's)
 - Ensemble mode
- Data must be tabular CSV or Parquet

Autopilot Training Modes

- HPO (Hyperparameter optimization)
 - Selects algorithms most relevant to your dataset
 - Linear Learner, XGBoost, or Deep Learning
 - Selects best range of hyperparameters to tune your models
 - Runs up to 100 trials to find optimal hyperparameters in the range
 - Bayesian optimization used if dataset < 100MB
 - Multi-fidelity optimization if > 100MB
 - Early stopping if a trial is performing poorly
- Ensembling
 - Trains several base models using AutoGluon library
 - Wider range of models, including more tree-based and neural network algorithms
 - Runs 10 trials with different model and parameter settings
 - Models are combined with a stacking ensemble method
- Auto
 - HPO if > 100MB
 - Ensembling if < 100MB
 - Autopilot needs to be able to read the size of your dataset, or will default to HPO
 - S3 bucket hidden inside a VPC
 - S3DataType is ManifestFile
 - S3Uri contains more than 1000 items



Autopilot Explainability

- Transparency on how models arrive at predictions
- Feature attribution
 - Uses SHAP Baselines / Shapley Values
 - Research from cooperative game theory
 - Assigns each feature an importance value for a given prediction

A Unified Approach to Interpreting Model Predictions

Scott M. Lundberg
Paul G. Allen School of Computer Science
University of Washington
Seattle, WA 98105
slund1@cs.washington.edu

Su-In Lee
Paul G. Allen School of Computer Science
Department of Genome Sciences
University of Washington
Seattle, WA 98105
suinlee@cs.washington.edu

Abstract

Understanding why a model makes a certain prediction can be as crucial as the prediction's accuracy in many applications. However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, such as ensemble or deep learning models, creating a tension between *accuracy* and *interpretability*. In response, various methods have recently been proposed to help users interpret the predictions of complex models, but it is often unclear how these methods are related and when one method is preferable over another. To address this problem, we present a unified framework for interpreting predictions, SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. Its novel components include: (1) the identification of a new class of additive feature importance measures, and (2) theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class unifies six existing methods, notable because several recent methods in the class lack the proposed desirable properties. Based on insights from this unification, we present new methods that show improved computational performance and/or better consistency with human intuition than previous approaches.

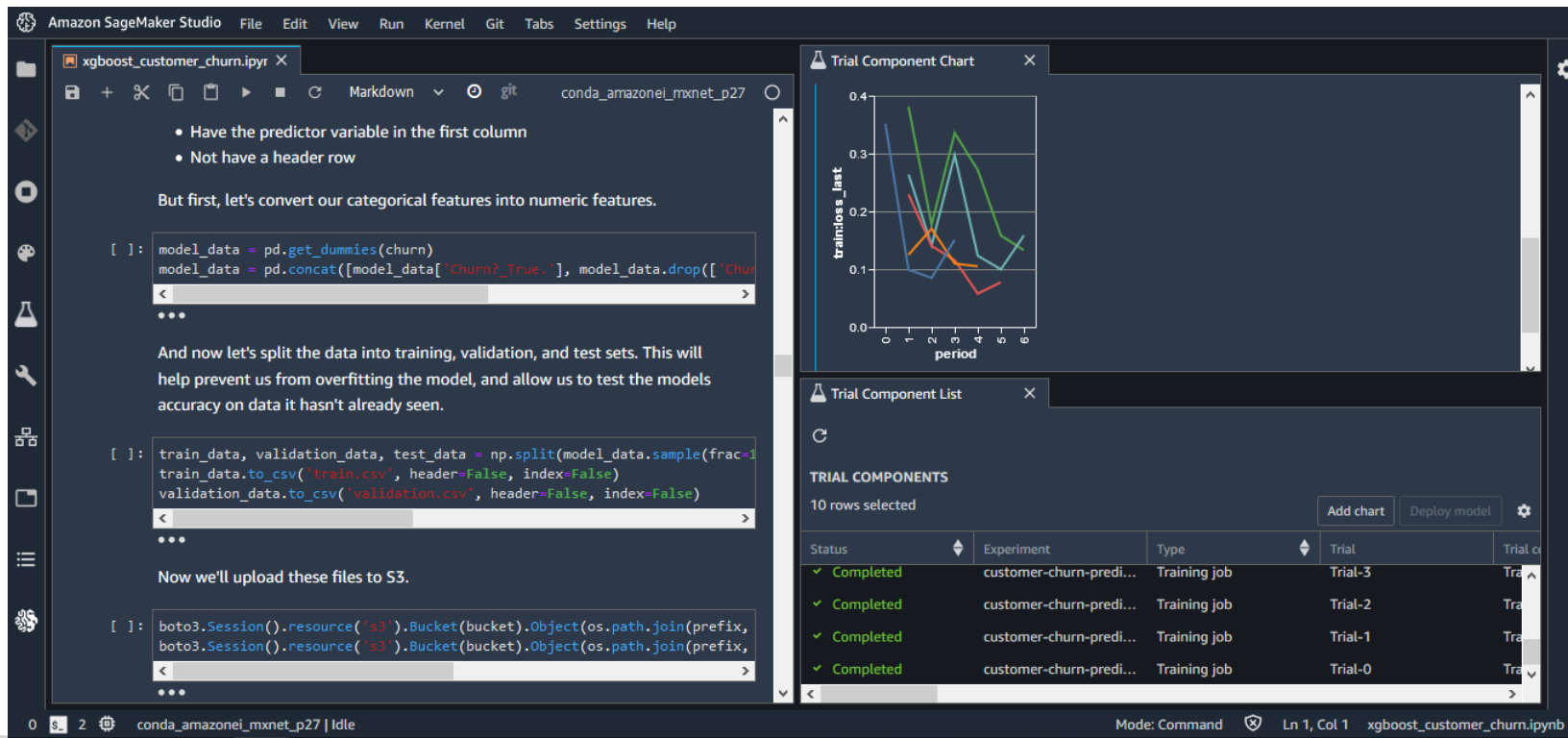
1 Introduction

The ability to correctly interpret a prediction model's output is extremely important. It engenders appropriate user trust, provides insight into how a model may be improved, and supports understanding of the process being modeled. In some applications, simple models (e.g., linear models) are often preferred for their ease of interpretation, even if they may be less accurate than complex ones. However, the growing availability of big data has increased the benefits of using complex models, so bringing to the forefront the trade-off between accuracy and interpretability of a model's output. A wide variety of different methods have been recently proposed to address this issue [5, 8, 9, 3, 4, 1]. But an understanding of how these methods relate and when one method is preferable to another is still lacking.

Here, we present a novel unified approach to interpreting model predictions.¹ Our approach leads to three potentially surprising results that bring clarity to the growing space of methods:

SageMaker Studio

- Visual IDE for machine learning!
- Integrates many of the features we're about to cover.



The screenshot displays the Amazon SageMaker Studio interface. The main window shows a Jupyter notebook titled 'xgboost_customer_churn.ipynb' with the following content:

- Have the predictor variable in the first column
- Not have a header row

But first, let's convert our categorical features into numeric features.

```
[ ]: model_data = pd.get_dummies(churn)
model_data = pd.concat([model_data['Churn?_True'], model_data.drop(['Churn?_True', 'Churn?_False'], axis=1)], axis=1)
...
```

And now let's split the data into training, validation, and test sets. This will help prevent us from overfitting the model, and allow us to test the models accuracy on data it hasn't already seen.

```
[ ]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1, random_state=42), [int(0.33 * len(model_data)), int(0.33 * len(model_data))])
train_data.to_csv('train.csv', header=False, index=False)
validation_data.to_csv('validation.csv', header=False, index=False)
...
```

Now we'll upload these files to S3.

```
[ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train.csv')).upload_file(train_data.to_csv('train.csv', header=False, index=False))
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'validation.csv')).upload_file(validation_data.to_csv('validation.csv', header=False, index=False))
...
```

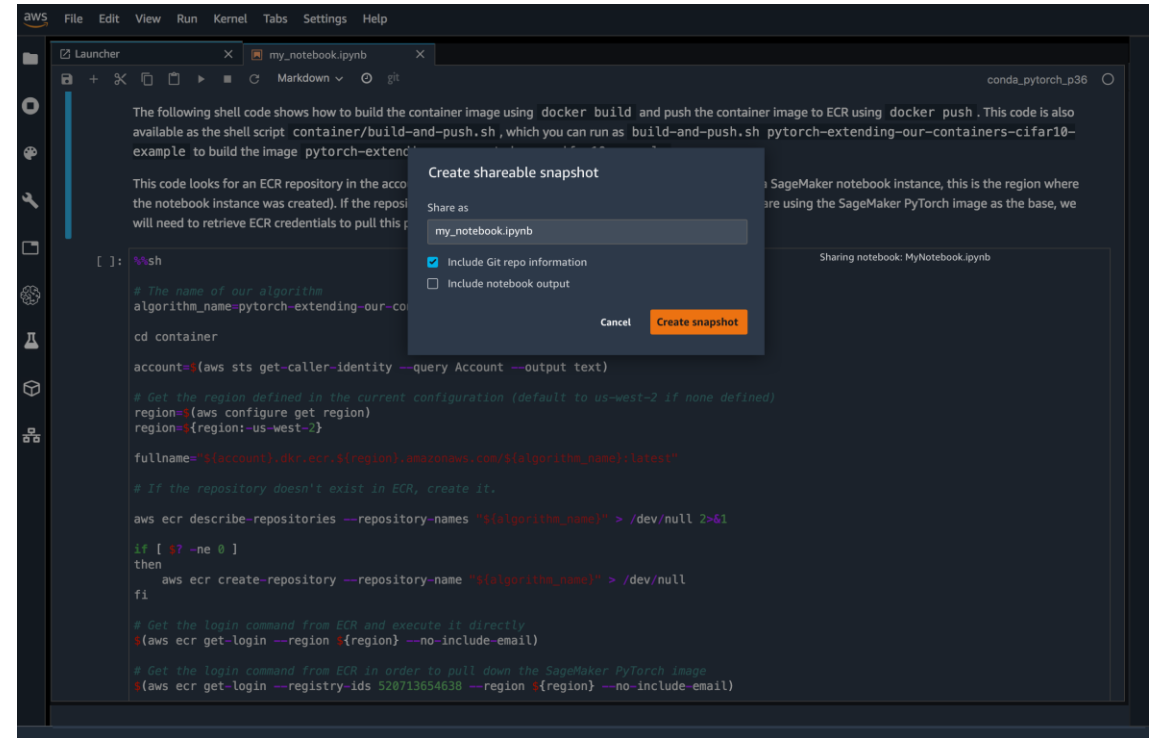
The right sidebar contains two panels:

- Trial Component Chart:** A line chart showing 'trainloss_last' (y-axis, 0.0 to 0.4) versus 'period' (x-axis, 0 to 10). The chart displays several lines representing different trials, with Trial-3 showing the highest loss and Trial-0 showing the lowest loss.
- Trial Component List:** A table listing trial components. The table has columns for Status, Experiment, Type, Trial, and Trial ID. The table shows 10 rows selected, with the first four rows being 'Completed' training jobs for 'customer-churn-predi...'.

Status	Experiment	Type	Trial	Trial ID
✓ Completed	customer-churn-predi...	Training job	Trial-3	Tra...
✓ Completed	customer-churn-predi...	Training job	Trial-2	Tra...
✓ Completed	customer-churn-predi...	Training job	Trial-1	Tra...
✓ Completed	customer-churn-predi...	Training job	Trial-0	Tra...

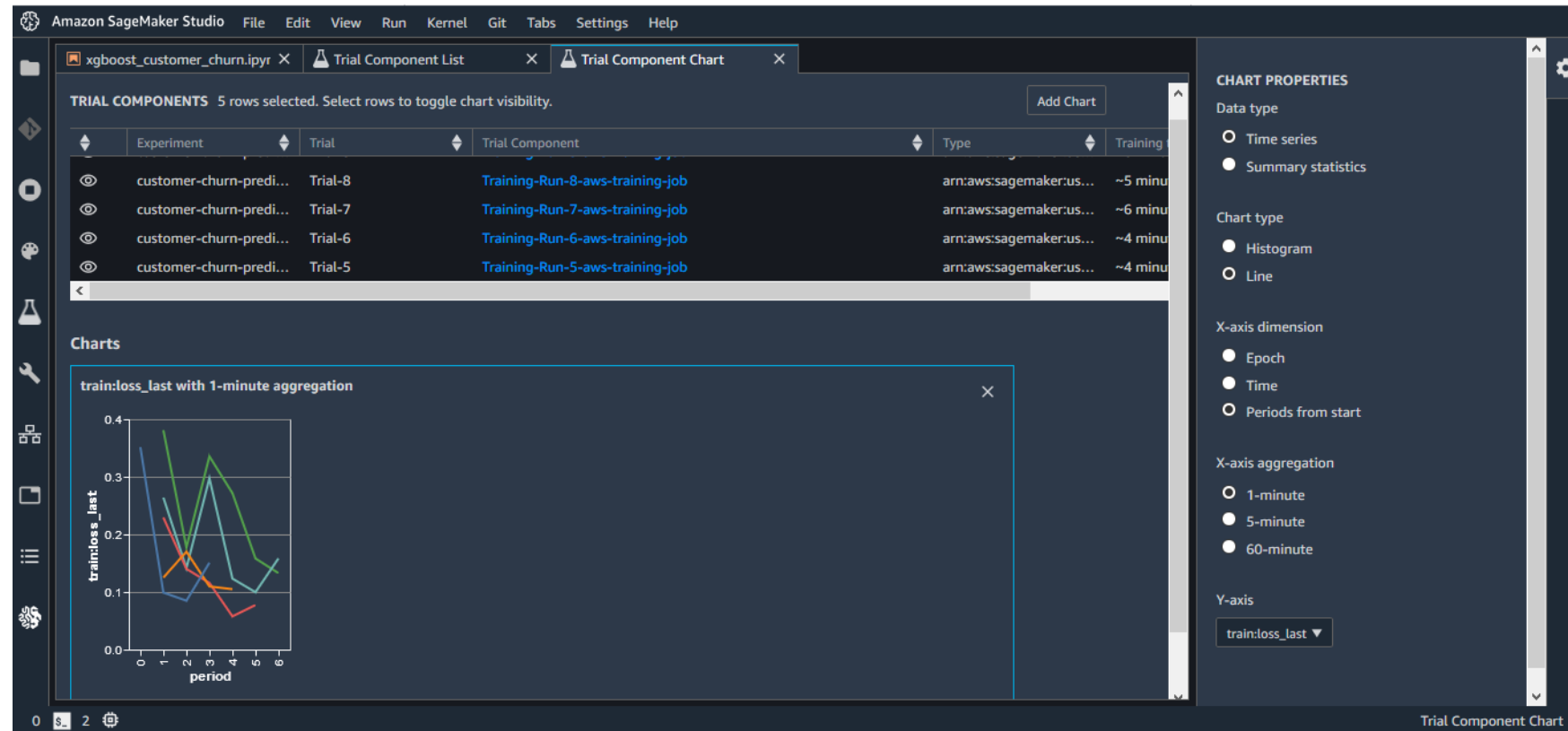
SageMaker Notebooks

- Create and share Jupyter notebooks with SageMaker Studio
- Switch between hardware configurations (no infrastructure to manage)



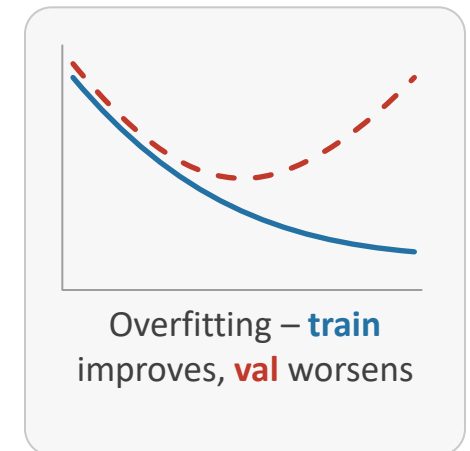
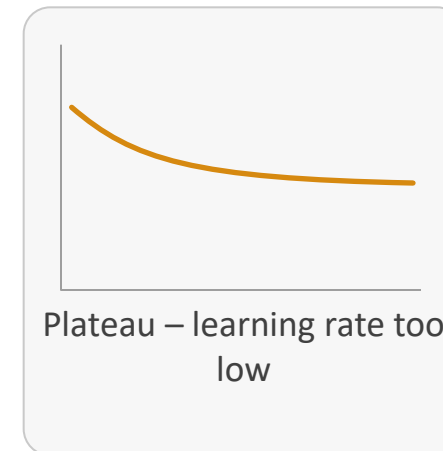
SageMaker Experiments

- Organize, capture, compare, and search your ML jobs



Debugging Model Convergence and Training Performance

- Loss doesn't decrease, or plateaus too early
 - Learning rate too low, vanishing gradients, underfitting, bad training data
- Loss oscillates wildly or diverges
 - Learning rate too high, batch size too small, unstable optimization
- Loss becomes NaN or infinite
 - Exploding gradients, poor normalization or preprocessing
- Validation stalls while training keeps improving
 - Overfitting – regularize, simplify, or add data
- “Identical” runs behave very differently
 - Unstable optimization, hyperparameter or data differences
- Low or erratic CPU / GPU utilization
 - Data loading bottlenecks; CPU, GPU, memory, or disk starvation; wrong batch size



Debugging Model Convergence and Training Performance

- SageMaker Debugger is now in maintenance mode – its old responsibilities are covered by current tools:

SageMaker AI with MLflow

Experiment tracking

- Log loss and other scalar metrics per step
- Log parameters and hyperparameters
- Compare runs side by side to spot regressions and improvements
- Track artifacts and model lineage

TensorBoard on SageMaker AI

Model internals

- Visualize training and validation curves
- Inspect gradient distributions and weight histograms
- Inspect activations and deeper model-training behavior

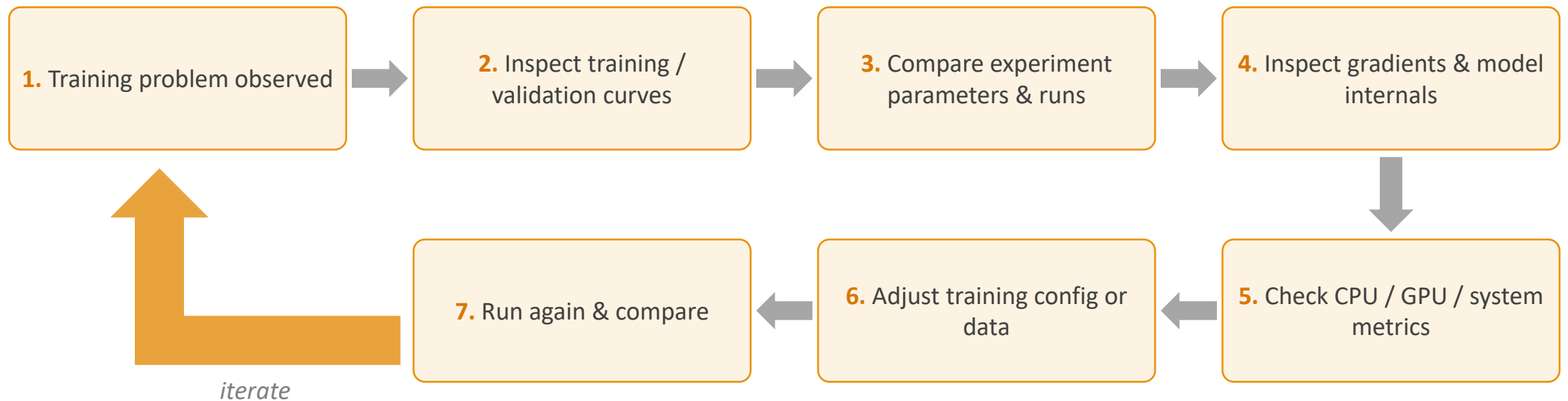
Amazon CloudWatch

Infrastructure & system metrics

- CPU, GPU, memory, and disk utilization
- Training and infrastructure metrics
- Dashboards and alarms
- Spot resource bottlenecks and anomalous behavior

Debugging Model Convergence and Training Performance

- A diagnostic workflow: observe, diagnose, adjust – and repeat until training is healthy



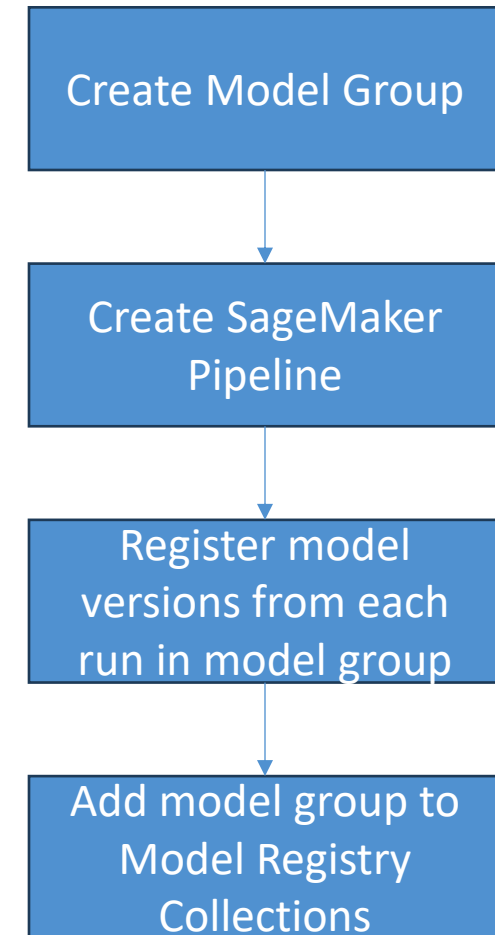
MLflow – runs & parameters

TensorBoard – curves & model internals

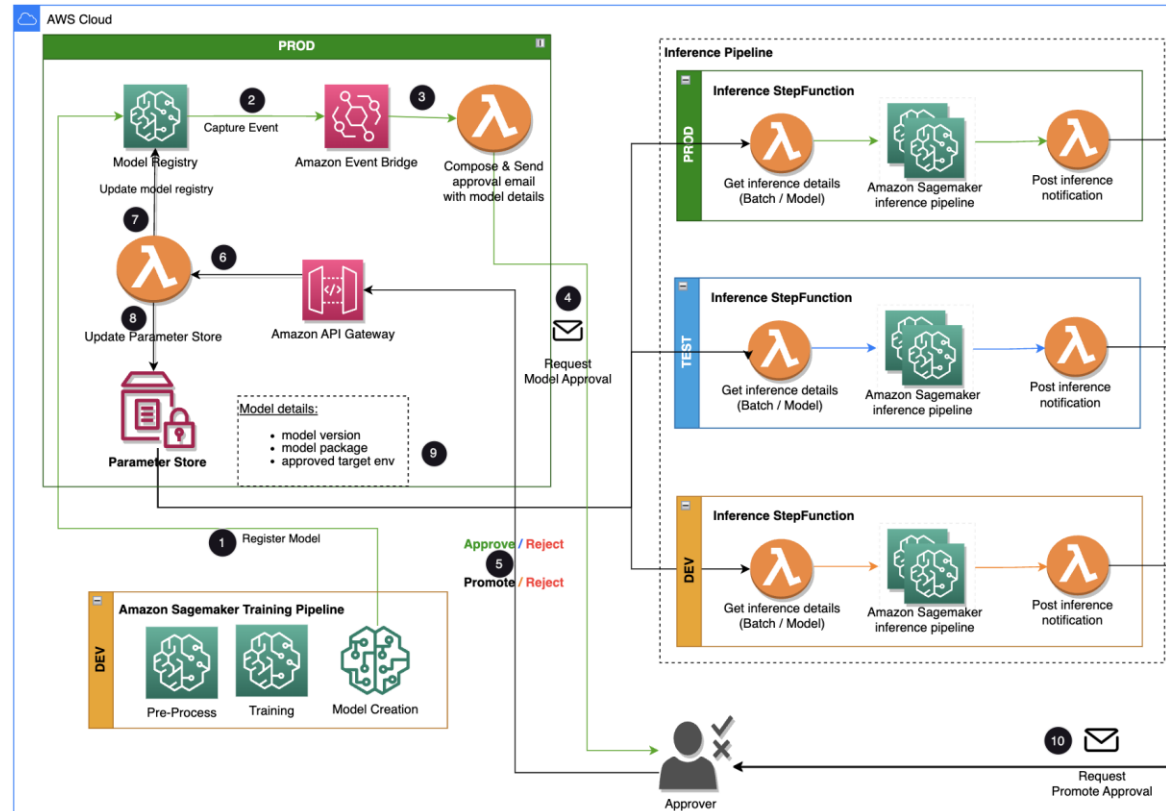
CloudWatch – system metrics

SageMaker Model Registry

- Catalog your models, manage model versions
- Associate metadata with models
- Manage approval status of a model
- Deploy models to production
- Automate deployment with CI/CD
- Share models
- Integrate with SageMaker Model



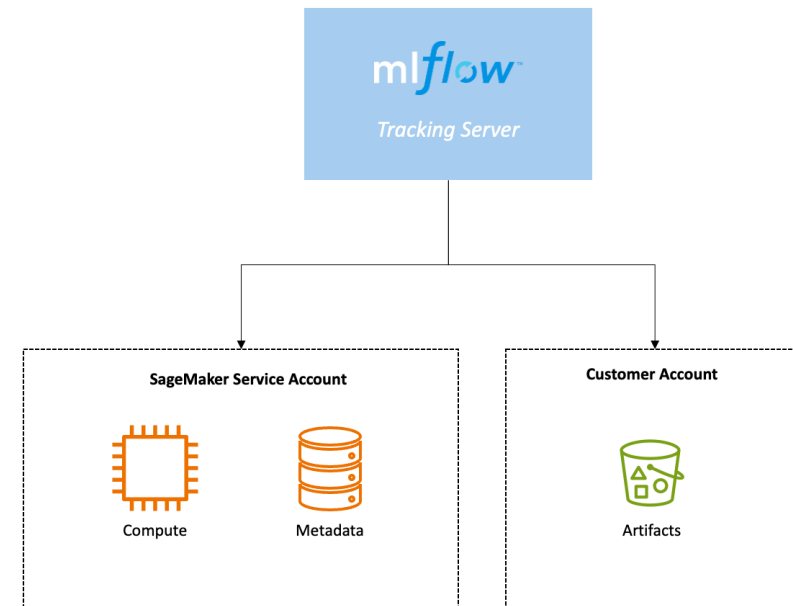
Example workflow for approving and promoting models with Model Registry



<https://aws.amazon.com/blogs/machine-learning/build-an-amazon-sagemaker-model-registry-approval-and-promotion-workflow-with-human-intervention/>

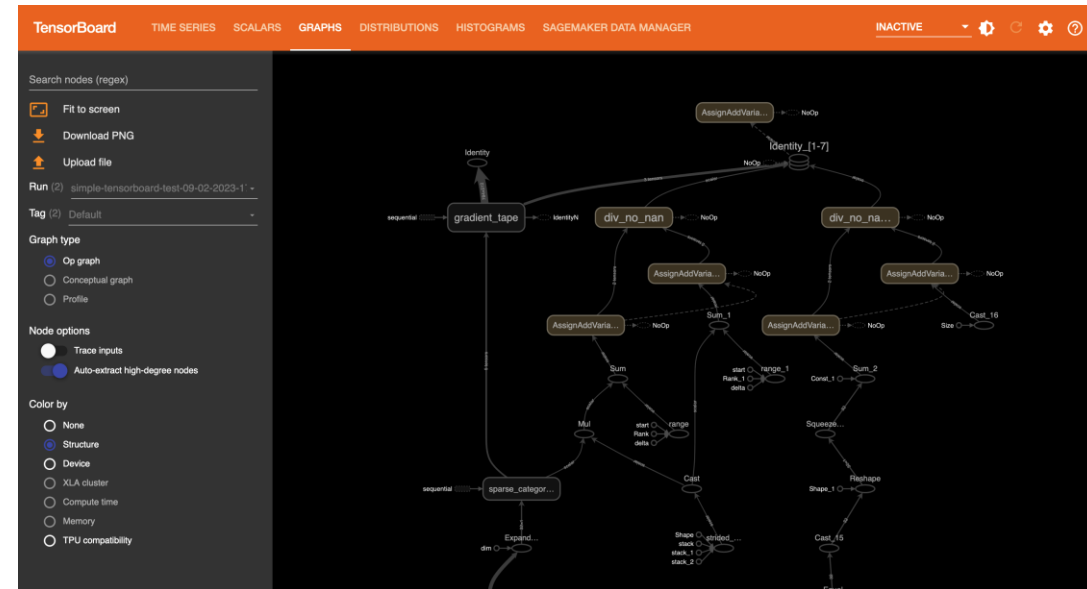
MLFlow

- Open-source platform for ML / GenAI workflows
- Offers observability, evaluations, tracking, tracing, AI gateway
- Model management and deployment
- MLFlow is integrated into Amazon SageMaker AI
 - Managed service
 - Integrates with model development, management, deployment, and tracking
 - Integration points:
 - SageMaker Studio
 - SageMaker Model Registry
 - SageMaker AI Inference
 - IAM
 - CloudTrail
 - EventBridge



Analyzing Training Jobs with Tensorboard

- TensorBoard is a visualization toolkit for Tensorflow or PyTorch
 - Visualize loss and accuracy
 - Visualize model graph
 - View histograms of weight, biases over time
 - Project embeddings to lower dimensions
 - Profiling
- Integrated with SageMaker console or via URL
- Requires modifications to your training script
 - <https://docs.aws.amazon.com/sagemaker/latest/dg/tensorboard-on-sagemaker.html>



SageMaker Training Techniques

SageMaker Training Compiler

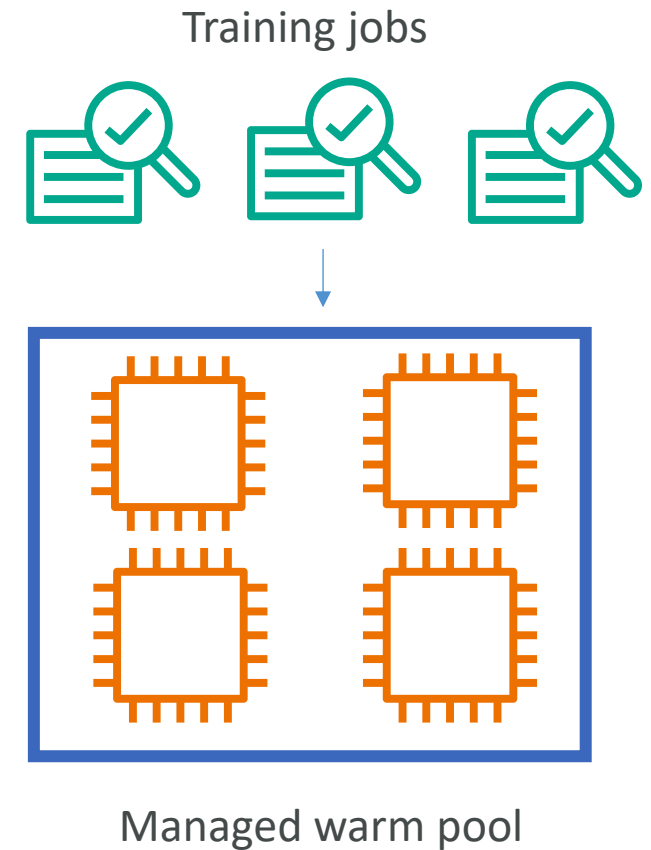
- Integrated into AWS Deep Learning Containers (DLCs)
 - Can't bring your own container
 - DLC's are pre-made Docker images for:
 - Tensorflow, PyTorch, MXNet
- Just use `compiler_config=TrainingCompilerConfig()` in your estimator class
- Compile & optimize training jobs on GPU instances
- Can accelerate training up to 50%
- Converts models into hardware-optimized instructions
- Tested with Hugging Face transformers library, or bring your own model
- Incompatible with SageMaker distributed training libraries
- Best practices:
 - Ensure GPU instances are used (ml.p3, ml.p4, ml.g4dn, ml.g5)
 - PyTorch models must use PyTorch/XLA's model save function
 - Enable debug flag in `compiler_config` parameter to enable debugging
- No longer maintained ☹️



**SageMaker
Training Compiler**
Automatically optimizes
training job

Warm Pools

- Retain and re-use provisioned infrastructure
- Useful if repeatedly training a model to speed things up
- Use by setting *KeepAlivePeriodInSeconds* in your training job's resource config
- Requires a service limit increase request
- Use a *persistent cache* to store data across training jobs, to reduce costs



Warm Pools: Where it Lives

Amazon SageMaker > Training jobs > Create training job

Create training job

When you create a training job, Amazon SageMaker sets up the distributed compute cluster, performs the training, and deletes the cluster when training has completed. The resulting model artifacts are stored in the location you specified when you created the training job. [Learn more](#)

Job settings

...

Resource configuration

Instance type	Instance count	Additional storage volume per instance (GB)
ml.m4.xlarge ▼	1	1

Keep alive period
Use [SageMaker Training Managed Warm Pools](#)
 minutes ▼
Maximum runtime is 1 hour (60 minutes or 3600 seconds).

Encryption key - *optional*
Encrypt your data. Choose an existing KMS key or enter a key's ARN.
No Custom Encryption ▼

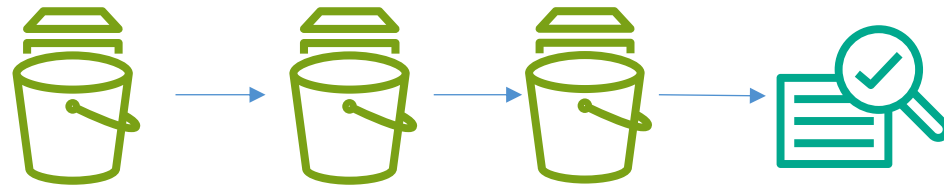
Stopping condition

Maximum runtime
 hours ▼
24



Checkpointing

- Creates snapshots during your training
 - You can re-start from these points if necessary
 - Or use them for troubleshooting, to analyze the model at different points
 - Automatic synchronization with S3 (from /opt/ml/checkpoint)
- To use, define **checkpoint_s3_uri** and **checkpoint_local_path** in your SageMaker estimator.



Checkpointing: Where it Lives

[Amazon SageMaker](#) > [Training jobs](#) > [Create training job](#)

Create training job

When you create a training job, Amazon SageMaker sets up the distributed compute cluster, performs the training, and deletes the cluster when training has completed. The resulting model artifacts are stored in the location you specified when you created the training job. [Learn more](#) 

Job settings

...

Checkpoint configuration - optional

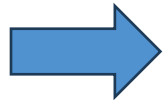
The algorithm is responsible for periodically generating checkpoints. The checkpoints are saved to this location and used to resume managed spot training jobs. [Learn more](#)

S3 location

To find a path, [go to Amazon S3](#)

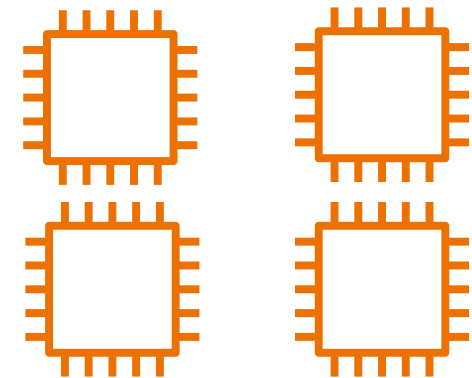
Local path - optional

If the algorithm provides checkpoints, this is the local location the checkpoints are written to. If you do not specify a location, the checkpoints are written to `/opt/ml/checkpoints`.



Cluster Health Checks and Automatic Restarts

- Run automatically when using ml.g or ml.p instance types
- Replaces any faulty instances
- Runs GPU health checks
- Ensures NVidia Collective Communication Library is working
- SageMaker internal service errors will result in an automatic restart of your training job
 - Replaces bad instances
 - Restarts healthy ones
 - Restarts the job



Distributed Training

- You can of course run multiple training jobs in parallel
 - “job parallelism”
- Individual training can also be parallelized
 - Distributed data parallelism
 - Distributed model parallelism
- Use larger instance types before multiple, parallel instances
 - i.e., a ml.p4d.24xlarge gives you 8 GPU's.
 - Max that out before moving to two 8-GPU instances.
 - Did I mention this is expensive?



Sagemaker's Distributed Training Libraries

- Built on the AWS Custom Collective Library for EC2
- Solves a similar problem as MapReduce / Spark
 - But for distributing computation of gradients in gradient descent
- **AllReduce** collective
 - Distributes computation of gradient updates to/from GPU's
 - Implemented in the **SageMaker Distributed Data Parallelism Library**
 - Specify a backend of **smdpp** to **torch.distributed.init_process_group** in your training scripts
 - Then specify `distribution={ "smdistributed": { "dataparallel": { "enabled": True } } }` in your PyTorch estimator.
- **AllGather** collective
 - Manages communication between nodes to improve performance
 - Offloads communications overhead to the CPU, freeing up GPU's.



Other Distributed Training Libraries

- You don't have to use what SageMaker provides
- PyTorch DistributedDataParallel (DDP)
 - `distribution={"pytorchddp": {"enabled": True}}`
- `torchrun`
 - `distribution={"torch_distributed": {"enabled": True}}`
 - Requires p3, p4, or trn1 instances
- `mpirun`
- DeepSpeed
 - Open source from Microsoft
 - For PyTorch
- Horovod

SageMaker Model Parallelism Library

- A Large Language Model won't fit on a single machine
 - Need to distribute the model itself to overcome GPU memory limits
 - Or you can use extra GPU memory to increase batch size
- SageMaker's interleaved pipelines offers some benefits
 - For both Tensorflow and PyTorch
- SageMaker MPP goes further
 - PyTorch only, though
 - Optimization state sharding
 - "Optimization state" is just its weights
 - Requires a stateful optimizer (adam, fp16)
 - Generally useful for >1B parameters
 - Activation checkpointing
 - Reduces memory usage by clearing activations of certain layers and recomputing them during a backward pass
 - Saves memory at expense of computation
 - Activation offloading
 - Swaps checkpointed activations in a microbatch to/from the CPU
- `import torch.sagemaker as tsm \ tsm.init()`
 - Requires a few modifications to your training job launcher object
 - Wrap your model and optimizer, split up your data set
 - Train with `mpi` and `mpp` in your estimator

```
estimator = PyTorch(
    framework_version=2.2.0,
    py_version="310"
    # image_uri="<smp-docker-image-uri>" # For using prior versions,
    # specify the SMP image URI directly.
    entry_point="your-training-script.py", # Pass the training script you
    adapted with SMP from Step 1.
    ... # Configure other required and optional parameters
    distribution={
        "torch_distributed": {"enabled": True},
        "smdistributed": {
            "modelparallel": {
                "enabled": True,
                "parameters": {
                    "hybrid_shard_degree": Integer,
                    "sm_activation_offloading": Boolean,
                    "activation_loading_horizon": Integer,
                    "fsdp_cache_flush_warnings": Boolean,
                    "allow_empty_shards": Boolean,
                    "tensor_parallel_degree": Integer,
                    "expert_parallel_degree": Integer,
                    "random_seed": Integer
                }
            }
        }
    }
)
```

Sharded Data Parallelism

- Combines parallel data and models
- Parallel models means optimizer states are sharded
 - Into *sharding groups*
- Sharded Data Parallelism:
 - Also shards the trainable parameters
 - And the associated gradients
 - ...across those optimizer sharding group GPU's
- This is implemented in the SageMaker Model Parallel Library.
- And MPP is there by default in a Deep Learning Container for PyTorch

Elastic Fabric Adapter (EFA)

- Network device attached to your SageMaker instances
- Makes better use of your bandwidth
 - Promises performance of an on-premises High Performance Computing (HPC) cluster in the cloud
- Use with NCCL (NVIDIA Collective Communication Library)
 - Requires NVIDIA GPU's of course
- To use EFA:
 - Include NCCL, EFA, and the AWS OFI NCCL plugin in your container
 - Set several environment variables (such as FI_PROVIDER="efa")

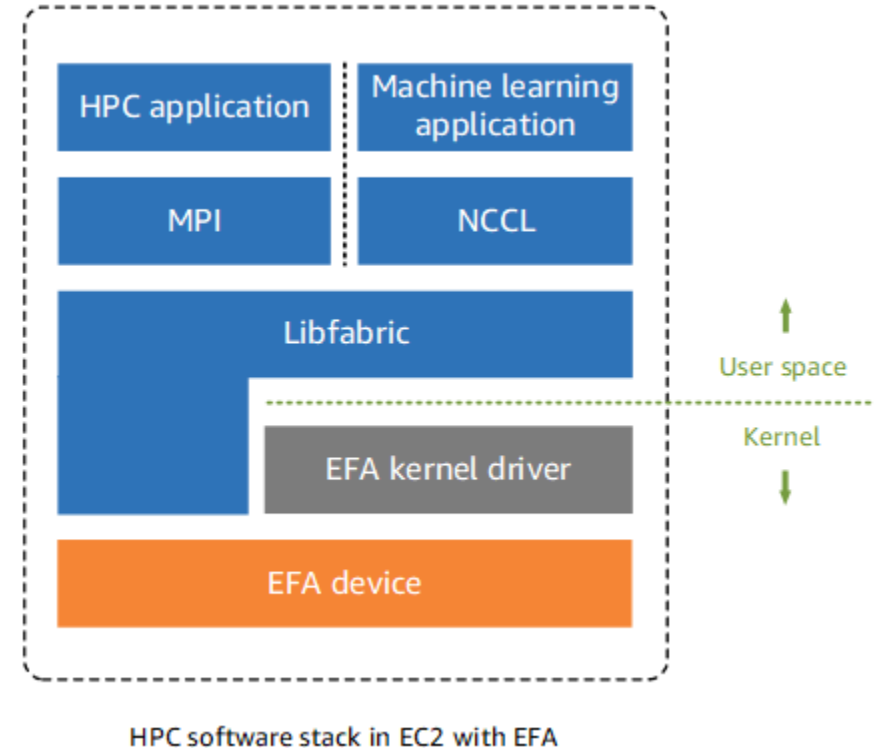


Image: AWS (<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/efa.html>)

MiCS

- What if billions of parameters aren't enough?
We want trillions!
- Amazon came up with “Minimize the Communication Scale” (MiCS)
- This is basically another name for what the SageMaker sharded data parallelism provides
- You just need to know all this distributed training stuff is what enables training models with >1T parameters
- Bigger instances helps too
 - Minimizes communication overhead
 - EC2 P4de GPU instances
 - 400 Gbps networking
 - 80GB GPU memory
- Did I mention this isn't cheap?

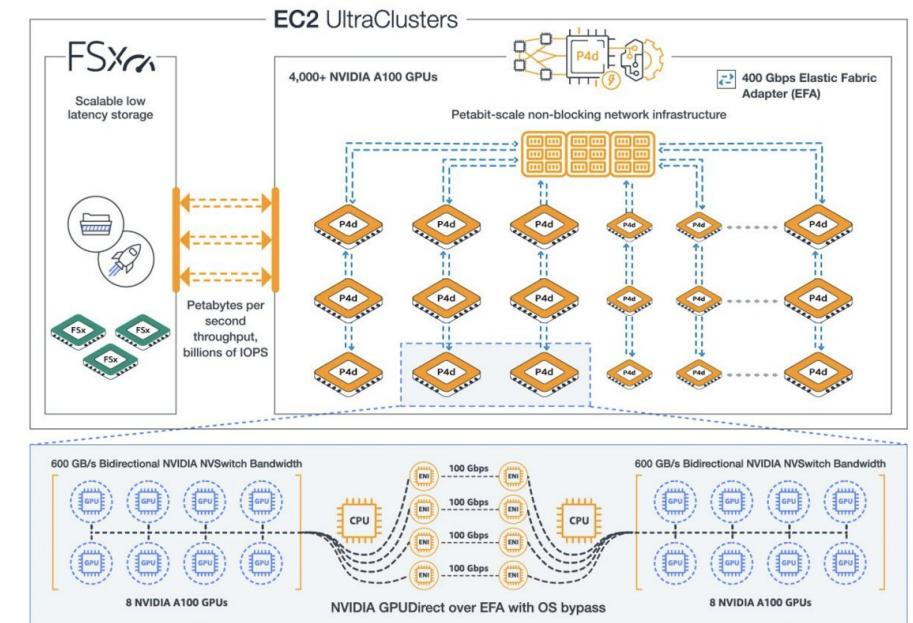


Image: AWS (<https://www.amazon.science/blog/scaling-to-trillion-parameter-model-training-on-aws>)

Transformers and Generative AI

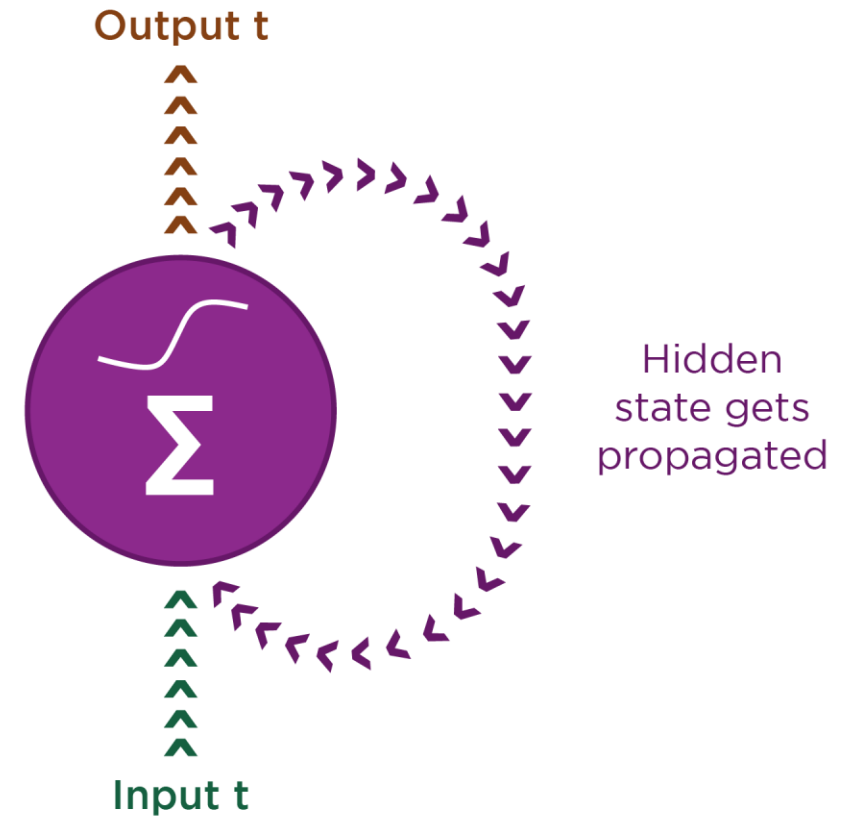
Large language models, transformers, GPT, and AWS support

The Transformer Architecture

And how self-attention works its magic

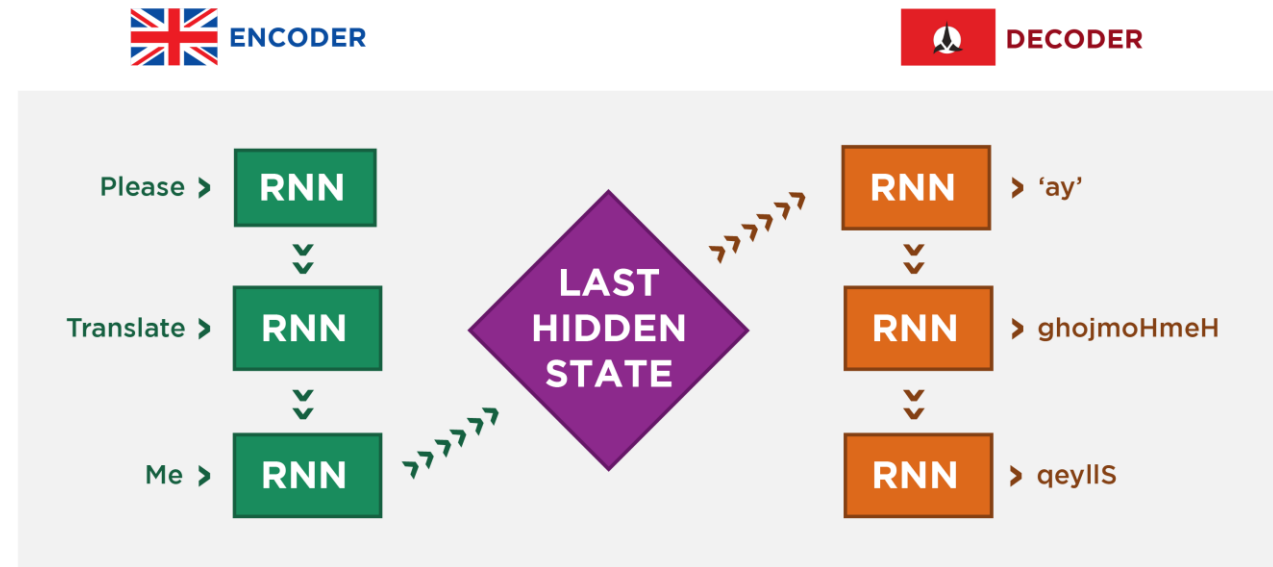
The Evolution of Transformers

- RNN's, LSTMs
- Introduced a feedback loop for propagating information forward
- Useful for modeling sequential things
 - Time series
 - Language! A sequence of words (or tokens)



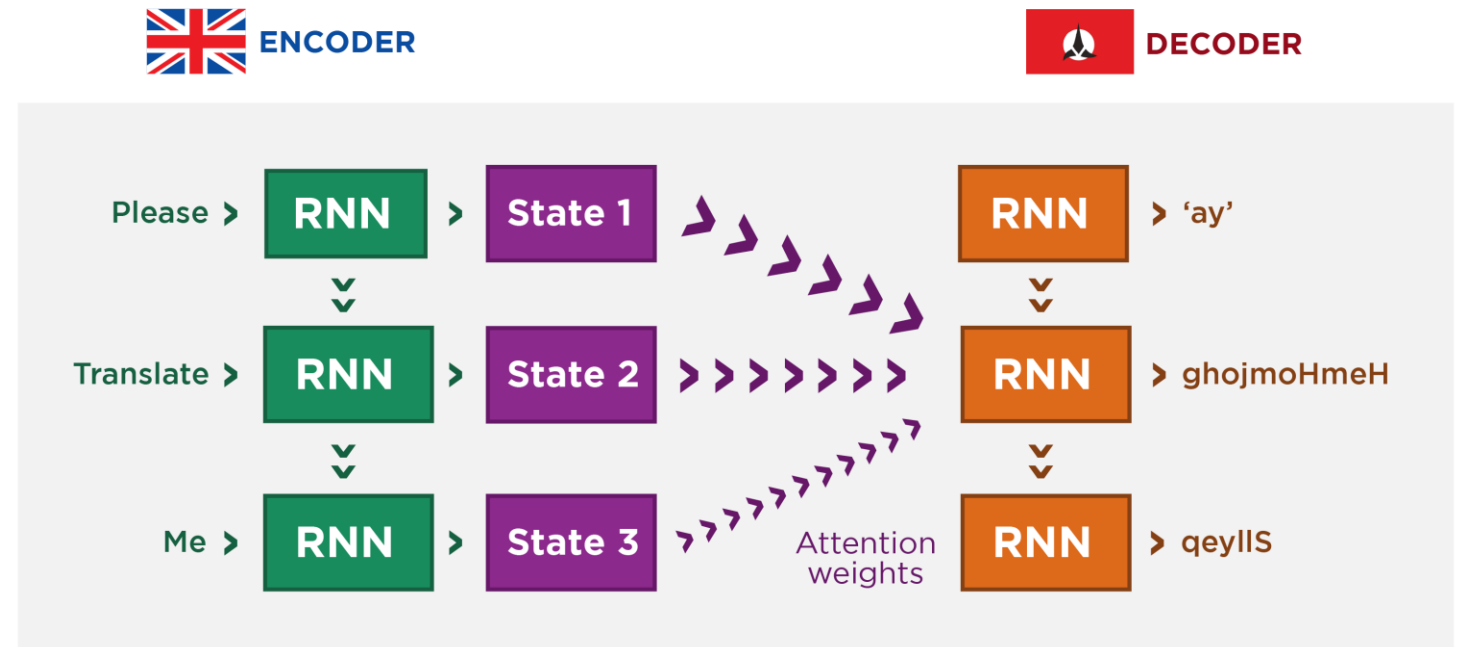
The Evolution of Transformers

- Machine translation
- Encoder / Decoder architecture
- Encoders and Decoders are RNN's
- But, the one vector tying them together creates an information bottleneck
 - Information from the start of the sequence may be lost



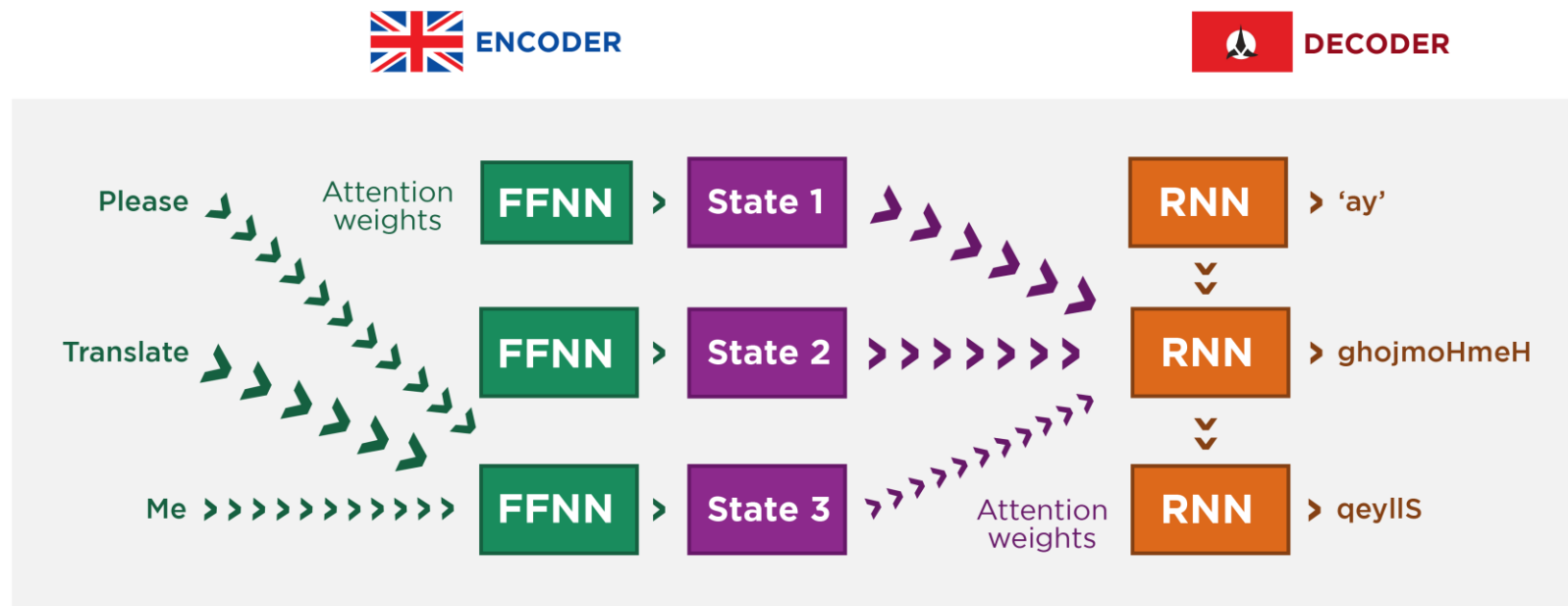
“Attention is all you need”

- A hidden state for each step (token)
- Deals better with differences in word order
- Starts to have a concept of relationships between words
- But RNN's are still sequential in nature, can't parallelize it



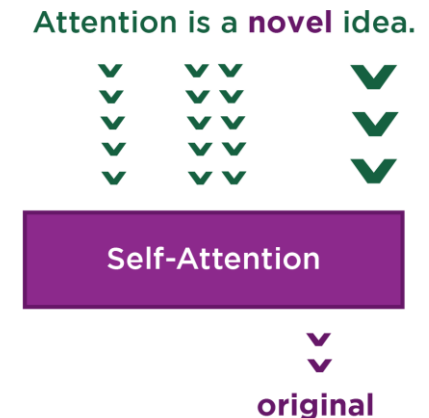
Transformers

- Ditch RNN's for feed-forward neural networks (FFNN's)
- Use “self-attention”
- This makes it parallelizable (so can train on much more data)



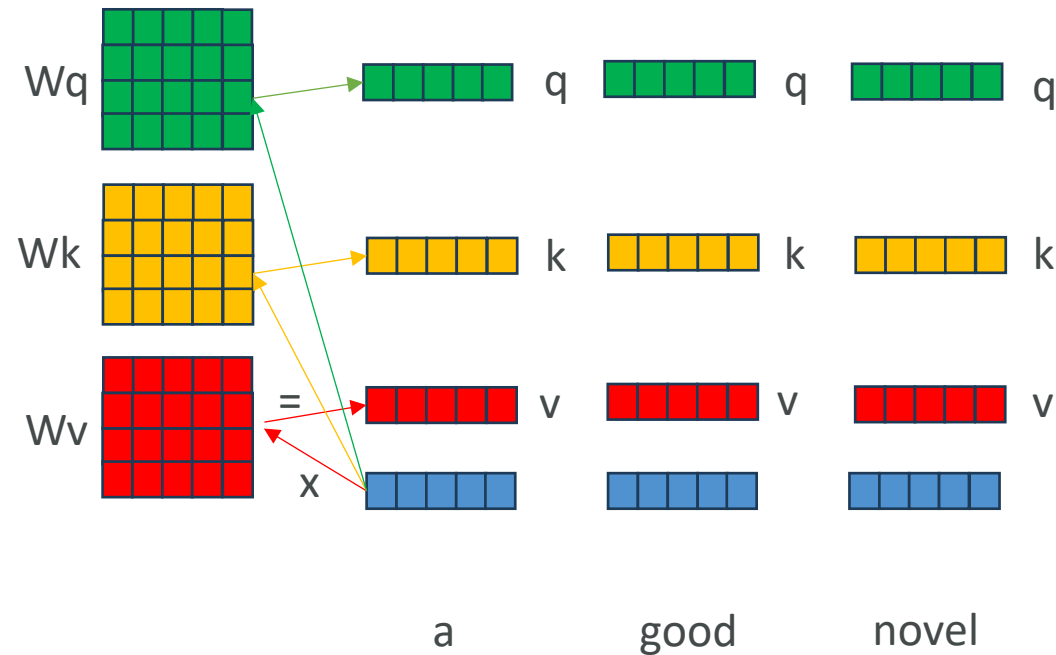
Self-Attention (in more depth)

- Each encoder or decoder has a list of embeddings (vectors) for each token
- Self-attention produces a *weighted average* of all token embeddings. The magic is in computing the attention weights.
- This results in tokens being tied to other tokens that are important for its context, and a new embedding that captures its “meaning” in context.



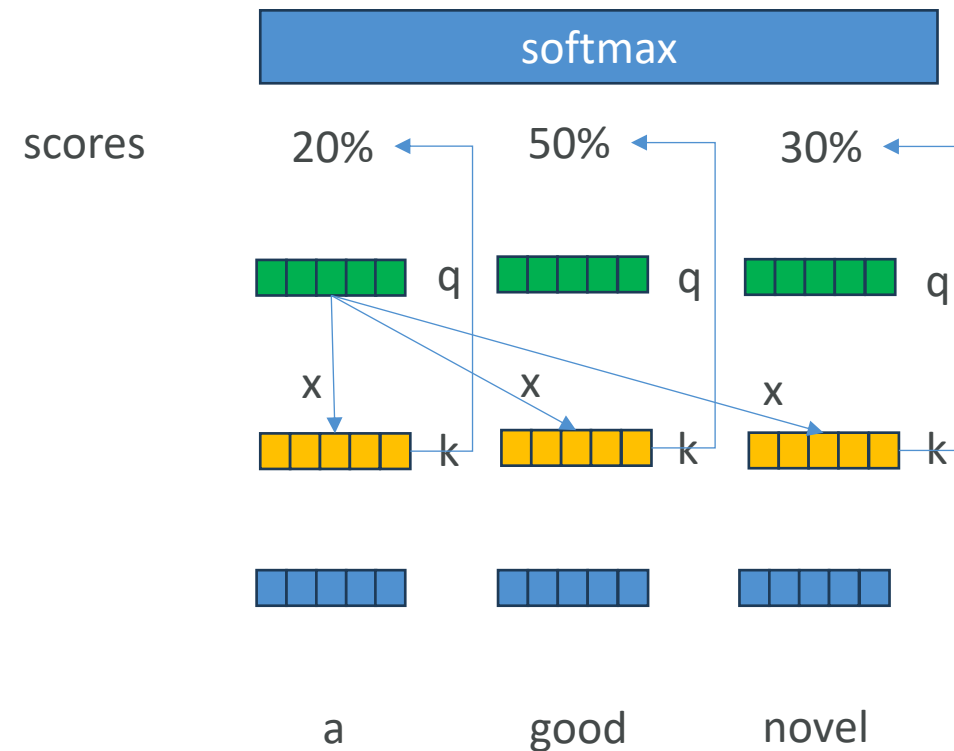
Self-Attention (in more depth)

- Three matrices of weights are learned through back-propagation
 - Query (W_q)
 - Key (W_k)
 - Value (W_v)
- Every token gets a query (q), key (k), and value (v) vector by multiplying its embedding against these matrices



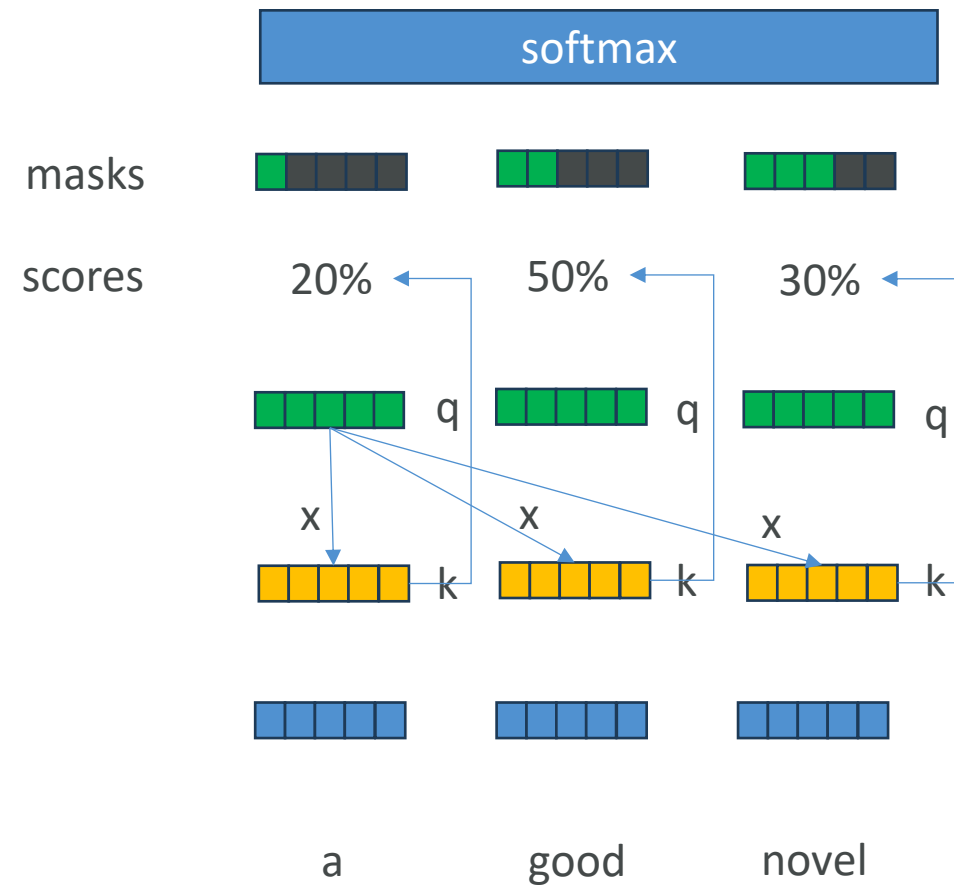
Self-Attention (in more depth)

- Compute a *score* for each token by multiplying (dot product) its query with each key
- “Scaled dot-product attention”
- Dot product is just one *similarity function* we can use.
- In practice, softmax is then applied to the scores to normalize them.



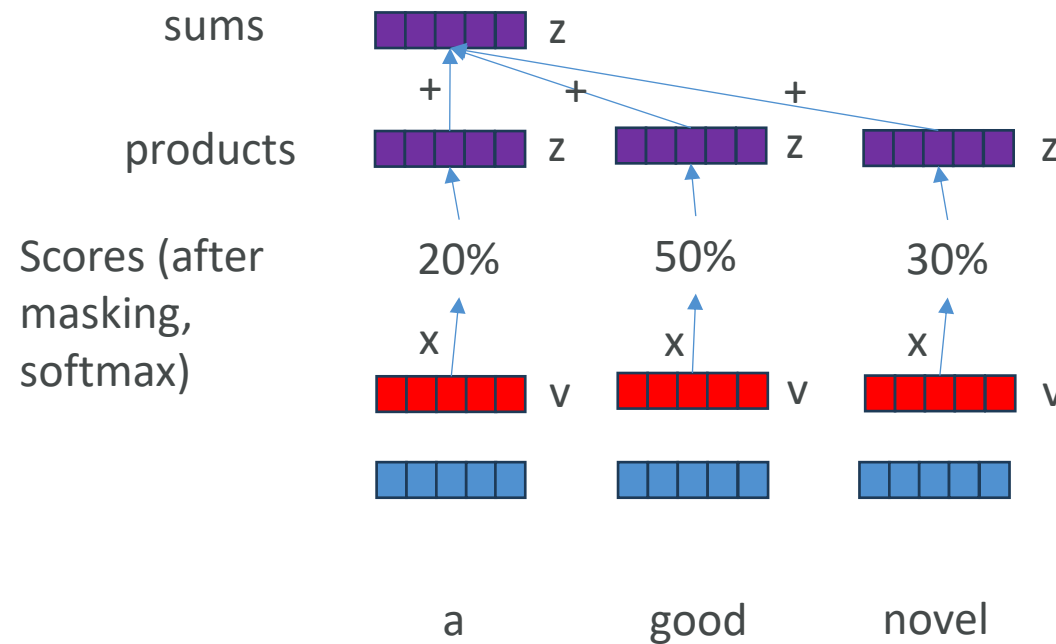
Masked Self-Attention

- A mask can be applied to prevent tokens from “peeking” into future tokens (words)
- GPT does this, but BERT does something else (masked language modeling)
- In this example, “good” wouldn’t be affected by “novel”, but “novel” could be affected by “good”.
- This is just the concept... actual implementation details will vary.



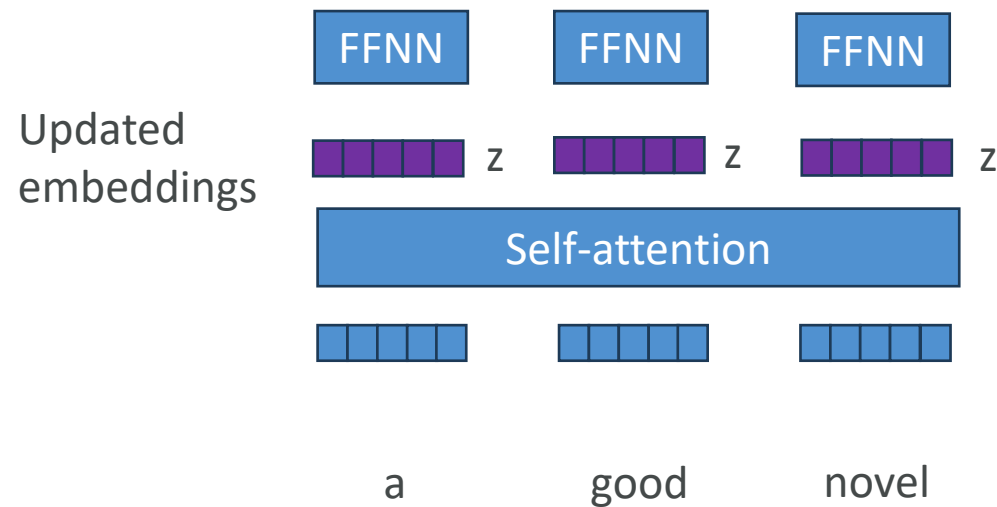
Self-Attention (in more depth)

- Now we sum.
- Multiply values by scores, and sum them up.



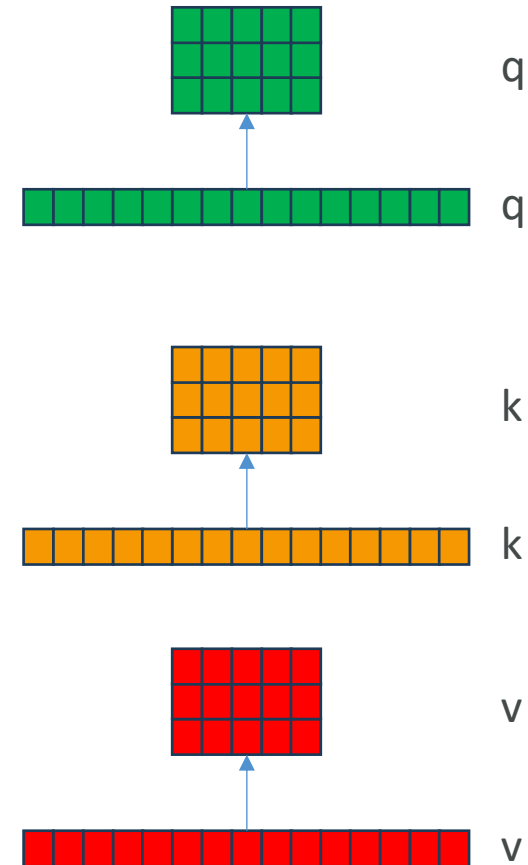
Self-Attention (in more depth)

- Repeat entire process for each token (in parallel)
- Now we have our updated embeddings for each token!
- These weight each token embedding as it's passed into the feed-forward neural network.



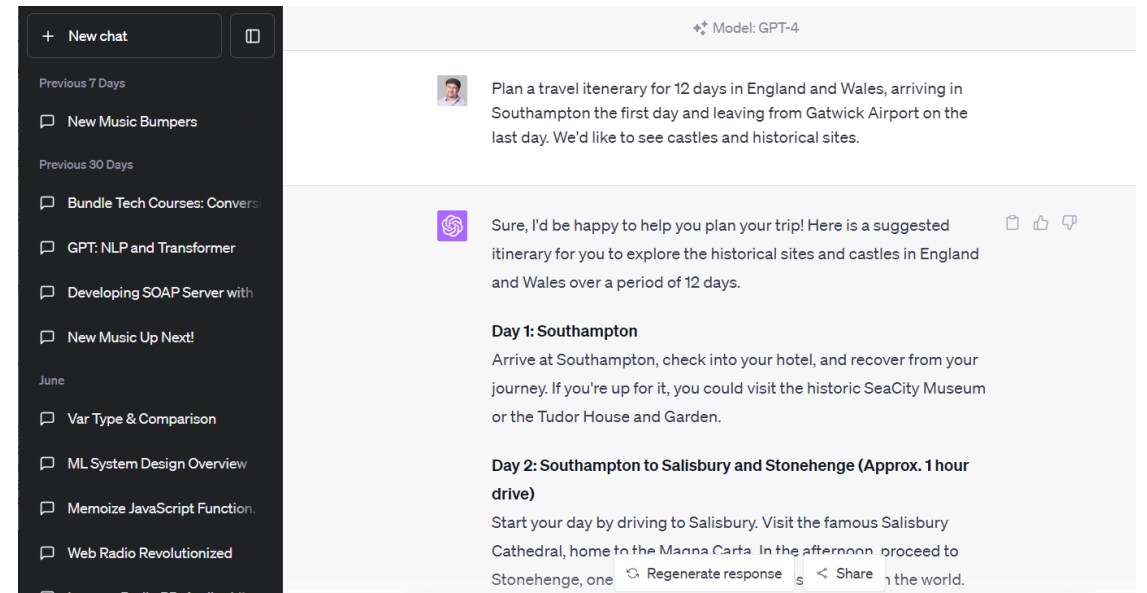
Multi-Headed Self-Attention

- The q , k , and v vectors are reshaped into matrices
- Then each row of the matrix can be processed in parallel
- The number of rows are the number of “heads”



Applications of Transformers

- Chat!
- Question answering
- Text classification
 - i.e., sentiment analysis
- Named entity recognition
- Summarization
- Translation
- Code generation
- Text generation
 - i.e., automated customer service

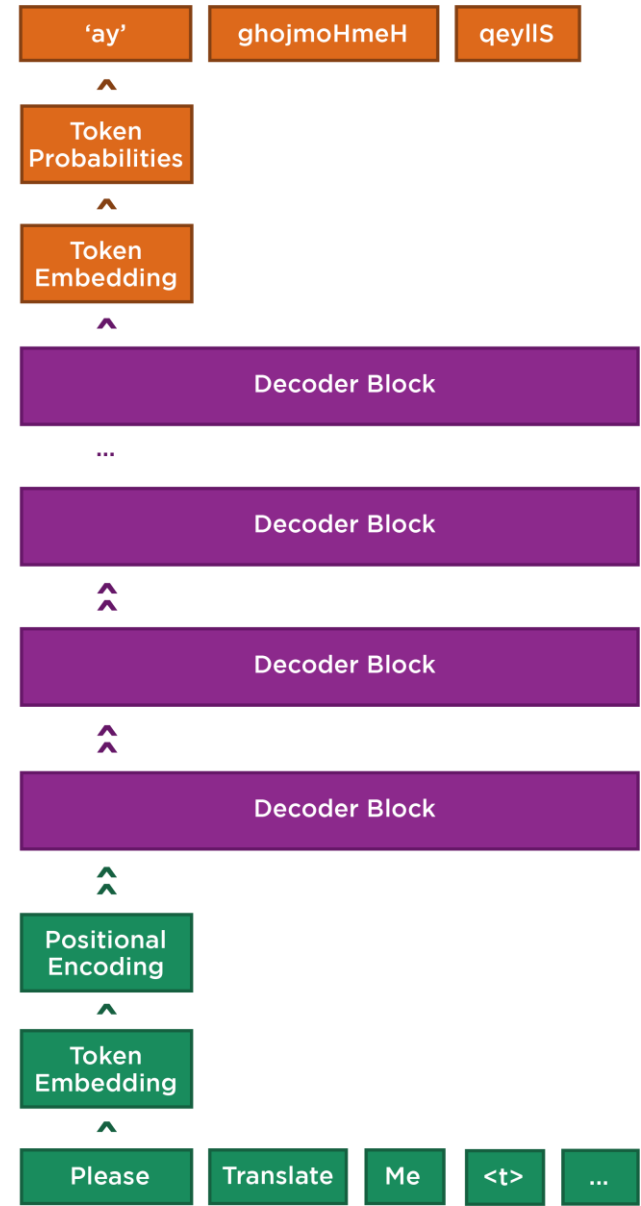
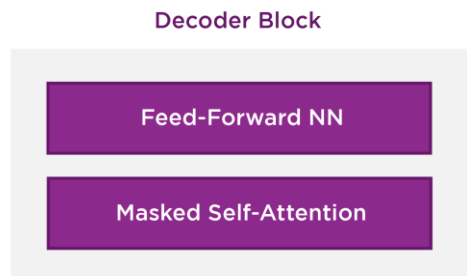


From Transformers to LLM's

Large Language Models

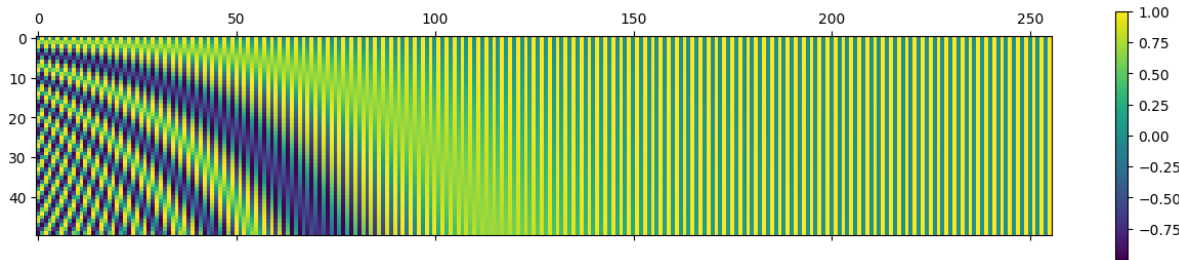
GPT architecture

- Generative Pre-Trained Transformer (GPT-2 in this example)
 - Other LLM's are similar
- Decoder-only – stacks of decoder blocks
 - Each consisting of a masked self-attention layer, and a feed-forward neural network
 - As an aside, BERT consists only of encoders. T5 is an example of a model that uses both encoders and decoders.
- No concept of input, all it does is generate the next token over and over
 - Using attention to maintain relationships to previous words / tokens
 - You “prompt” it with the tokens of your question or whatever
 - It then keeps on generating given the previous tokens
- Getting rid of the idea of inputs and outputs is what allows us to train it on unlabeled piles of text
 - It’s “learning a language” rather than optimizing for some specific task
- Hundreds of billions of parameters



LLM Input processing

- Tokenization, token encoding
- Token embedding
 - Captures semantic relationships between tokens, token similarities
- Positional encoding
 - Captures the position of the token in the input relative to other nearby tokens
 - Uses an interleaved sinusoidal function so it works on any length



GPT-3 Codex

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🍌

Sequences of characters commonly found next to each other may be grouped together: 1234567890

Clear

Show example

Tokens

64

Characters

252

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🍌🍌🍌🍌🍌

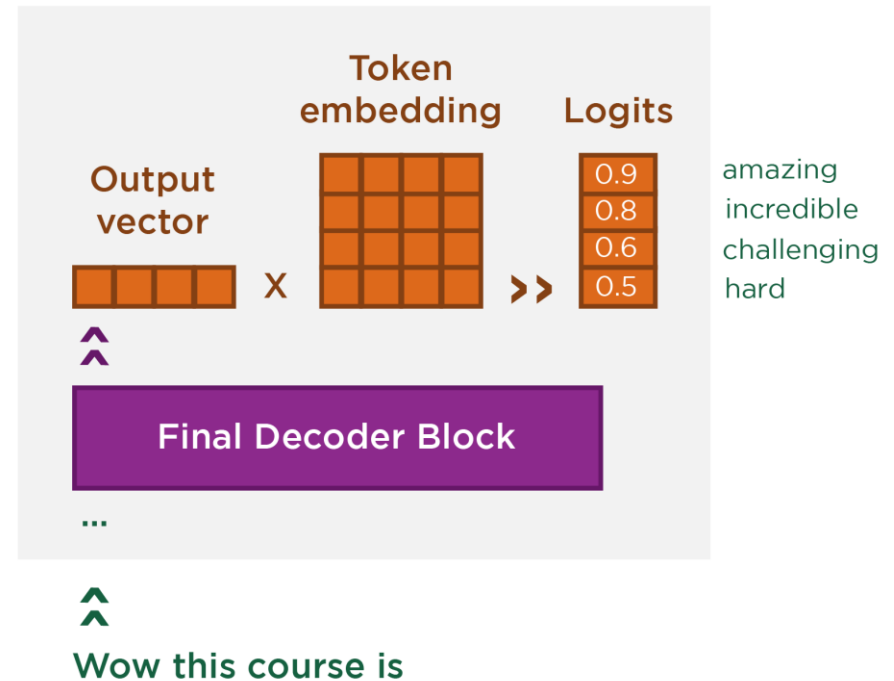
Sequences of characters commonly found next to each other may be grouped together: 1234567890

TEXT

TOKEN IDS

LLM Output processing

- The stack of decoders outputs a vector at the end
- Multiply this with the token embeddings
- This gives you probabilities (logits) of each token being the right next token (word) in the sequence
- You can randomize things a bit here (“temperature”) instead of always picking the highest probability



LLMs: Key Terms

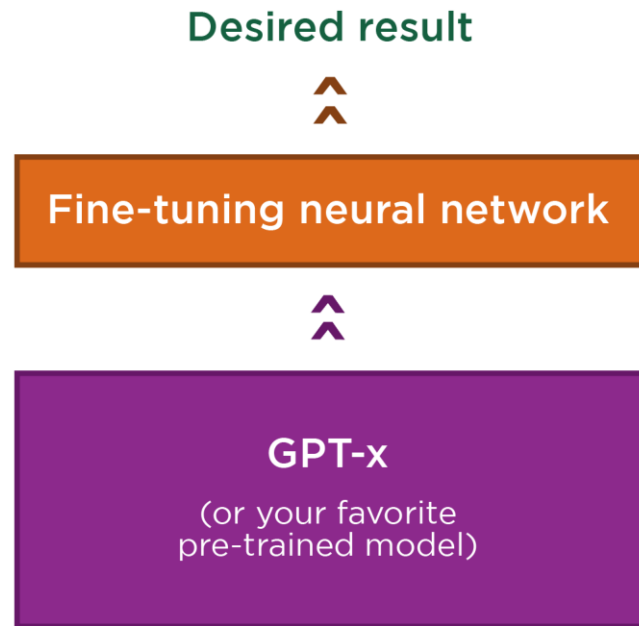
- **Tokens** – numerical representations of words or parts of words
- **Embeddings** – mathematical representations (vectors) that encode the “meaning” of a token
- **Top P** – Threshold probability for token inclusion (higher = more random)
- **Top K** – Alternate mechanism where K candidates exist for token inclusion (higher = more random)
- **Temperature** – the level of randomness in selecting the next word in the output from those tokens
 - High temperature: More random
 - Low temperature: More consistent

Context window – The number of tokens an LLM can process at once

Max tokens – Limit for total number of tokens (on input or output)

Transfer Learning (Fine Tuning) with Transformers

- Add additional training data through the whole thing
- Freeze specific layers, re-train others
 - Train a new tokenizer to learn a new language
- Add a layer on top of the pre-trained model
 - Just a few may be all that's needed!
 - Provide examples of prompts and desired completions
 - “How’s the weather?” -> “What’s it to you, bucko?”
 - Adapt it to classification or other tasks
 - “Wow, I love this course!” -> “Positive”



Generative AI in AWS

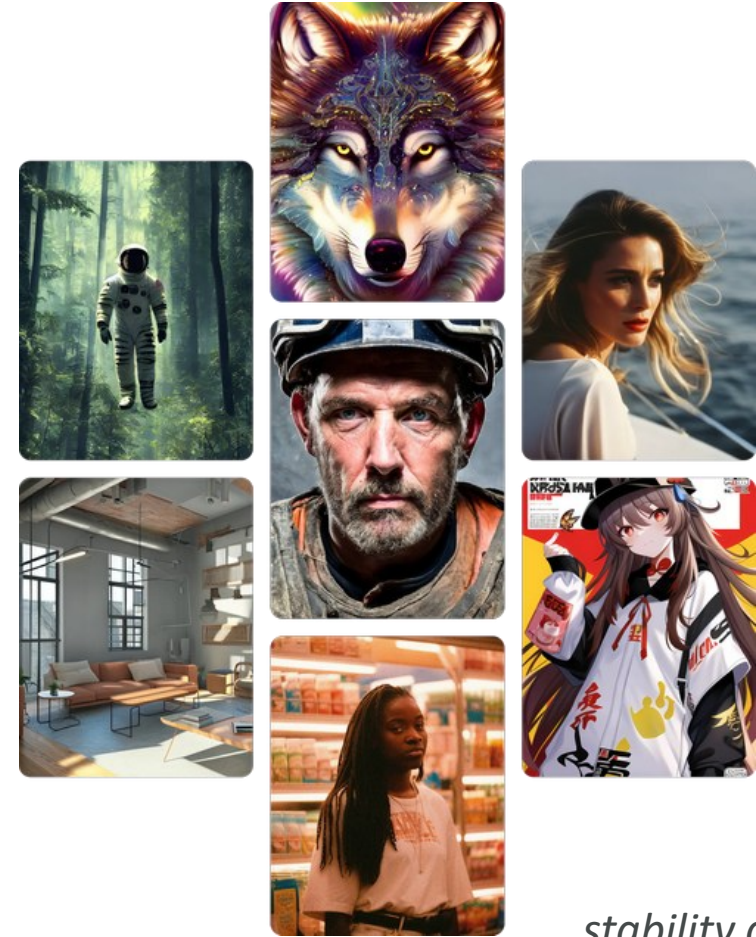
Foundation Models

- The giant, pre-trained transformer models we are fine tuning for specific tasks, or applying to new applications
- GPT-n (OpenAI, Microsoft)
- Claude (Anthropic)
- BERT (Google)
- DALL-E (OpenAI, Microsoft)
- LLaMa (Meta)
- Segment Anything (Meta)
- Where's Amazon?



AWS Foundation Models (Base Models)

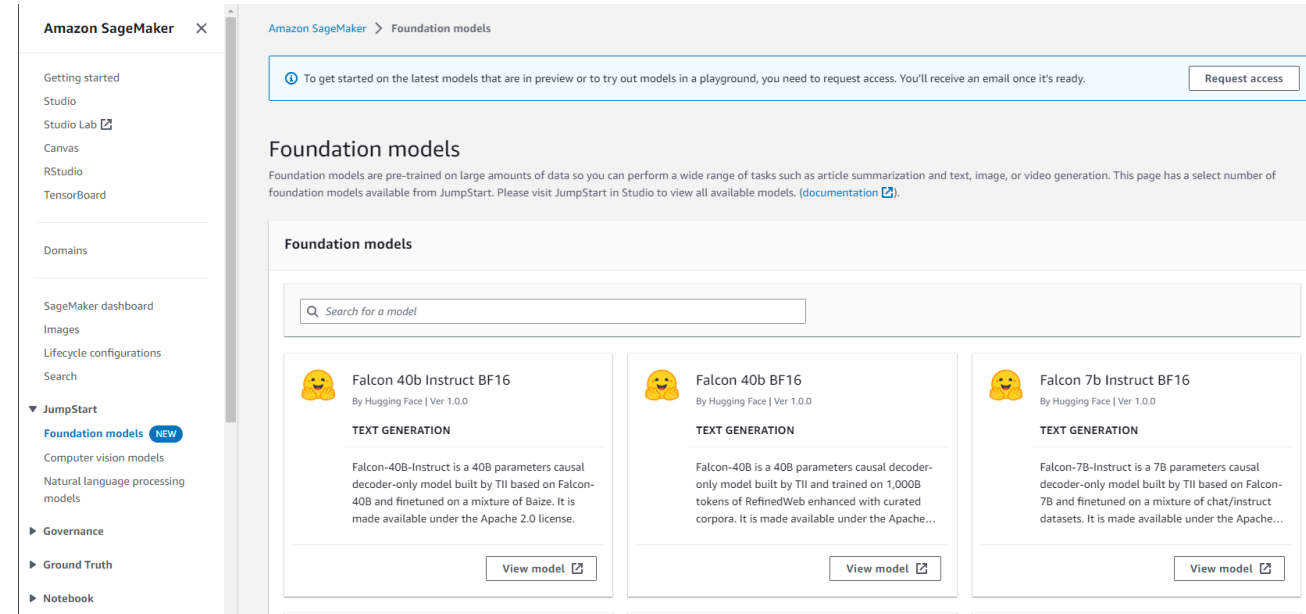
- Jurassic-2 (AI21labs)
 - Multilingual LLMs for text generation
 - Spanish, French, German, Portuguese, Italian, Dutch
- Claude (Anthropic)
 - LLM's for conversations
 - Question answering
 - Workflow automation
- Stable Diffusion (stability.ai)
 - Image, art, logo, design generation
- Llama (Meta)
 - LLM
- Amazon Titan
 - Text summarization
 - Text generation
 - Q&A
 - Embeddings
 - Personalization
 - Search



stability.ai

Amazon SageMaker Jumpstart with Generative AI

- SageMaker Studio has a “JumpStart” feature
- Lets you quickly open up a notebook with a given model loaded up and ready to go
- Current foundation models
 - Hugging Face models (text generation)
 - Falcon, Flan, BloomZ, GPT-J
 - Stable Diffusion (image generation)
 - Amazon Alexa (encoder/decoder multilingual LLM)



Amazon Bedrock

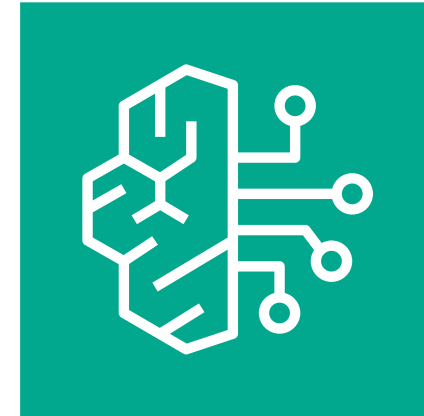
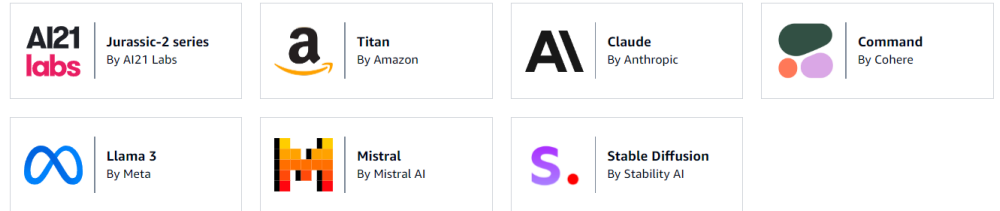
Using Foundation Models in AWS

Amazon Bedrock

- An API for generative AI Foundation Models
 - Invoke chat, text, or image models
 - Pre-built, your own fine-tuned models, or your own models
 - Third-party models bill you through AWS via their own pricing
 - Support for RAG (Retrieval-Augmented Generation... we'll get there)
 - Support for LLM agents
- Serverless
- Can integrate with SageMaker Canvas

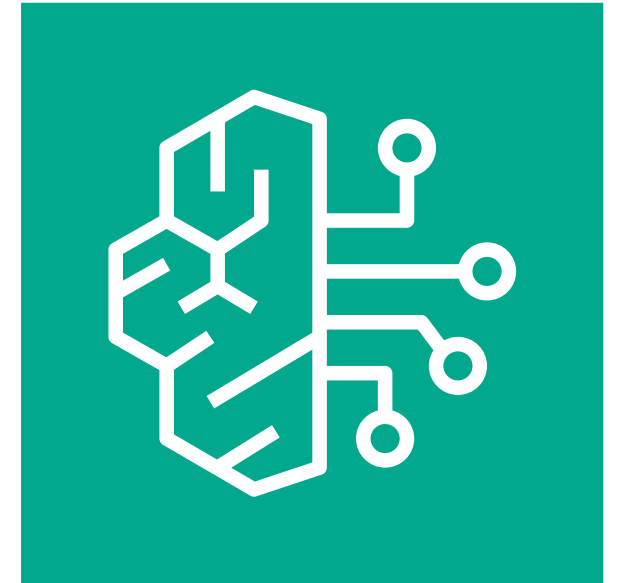
Foundation models

Amazon Bedrock supports foundation models from industry-leading providers. Choose the model that is best suited to achieving your unique goals.



The Bedrock API Endpoints

- **bedrock**: Manage, deploy, train models
- **bedrock-runtime**: Perform inference (execute prompts, generate embeddings) against these models
 - Converse, ConverseStream, InvokeModel, InvokeModelWithResponseStream
- **bedrock-agent**: Manage, deploy, train LLM agents and knowledge bases
- **bedrock-agent-runtime**: Perform inference against agents and knowledge bases
 - InvokeAgent, Retrieve, RetrieveAndGenerate



Bedrock IAM permissions

- Must use with an IAM user (not root)
- User must have relevant Bedrock permissions
 - AmazonBedrockFullAccess
 - AmazonBedrockReadOnly

Permissions policies (1275)

Filter by Type

Search Bedrock X All types

☐	Policy name ↗	Type
☐	+ AmazonBedrockAgentBedrockApplyGuardrailPo...	Customer managed
☐	+ AmazonBedrockAgentBedrockFoundationModel...	Customer managed
☐	+ AmazonBedrockAgentQuickCreateLambdaPolic...	Customer managed
☐	+ AmazonBedrockAgentRetrieveKnowledgeBaseP...	Customer managed
☐	+ AmazonBedrockFoundationModelPolicyForKno...	Customer managed
☐	+  AmazonBedrockFullAccess	AWS managed
☐	+ AmazonBedrockOSSPolicyForKnowledgeBase_i...	Customer managed
☐	+  AmazonBedrockReadOnly	AWS managed
☐	+ AmazonBedrockS3PolicyForKnowledgeBase_ivjrp	Customer managed
☐	+  AmazonSageMakerCanvasBedrockAccess	AWS managed

Amazon Bedrock: Model Access

- Before using any base model in Bedrock, you must first request access
- Amazon (Titan) models will approve immediately
- Third-party models may require you to submit additional information
 - You will be billed the third party's rates through AWS
 - It only takes a few minutes for approval
- Be sure to check pricing
 - aws.amazon.com/bedrock/pricing

Models	Access status	Modality	EULA 
▼ AI21 Labs (2)	0/2 access granted		
Jurassic-2 Ultra	 Available to request	Text	EULA
Jurassic-2 Mid	 Available to request	Text	EULA
▼ Amazon (7)	1/7 access granted		
Titan Embeddings G1 - Text	 Available to request	Embedding	EULA
Titan Text G1 - Lite	 Available to request	Text	EULA
Titan Text G1 - Express	 Available to request	Text	EULA
Titan Image Generator G1	 Available to request	Image	EULA
Titan Multimodal Embeddings G1	 Available to request	Embedding	EULA
Titan Text G1 - Premier	 Available to request	Text	EULA
Titan Text Embeddings V2	 Access granted	Embedding	EULA
▼ Anthropic (4)	1/4 access granted		
Claude 3 Sonnet	 Available to request	Text & Vision	EULA
Claude 3 Haiku	 Available to request	Text & Vision	EULA
Claude	 Access granted	Text	EULA
Claude Instant	 Available to request	Text	EULA
▼ Cohere (6)	0/6 access granted		
Command R+	 Available to request	Text	EULA
Command R	 Available to request	Text	EULA

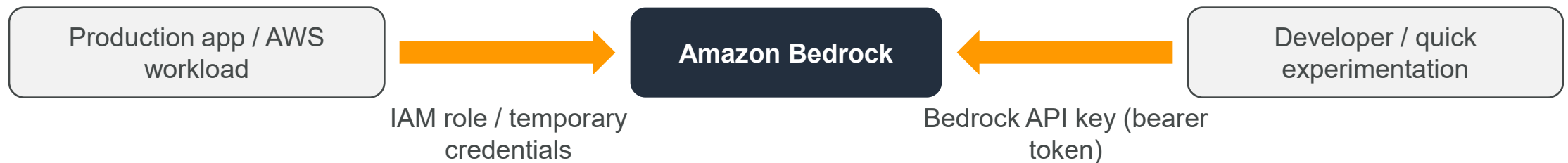
Amazon Bedrock Credentials: API Keys vs. IAM

IAM Roles / Temporary Credentials

- Preferred for production AWS workloads & applications with stronger security requirements
- Normal AWS authorization via IAM permissions
- Supports least privilege

Amazon Bedrock API Keys

- Simpler, bearer-token-style way to call Bedrock APIs
- Short-term: inherits permissions of the IAM principal that generated it, expires (max 12 hours)
- Long-term: exploration / development only – prefer IAM for production



Exam tip: production AWS workload → IAM role / temporary credentials; quick Bedrock experimentation → an API key may be simpler

Never hard-code credentials in source code – protect them appropriately

Let's Play

- Bedrock provides “playground” environments
 - Chat
 - Text
 - Images
- Must have model access first
- Also useful for evaluating your own custom or imported models
- Let's go hands-on and get a feel for it.



Fine-tuning

- Adapt an existing large language model to your specific use case!
- Additional training using your own data – potentially lots of it
 - Eliminates need to build up a big conversation to get the results you want (“prompt engineering” / “prompt design”)
 - Saves on tokens in the long run
- Your fine-tuned model can be used like any other
- You can fine-tune a fine-tuned model, making it “smarter” over time
- Applications:
 - Chatbot with a certain personality or style, or with a certain objective (i.e., customer support, writing ads)
 - Training with data more recent than what the LLM had
 - Training with proprietary data (i.e., your past emails or messages, customer support transcripts)



Fine-tuning in Bedrock: “Custom Models”

- Titan, Cohere, and Meta models may be “fine-tuned”
- Text models: provide labeled training pairs of prompts and completions
 - Can be questions and answers
 - Upload training data into S3
- Image models: provide pairs of image S3 paths to image descriptions (prompts)
 - Used for text-to-image or image-to-embedding models
- Use a VPC and PrivateLink for sensitive training data
- This can get expensive
- Your resulting “custom model” may then be used like any other.

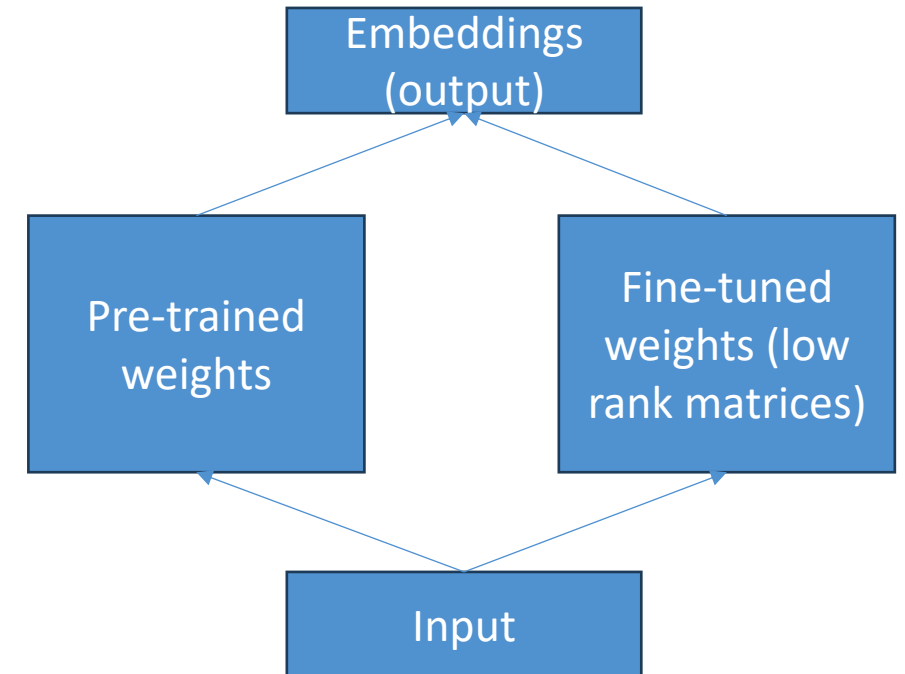
```
{"prompt": "What is the meaning of life?",  
"completion": "The meaning of life is 42."}
```

```
{"prompt": "Who was the best Dr. Who?", "completion":  
"Matt Smith in Series 5 was the best, and anyone who says otherwise is wrong."}
```

```
{"prompt": "Is Dr. Who better than Star Trek?",  
"completion": "Blasphemer! Star Trek changed the world like no other science fiction has."}
```


Low-Rank Adaptation (LoRA)

- We don't update the entire model, just slap on some “low-rank matrices” to the attention weights (usually), and train those.
 - “Low-rank” refers to the complexity of the underlying matrices in the model, but you don't need that level of understanding for this exam.
- At inference, these fine-tuned weights get added into the base model
- Base model remains unchanged
- Very efficient for storage, training, and inference
- This is different from an “adapter layer” added to the top of a model



“Continued Pre-Training”

- Like fine-tuning, but with unlabeled data
- Just feed it text to familiarize the model with
 - Your own business documents
 - Whatever
- Basically including extra data into the model itself
 - So you don't need to include it in the prompts

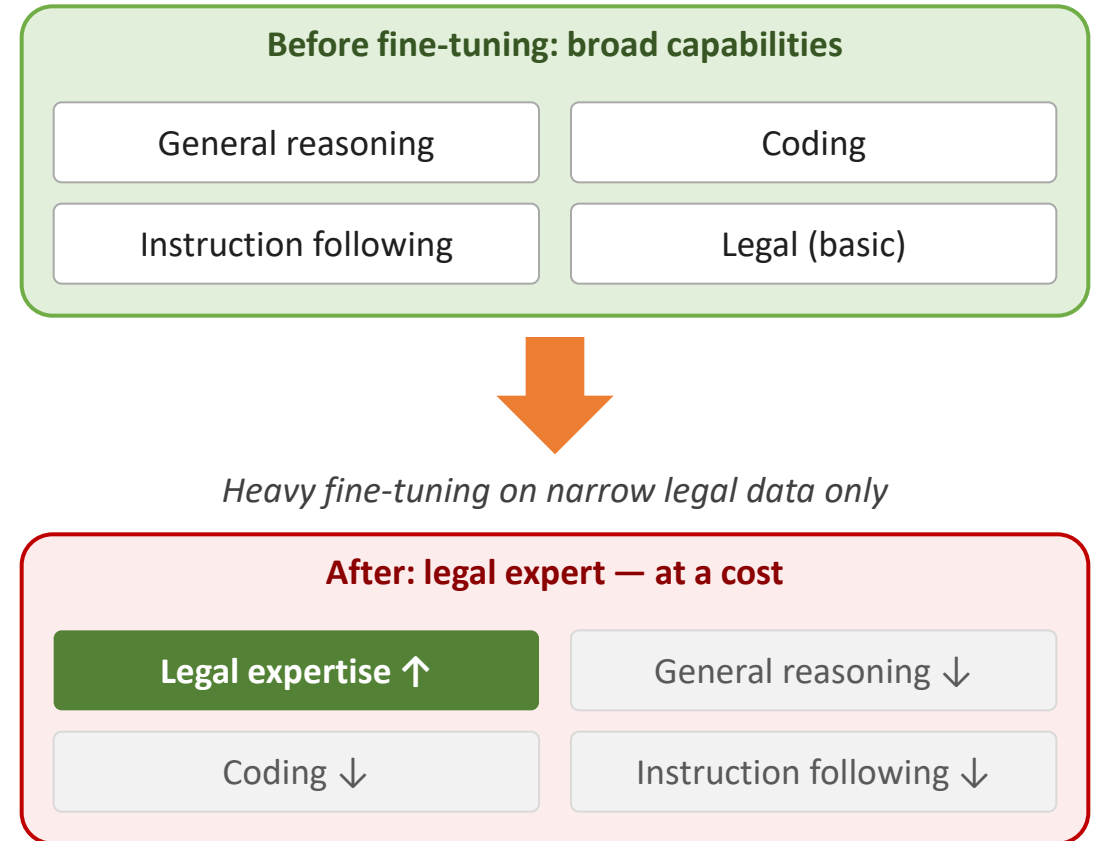
```
{"input": "Spring has sprung. "}
```

```
{"input": "The grass has riz."}
```

```
{"input": "I wonder where the flowers is."}
```

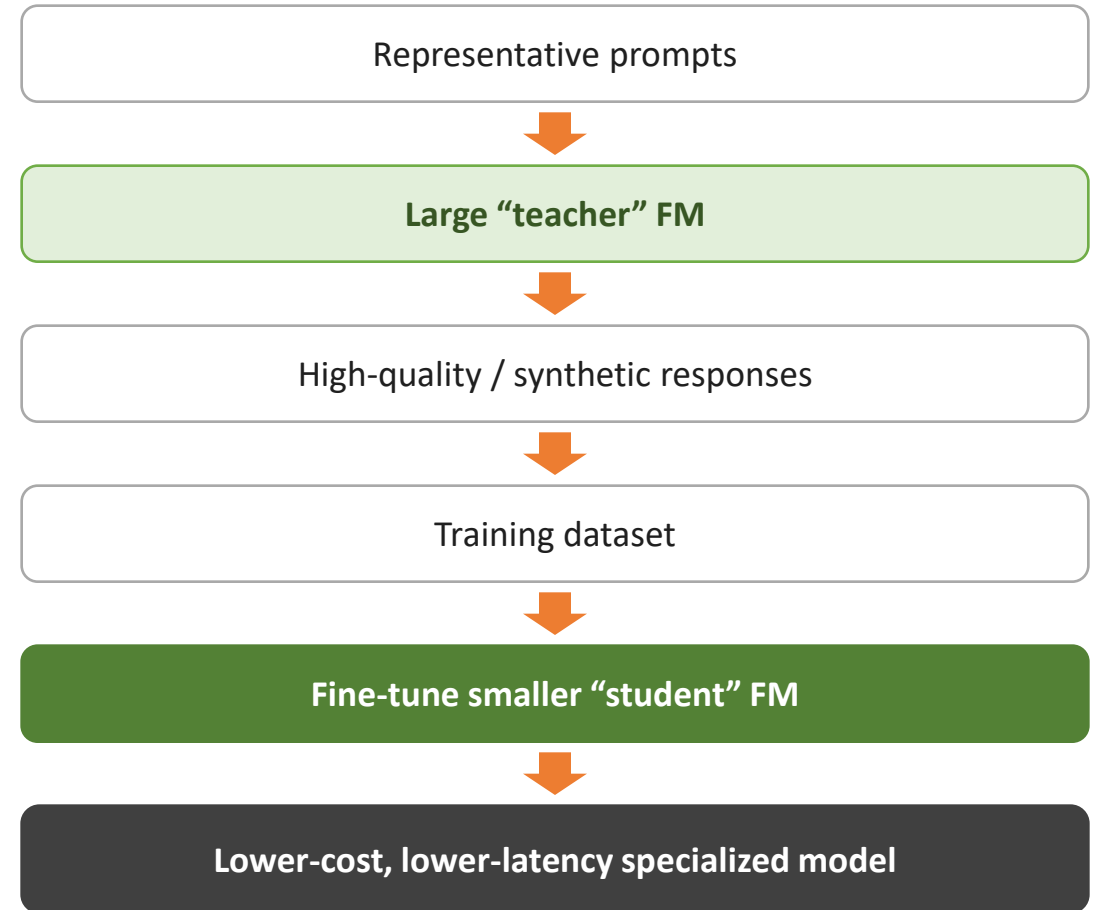
“Catastrophic Forgetting”

- Fine-tuning can make a model lose capabilities it already learned
- Example: tuning on narrow legal data improves legal answers...
 - ...but degrades reasoning, coding, instruction following
- Most likely when new data is narrow or very different from the original mix
- Mitigate: mix in general data (Nova “data mixing”), modest learning rate / epochs
- Evaluate the new task and retained general capabilities
- Exam tip: customization should improve the target task without breaking the rest



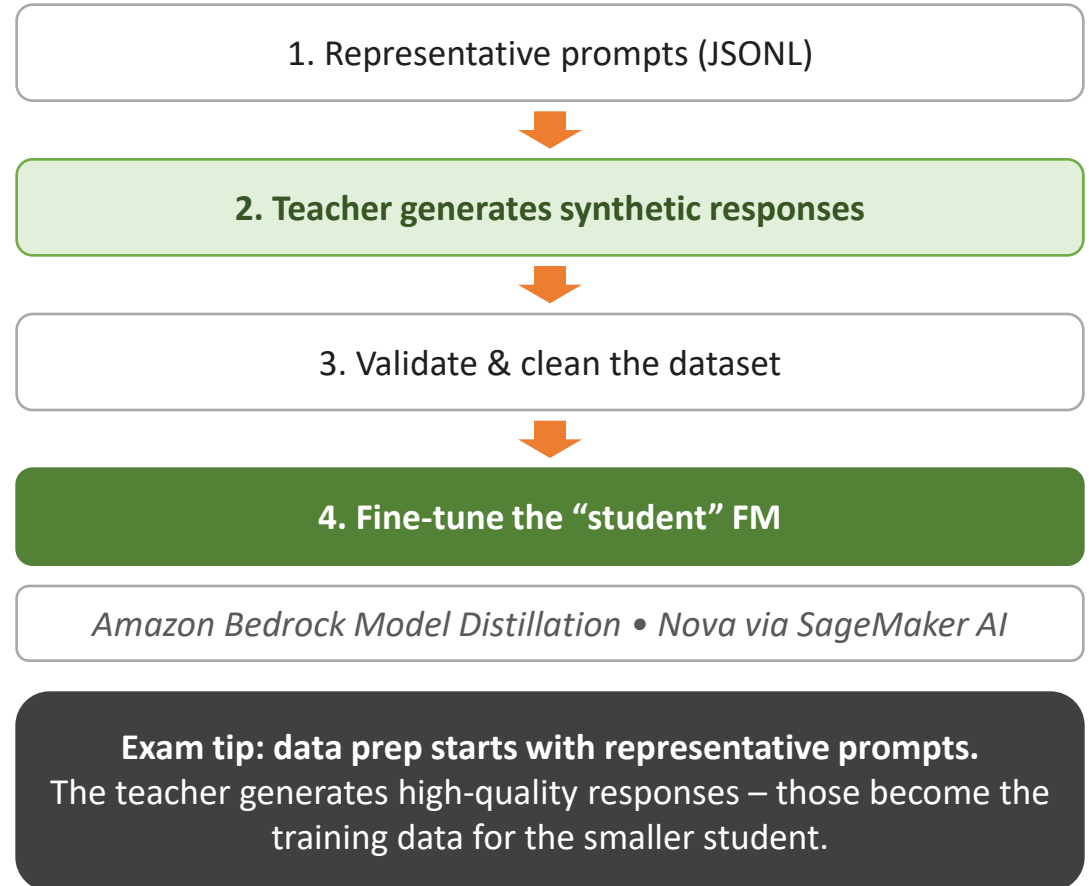
Model Distillation

- Transfers capability from a large “teacher” FM to a smaller, faster “student”
 - Goal: the student approaches the teacher’s quality for your specific workload
- The teacher generates high-quality (synthetic) responses
 - The student is fine-tuned on this teacher-generated data
- Result: lower inference cost and latency, with task-specific quality retained
- Not the same as quantization or pruning
 - Distillation trains a new, smaller model rather than compressing the original



Preparing Data for Model Distillation

- Start with prompts that represent your production workload
 - Coverage matters: typical tasks, variations, edge cases, domain scenarios
 - Format them for the use case – RAG, agents, classification, summarization all differ
 - Commonly JSONL conversation records
- The teacher turns your prompts into synthetic training responses
 - Better prompts → better training data
- Validate before you fine-tune the student
 - Check samples for relevance, correctness, coherence
 - Fix or remove bad examples; keep separate evaluation data

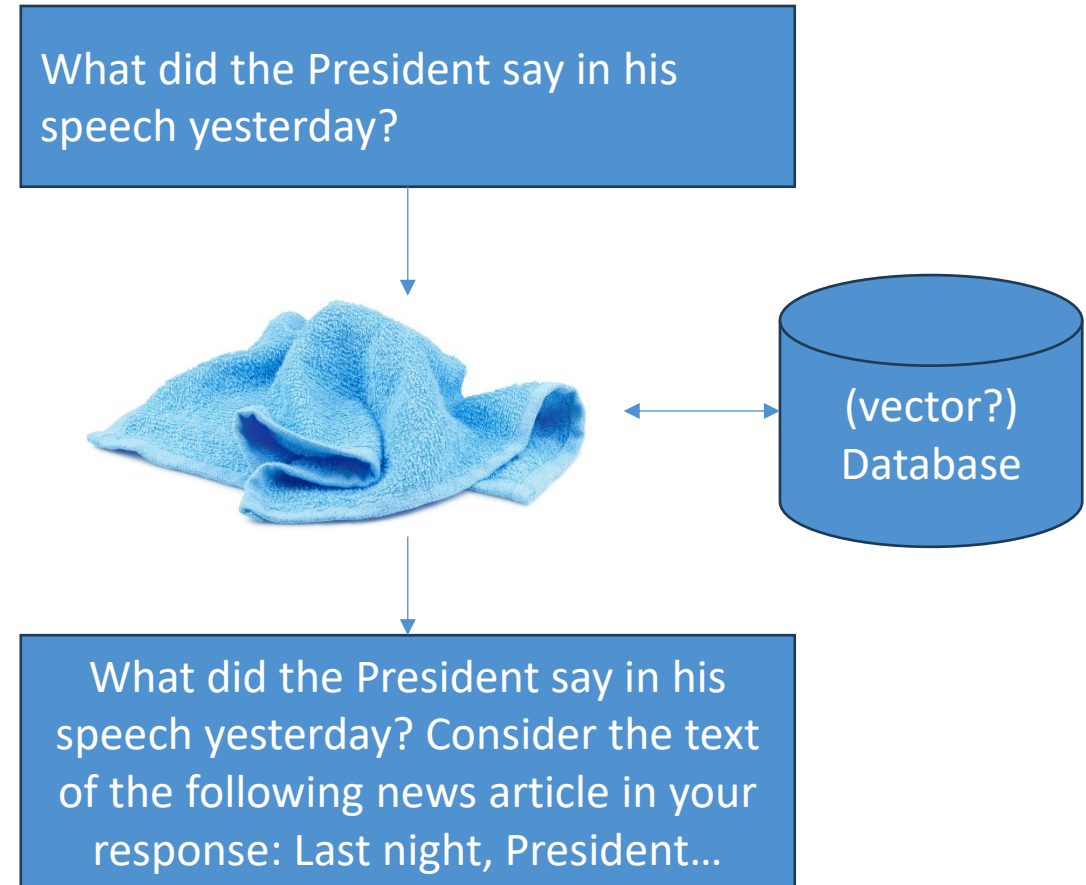


Retrieval Augmented Generation (RAG)

Bedrock Knowledge Bases

Retrieval Augmented Generation (RAG)

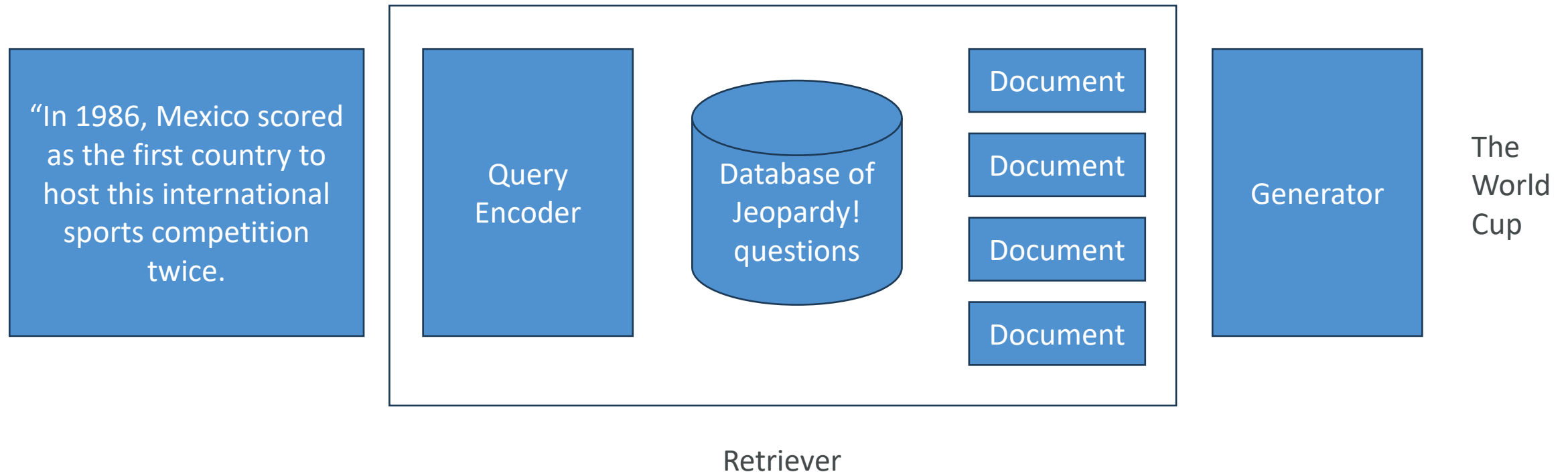
- Like an open-book exam for LLM's
- You query some external database for the answers instead of relying on the LLM
- Then, work those answers into the prompt for the LLM to work with
 - Or, use tools and functions to incorporate the search into the LLM in a slightly more principled way



RAG: Pros and Cons

- Faster & cheaper way to incorporate new or proprietary information into “GenAI” vs. fine-tuning
- Updating info is just a matter of updating a database
- Can leverage “semantic search” via vector stores
- Can prevent “hallucinations” when you ask the model about something it wasn’t trained on
- If your boss wants “AI search”, this is an easy way to deliver it.
- Technically you aren’t “training” a model with this data
- You have made the world’s most overcomplicated search engine
- Very sensitive to the prompt templates you use to incorporate your data
- Non-deterministic
- It can still hallucinate
- Very sensitive to the relevancy of the information you retrieve

RAG: Example Approach (winning at Jeopardy!)



Choosing a Database (Knowledge Base Data Store) for RAG

- You could just use whatever database is appropriate for the type of data you are retrieving
 - Graph database (i.e., Neo4j) for retrieving product recommendations or relationships between items
 - Opensearch or something for traditional text search (TF/IDF)
 - But almost every example you find of RAG uses a Vector database
 - Note Elasticsearch / Opensearch can function as a vector DB

Q: 'Which pink items are suitable for children?'

```
{  
  "color": "pink",  
  "age_group": "children"  
}
```

Q: 'Help me find gardening gear that is waterproof'

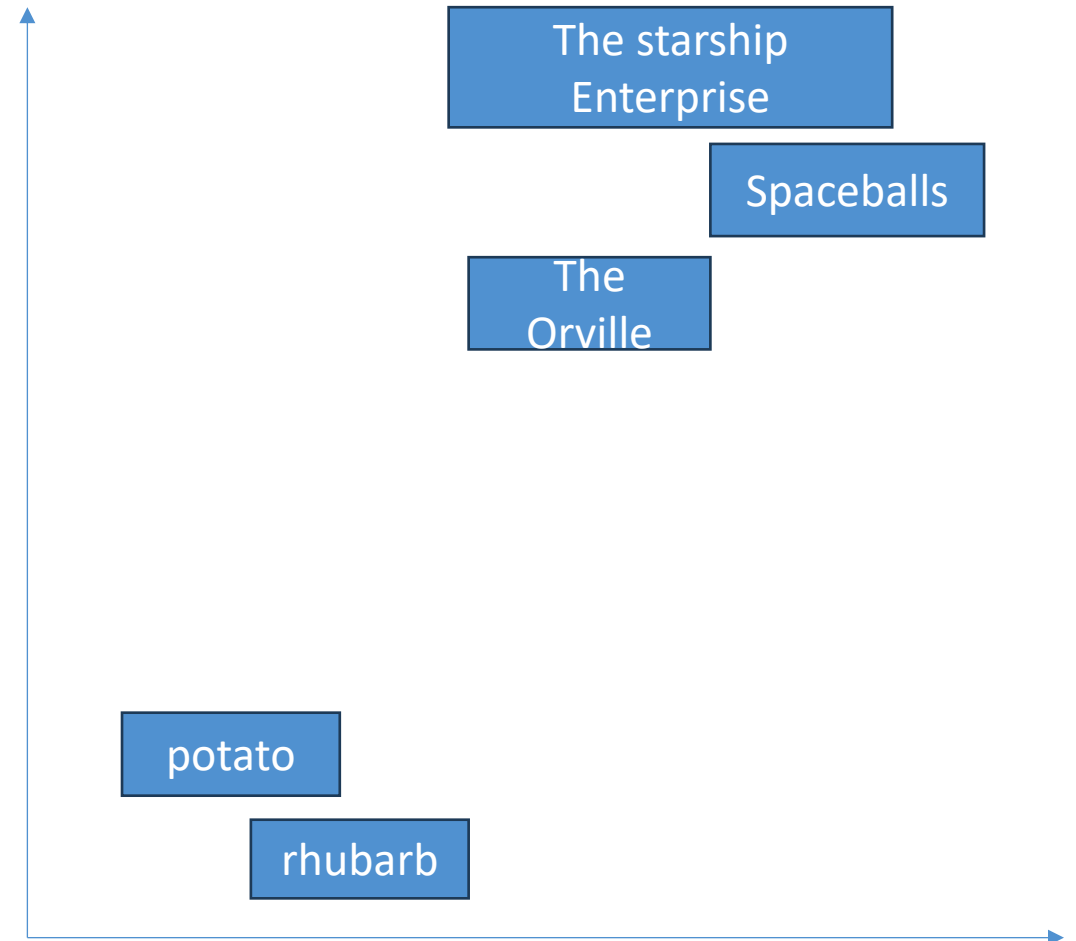
```
{  
  "category": "gardening gear",  
  "characteristic": "waterproof"  
}
```

Q: 'I'm looking for a bench with dimensions 100x50 for my living room'

```
{  
  "measurement": "100x50",  
  "category": "home decoration"  
}
```

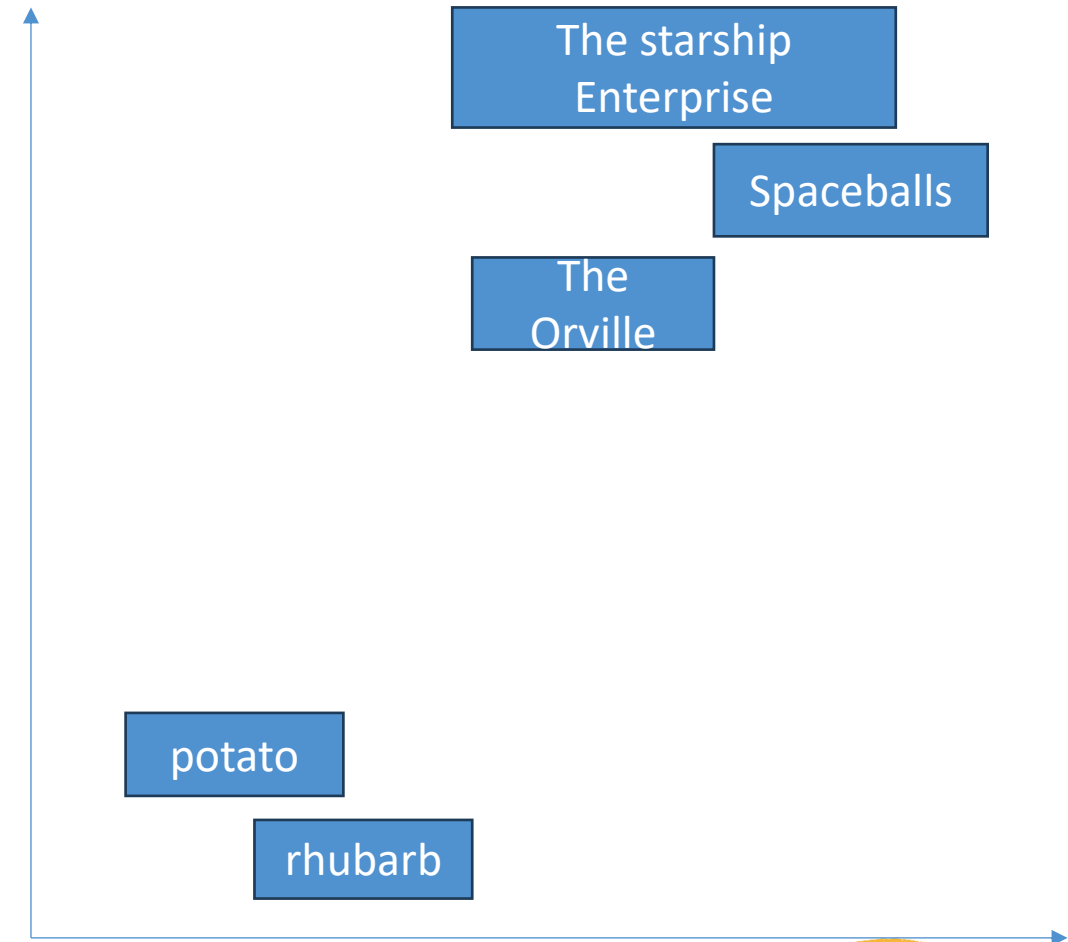
Review: embeddings

- An **embedding** is just a big vector associated with your data
- Think of it as a point in multi-dimensional space (typically 100's or thousands of dimensions)
- Embeddings are computed such that items that are similar to each other are close to each other in that space
- We can use embedding base models (like Titan) to compute



Embeddings are vectors, so store them in...

- ...a vector database!
- It just stores your data alongside their computed embedding vectors
- Leverages the embeddings you might already have for ML
- Retrieval looks like this:
 - Compute an embedding vector for the thing you want to search for
 - Query the vector database for the top items close to that vector
 - You get back the top-N most similar things (K-Nearest Neighbor)
 - “Vector search”
- Examples of vector databases
 - Coercing existing databases to do vector search
 - Opensearch / Elasticsearch, SQL, Neptune, Redis, MongoDB, Cassandra
 - Purpose-built vector DB's
 - Pinecone, Weaviate (commercial)
 - Chroma, Marqo, Vespa, Qdrant, LanceDB, Milvus, vectordb (open source)



Optimizing your Embeddings

- Smaller vector sizes are a good way to reduce cost
 - I'm talking vector size here, not chunk size
 - Smaller vectors = fewer dimensions per chunk = less cost
 - However there is a tradeoff with your retrieval performance
- Amazon Titan has a default of 1024+ dimensions
 - This might be more than you need!
 - Balance dimensionality with domain fit
 - Are the semantic concepts you are searching for really that complex?

Sparse vs. Dense Embeddings

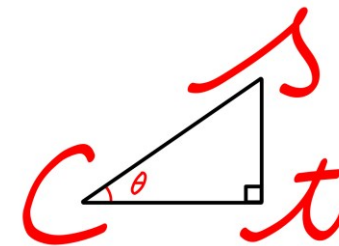
- Sparse = large vectors that are mostly empty
 - I.e. One-hot encoding
- Dense = smaller vectors that contain more semantic information
 - This is what you generally use
- There is a tradeoff
 - Sparse vectors give greater similarity factors
 - Dense vectors are more efficient (in memory, and in general)
 - Classical AI systems have ways to compress sparse vectors
- Wait, what's a similarity factor
 - Some measure of how close two vectors are
 - A common one is **cosine similarity**
 - Remember your trig? Cosine is related to the angle between two vectors.
 - And it has a nice smoothness to it

Chicken	Squid	Lemur	Bonobo	Ameoba
1	0	0	0	0
0	0	0	1	0

Sparse vectors

	Dim0	Dim1	Dim2	Dim3	Dim4
Chicken	0.43	0.2423	0.542	0.123	0.532
Bonobo	0.24	0.34	0.53	0.632	0.23

Dense vectors



Optimizing Retrieval with Metadata

- Your vector DB can store more than just vectors and the raw text of the chunk
 - Bedrock Knowledge Bases can treat certain columns as content vs. metadata
 - Pass in a metadata.json file to specify what's what
 - This makes sure you aren't chunking up metadata (like creation date) but can still retrieve it
- If you add metadata tags for things like topics, what section of the document we're in, etc... you can use this for better retrieval too.
 - Relevance scoring against metadata can be used for ranking
- Other examples of metadata
 - Document ID, category, access control, data lineage, additional context

"A chicken is a domesticated bird commonly raised for meat and eggs."

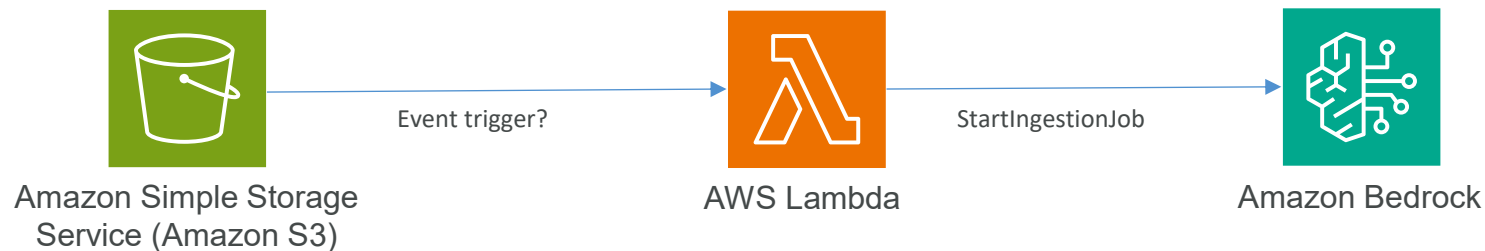


```
{  
  "section": "introduction",  
  "topic": "biology",  
  "keywords": ["chicken",  
"domesticated", "bird", "meat",  
"eggs"]  
}
```



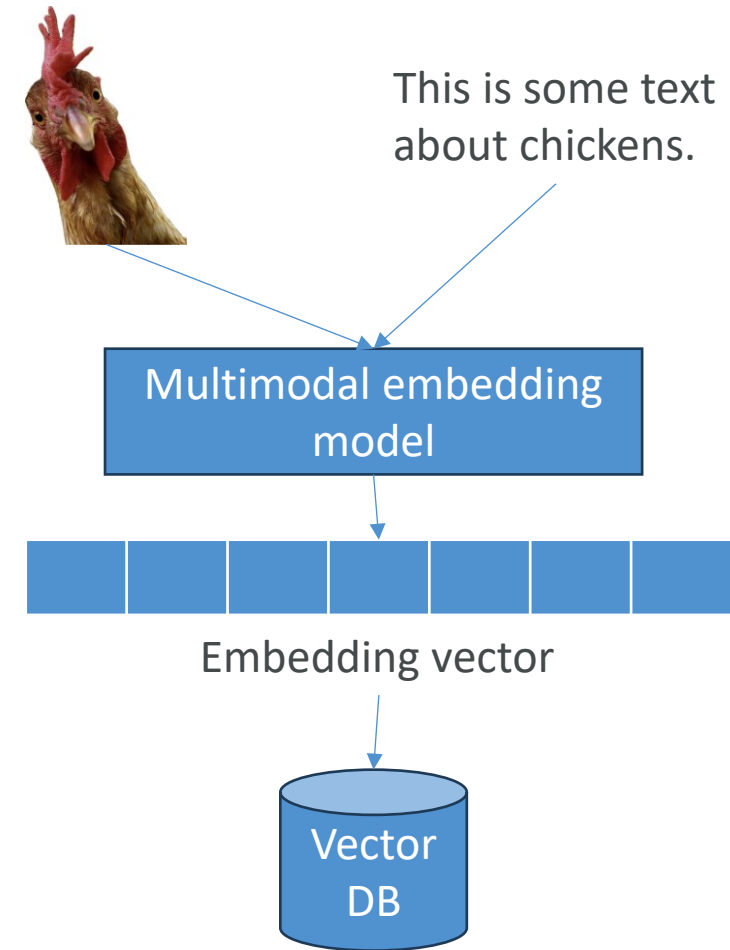
Keeping Your KB Up to Date

- New or changed content can trigger a Lambda function
- This Lambda function can then generate new embeddings
- Batch-generate them for greater efficiency
 - Instead of syncing on every change, maybe you can do so on a schedule



Multimodal Models and Pipelines

- “Multimodal” refers to mixing different media types
 - Text, images, audio, video, documents, whatever
 - For example, a model could contain accept text, a scanned image of some text, someone saying something, a PDF, etc.
 - Requires specialized encoders for each type
- Some models available for Bedrock are multimodal
 - Claude, Nova, Titan for example
- Multimodal embedding models can convert different media types into compatible embedding vectors.
- To search, hit the model for an embedding vector for the image / text / video / whatever, and find the closest match in the vector DB.



Multimodal Models and Pipelines

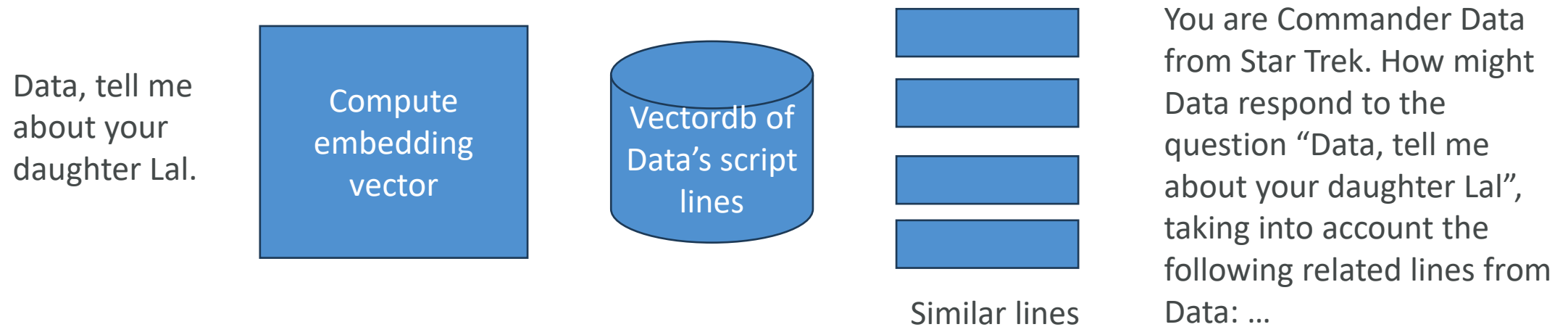
- Preparing multi-modal input data
 - For **Titan Multimodal Embeddings G1**, you need to pass in structured JSON like this example, where the image data is base64-encoded.
- Your data processing pipeline needs to perform this conversion at some point
 - SageMaker? Glue? We'll get to that.

```
model_id = "amazon.titan-embed-image-v1"
input_text = "A chicken"
with open("/path/to/image", "rb") as chick_pic:
    input_image =
base64.b64encode(chick_pic.read()).decode('utf8')

body = json.dumps({
    "inputText": input_text,
    "inputImage": input_image,
})
```

*You can pass in text, an image, or both

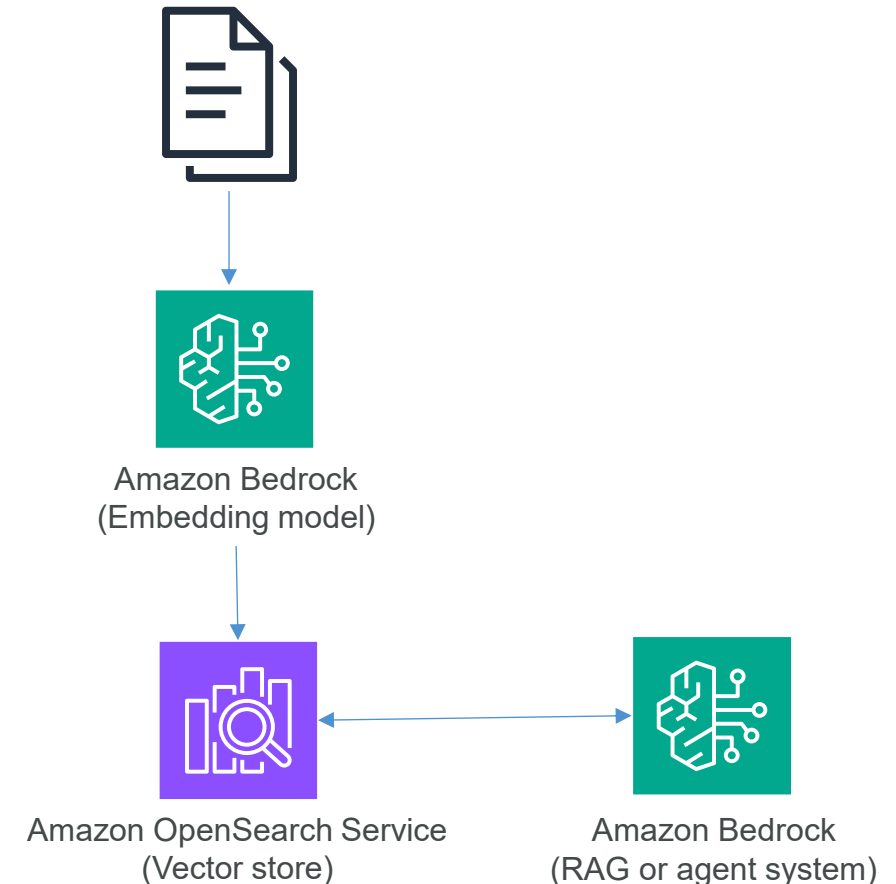
RAG Example with a Vector Database: Making Data from Star Trek, by Cheating.



Prompt + query + relevant data

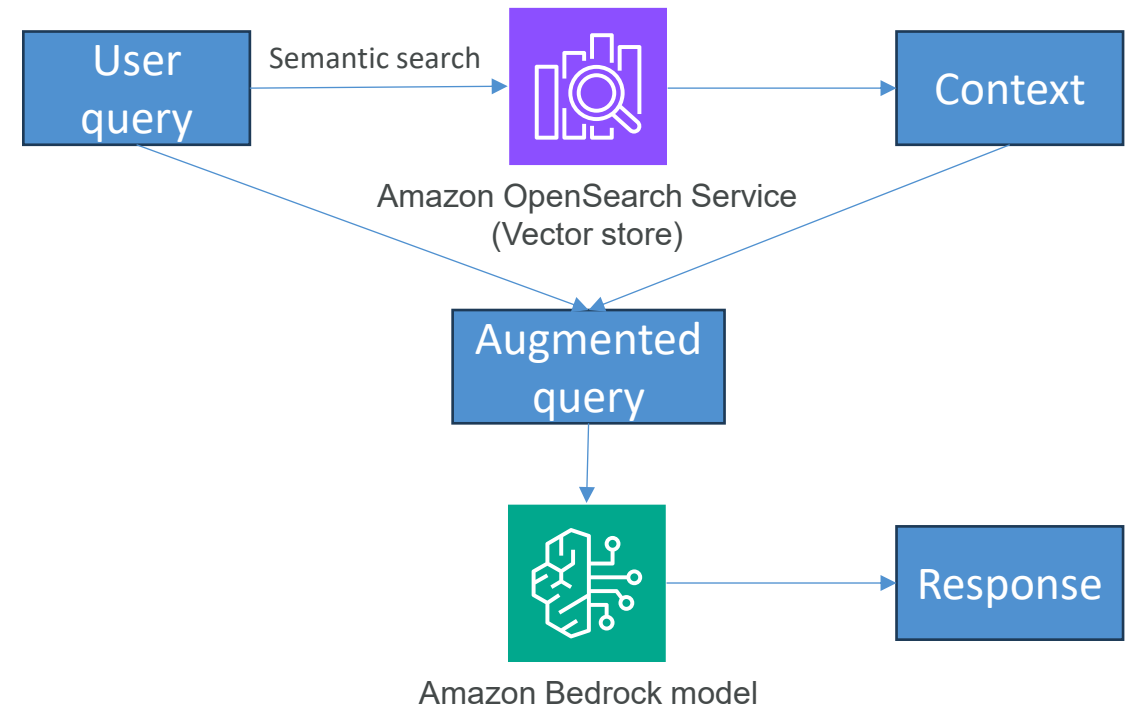
RAG in Bedrock: Knowledge Bases

- You can upload your own documents or structured data (via S3, maybe with a JSON schema) into Bedrock “Knowledge Bases”
 - Other sources: web crawler, Confluence, Salesforce, SharePoint
- Must use an embedding model
 - For which you must have obtained model access
 - Currently Cohere or Amazon Titan
 - You can control the vector dimension
- And a vector store of some sort
 - For development, a serverless OpenSearch instance may be used by default
 - MemoryDB now has vector capabilities
 - Or Aurora, MongoDB Atlas, Pinecone, Redis Enterprise Cloud
 - You can control the “chunking” of your data
 - How many tokens are represented by each vector



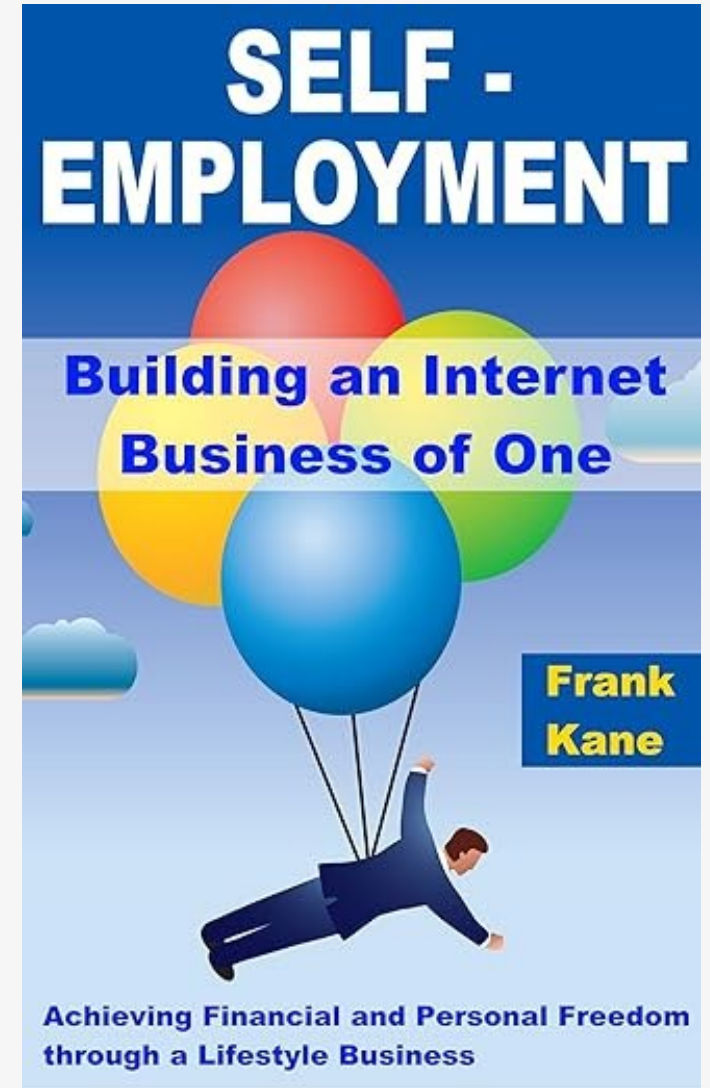
Using Knowledge Bases

- “Chat with your document”
 - Basically, automatic RAG
- Integrate it into an application directly
- Incorporate it into an agent
 - “Agentic RAG”



Let's Make a Knowledge Base

- AKA a vector store
- Hands-on; let's use the text of a book I wrote.



OpenSearch as a Vector Store

- Serverless vector stores
 - Note “serverless” does not mean “scales to zero”
 - Remember to shut down your serverless OpenSearch instances if you’re not using them – or you’ll keep getting billed
 - The exam treats OpenSearch as the primary implementation for backing Bedrock Knowledge Bases
 - But I think this might get subsumed by the newer S3 vectors feature
- Not limited to Bedrock Knowledge Bases
 - Can also integrate with SageMaker AI and HuggingFace or custom models for RAG
- SEMANTIC vs. HYBRID search
 - Semantic refers to vector search
 - As we’ve seen relevancy can be elusive and dependent on your chunking strategy
 - Hybrid **combines vector and keyword search**
 - Requires indexing filterable keyword fields for metadata
 - This is important



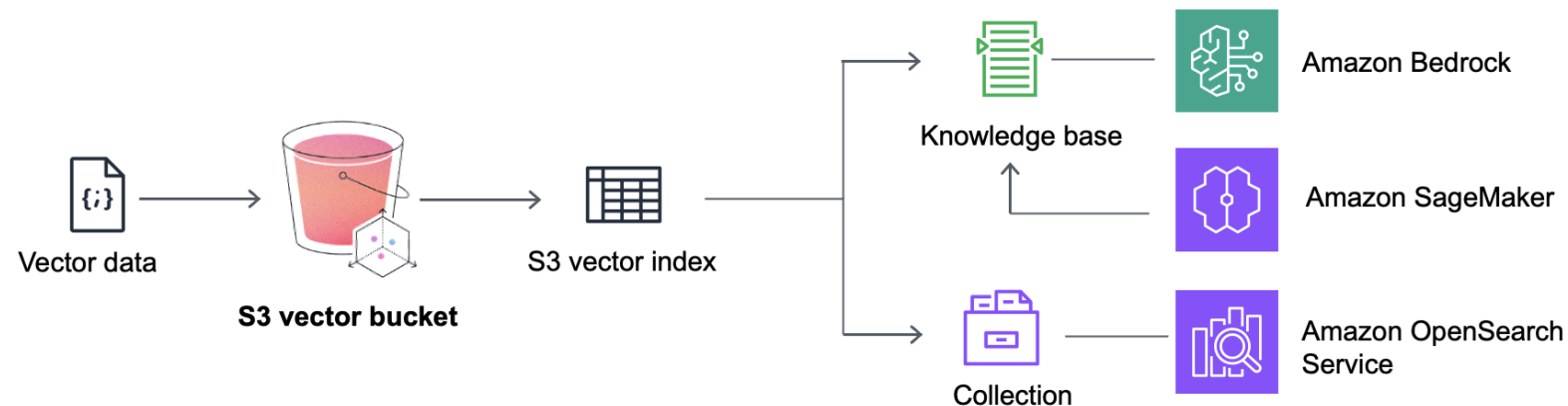
Amazon Bedrock



Amazon OpenSearch Service

Amazon S3 Vectors

- Up to 90% cheaper (and a lot simpler)
- Create a S3 vector bucket
- Then create a vector index
 - Specify dimensions and distance metric
- Now you can store your vector embeddings with `put_vectors`
 - Include whatever metadata you want
 - Query them with `query_vectors`



S3 Vectors: More specifics?

- They are strongly consistent
 - Newly added data is immediately available
- The vector data is automatically optimized over time for *price performance*
- But actual performance is only advertised as 100ms-1s
 - **AWS recommends a hybrid approach with OpenSearch for performance-critical queries**
 - They call this a “tiered search strategy” – use S3 for infrequently queried vectors only
 - S3 Vectors has connectivity with OpenSearch for this purpose
 - This COPIES (exports) data from S3 into OpenSearch – you end up paying for both if you’re not careful!
- You can also use OpenSearch with S3 Vectors engine
 - Only works with OpenSearch managed clusters
 - Give you OpenSearch functionality with S3 Vectors as the backend
- Max indices per bucket: 10,000 (2B vectors per index)

Amazon Aurora as a Vector Store

- **pgvector** extension
 - Creates a new “vector” column type
 - Can compute cosine, L2, or inner product distances
 - Can do similarity search with IVF or
 - Also adds vector operators for SQL
 - This gives it an advantage over OpenSearch if you need complex filtering logic and stuff.
- Allows you to store structured and vector data in the same place
- Can be appropriate for small / medium RAG systems with mostly structured data



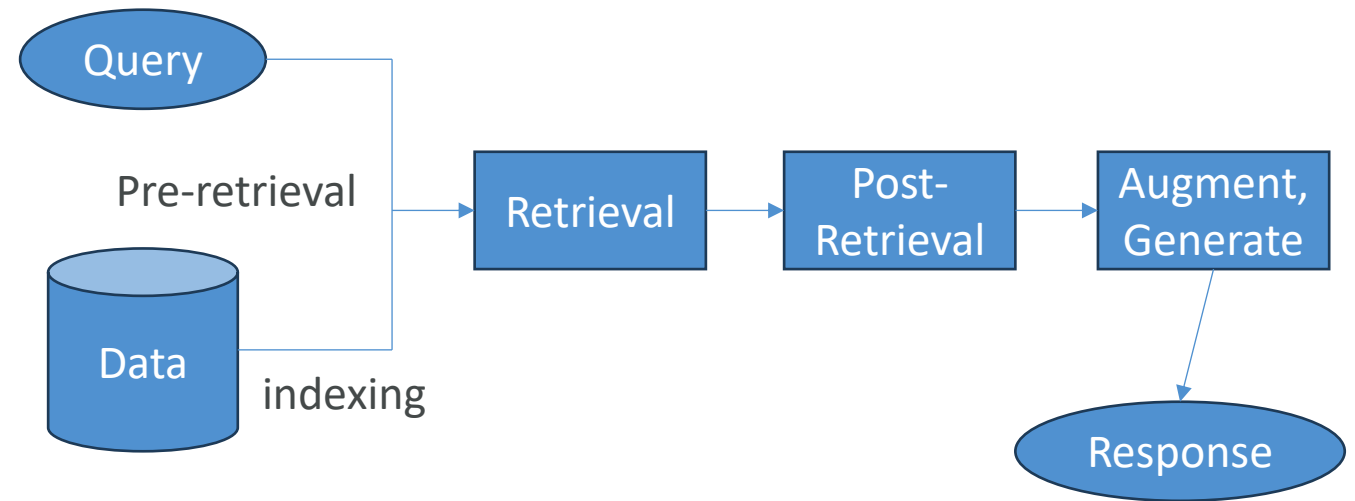
Amazon Aurora



Trusted Language
Extensions for
PostgreSQL

Breaking up the R in RAG

- Pre-Retrieval
 - Indexing
 - Granularity / chunking
 - Data extraction
 - Query Rewriting
- Retrieval
- Post-Retrieval



Pre-Retrieval

- What should you be storing in your database?
- Data granularity matters
- Example: Every set of lines Data had, or individual sentences? Or blocks of text of a fixed length? Why just Data's lines?
 - Or maybe summaries of those lines?
 - You can use an LLM for this too...
- The process of splitting up your data prior to storage is referred to as CHUNKING.

DATA:
I cannot become nervous. However,
I do sense a certain...
anticipation regarding my role in the wedding.
(beat)
All systems normal, sir.
Sickbay reported that Lieutenant Juarez
went into labor at zero four hundred hours.
We remain on station awaiting the arrival
of the starship Zhukov and guest quarters
have been prepared for Ambassador T'Pel.



I cannot become nervous.

However,
I do sense a certain...
anticipation regarding my role in the wedding.

All systems normal, sir.

Sickbay reported that Lieutenant Juarez
went into labor at zero four hundred hours.

We remain on station awaiting the arrival
of the starship Zhukov and guest quarters
have been prepared for Ambassador T'Pel.

More on Chunking

- Make sure we don't end up feeding more info into the LLM than it can handle
- Stay within the limit of how many tokens can be used within a context
 - This limit is becoming larger and larger though.
- Semantic Chunking
 - Ensure each chunk contains semantically independent information
 - Embedding-based (LlamaIndex / Langchain)
 - Model-based (BERT)
 - LLM-based (Basically tell it to do semantic chunking)

Decompose the "Content" into clear and simple propositions, ensuring they are interpretable out of context.

1. Split compound sentence into simple sentences.

Maintain the original phrasing from the input whenever possible.

2. For any named entity that is accompanied by additional descriptive information, separate this information into its own distinct proposition.

3. Decontextualize the proposition by adding necessary modifier to nouns or entire sentences and replacing pronouns (e.g., "it", "he", "she", "they", "this", "that") with the full name of the entities they refer to.

4. Present the results as a list of strings, formatted in JSON.

Sample LLM prompt for chunking from the paper "Dense Retrieval: What Retrieval Granularity Should We Use?" Chen/Wang/Chen/Yu/Ma/Zhao/Zhang/Yu and also using within LlamaIndex

Chunking with Bedrock: Standard Chunking

- Fixed Size
 - You specify tokens per chunk and overlap percentage
- Default
 - 300 tokens per chunk, while honoring sentence boundaries (complete sentences preserved within each chunk)
- No chunking
 - Every document is a chunk
 - You could pre-process your own data to “chunk” it however you want per document

Space, the final frontier. These are the voyages of the Starship Enterprise.

Fixed chunking;
Chunk size 5,
Overlap 20%

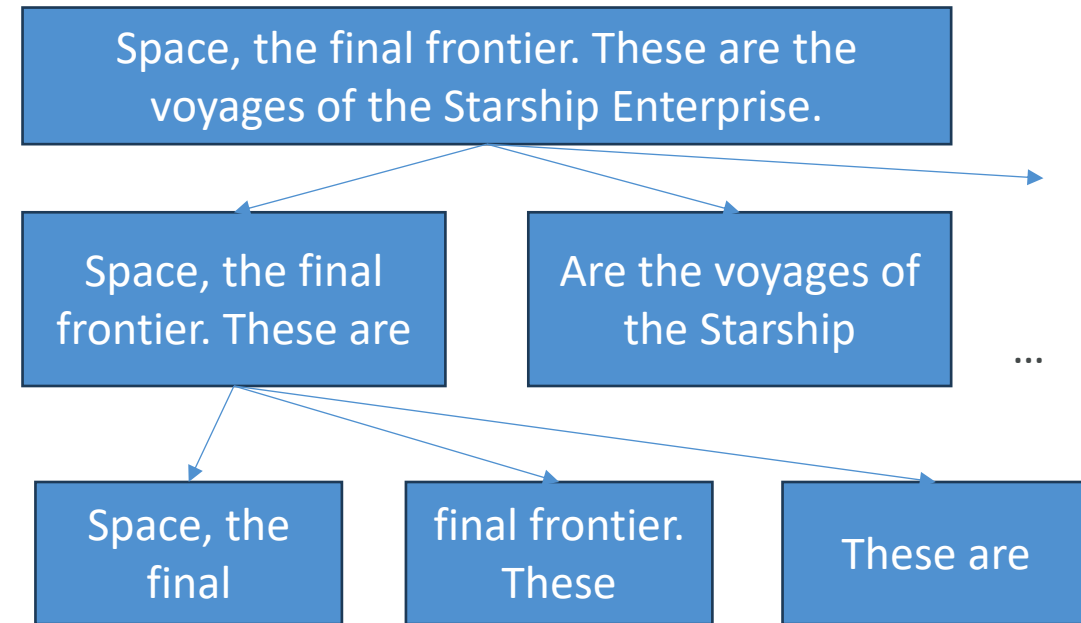
Space, the final frontier. These

These are the voyages of

of the Starship Enterprise.

Chunking with Bedrock: Hierarchical Chunking

- Nested parent/child chunks
- Initial search hits the child chunks
 - But then replaces them with parent chunks to get more context when appropriate
- The idea is to get better precision in retrieval from the smaller embeddings
 - But then reclaim comprehensiveness from the parents



Parent chunk size: 6
Child chunk size: 3
Overlap: 1

*Note we're assuming 1 token = 1 word in this example. That's not necessarily true.

Chunking with Bedrock: Semantic Chunking

- Hits a foundation model to break up chunks based on “semantic content” and not just things like sentences or a fixed chunk size
- You specify:
 - Maximum tokens (while honoring sentence boundaries)
 - Buffer size
 - Number of surrounding sentences per sentence to consider when embeddings
 - i.e., buffer size of 1 results in 3 sentences
 - Too large = introducing noise; too small = missing important context (but more precise)
 - Breakpoint percentile threshold
 - How “semantically similar” a chunk must be
 - Higher = more distinguishable chunks, less of them, larger chunks.
- This costs money; you’re charged for the underlying FM used for this

Space, the final frontier. These are the voyages of the Starship Enterprise.

Space, the final frontier. These are the voyages of the Starship Enterprise.

Well, this is all part of the same idea, so it would probably just keep it all together as long as it’s within your max tokens.

Re-ranker Models in Amazon Bedrock

- Attempts to improve relevance of retrieved results for RAG from your KB
- Calculates **relevance of chunks to the query**, orders results accordingly
- There is a Rerank operation in the API you can use to rank whatever documents you want relative to a query
- Or you can specify a reranker model when hitting your Knowledge Base
- Currently you can choose from Amazon or Cohere models
 - In limited regions



Amazon Bedrock Guardrails

- Content filtering for prompts and responses
- Works with text foundation models
- Word filtering
- Topic filtering
- Profanities
- PII removal (or masking)
- Contextual Grounding Check
 - Helps prevent hallucination
 - Measures “grounding” (how similar the response is to the contextual data received)
 - And relevance (of response to the query)
- Can be incorporated into agents and knowledge bases
- May configure the “blocked message” response





Let's Make a Guardrail



Benefits of Effective Prompting

- Boost a model's abilities and improve safety.
- Augment the model with domain knowledge and external tools without changing model parameters or fine-tuning.
- Interact with language models to grasp their full capabilities.
- Achieve better quality outputs through better quality inputs.



Anatomy of a Prompt

- Instructions
- Context
- Input data
- Output indicator

Instructions:

Write a short dialogue between two friends about their favorite books.

Context:

The two friends are sitting in a cozy café on a rainy afternoon.

Input data:

- Friend 1:** Loves fantasy novels, particularly those involving dragons and magic.
- Friend 2:** Prefers mystery novels, especially those with clever detectives and unexpected twists.

Output Indicator:

The dialogue should be 200 words long and should express each friend's enthusiasm for their favorite genre.

Prompt Best Practices

- Clear and concise
 - Bad: Write something about books. Talk about two people and what they like. Make it interesting.
 - Good: Write a short dialogue between two friends discussing their favorite books.
- Include context
 - Bad: Write a short dialogue...
 - Good: Write a short dialogue to be used in a movie script...
- Specify the desired response type
 - Bad: Write a short dialogue...
 - Good: Write a short dialogue where each line is no more than two sentences long...
- Specify the desired output at the end of the prompt
 - Bad: Write a short dialogue.
 - Good: Write a short dialogue between two friends discussing their favorite books to be used in a movie script, where each line is no more than two sentences long.

Prompt Best Practices

- Phrase your input as a question
 - Bad: Explain the benefits of regular exercise.
 - Good: What are the benefits of regular exercise?
- Provide an example response
 - Bad: Determine the sentiment of the following sentence...
 - Good: Determine the sentiment of the following sentence, using these examples:
“I had a great time at the park today” => positive
“That restaurant had terrible service” => negative
- Break up complex tasks
 - LLMs aren’t good at reasoning (yet)
 - Split up complex tasks into simpler sub-tasks and execute each individually
 - Ask the model if it understands what you’re asking for (although this might become obvious)
 - Sometimes you can ask the model to “think step by step” or to generate sub-tasks
- Experiment, be creative
 - LLM’s are non-deterministic and change often
 - Just try different prompts to see what yields the best results

Types of Prompts

- Zero-shot
 - No examples given, relies on large models that already know about what you're asking for
 - Determine the sentiment of the following sentence...
- Few-shot
 - Provide some examples of the desired responses to given prompts
 - Determine the sentiment of the following sentence, using these examples:
 - “I had a great time at the park today” => positive
 - “That restaurant had terrible service” => negative
- Chain of Thought (CoT)
 - “Think step by step”
 - Describe how to solve a quadratic equation using the quadratic formula. Think step-by-step, starting with the standard form of a quadratic equation, identifying the coefficients, and then applying the quadratic formula. Make sure to include an example equation and solve it.

"First, write the quadratic equation in its standard form: $ax^2 + bx + c = 0$. Next, identify the coefficients a , b , and c . For example, in the equation $2x^2 - 4x - 6 = 0$, $a = 2$, $b = -4$, and $c = -6$. Then, apply the quadratic formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Plug in the values of a , b , and c :

$$x = \frac{-(-4) \pm \sqrt{(-4)^2 - 4(2)(-6)}}{2(2)}.$$

Simplify the expression under the square root and solve for x :

$$x = \frac{4 \pm \sqrt{16 + 48}}{4},$$

$$x = \frac{4 \pm \sqrt{64}}{4},$$

$$x = \frac{4 \pm 8}{4}.$$

This gives us two solutions:

$$x = \frac{4+8}{4} = 3$$

and

$$x = \frac{4-8}{4} = -1.$$

Therefore, the solutions to the equation $2x^2 - 4x - 6 = 0$ are $x = 3$ and $x = -1$."

Prompt Management with Amazon Bedrock

- Reusable prompts may be stored
 - Allows sharing of specialized prompts across applications
- They may be versioned
- Prompts may include variables – placeholders for some value
 - Enclose them in double curly braces
 - *Make me a music playlist for {{genre}} music with {{number}} songs*
 - You can image using this to pass in structured user input to an application.
- Prompt Variants can be used for different models, inference configurations, or whatever
- The Prompt Builder tool in the Bedrock console lets you play with it
- You may associate Tools and Caching with a prompt
- After testing, deployed prompts may be used within a **Flow**
- Quick demo...

Create Prompt

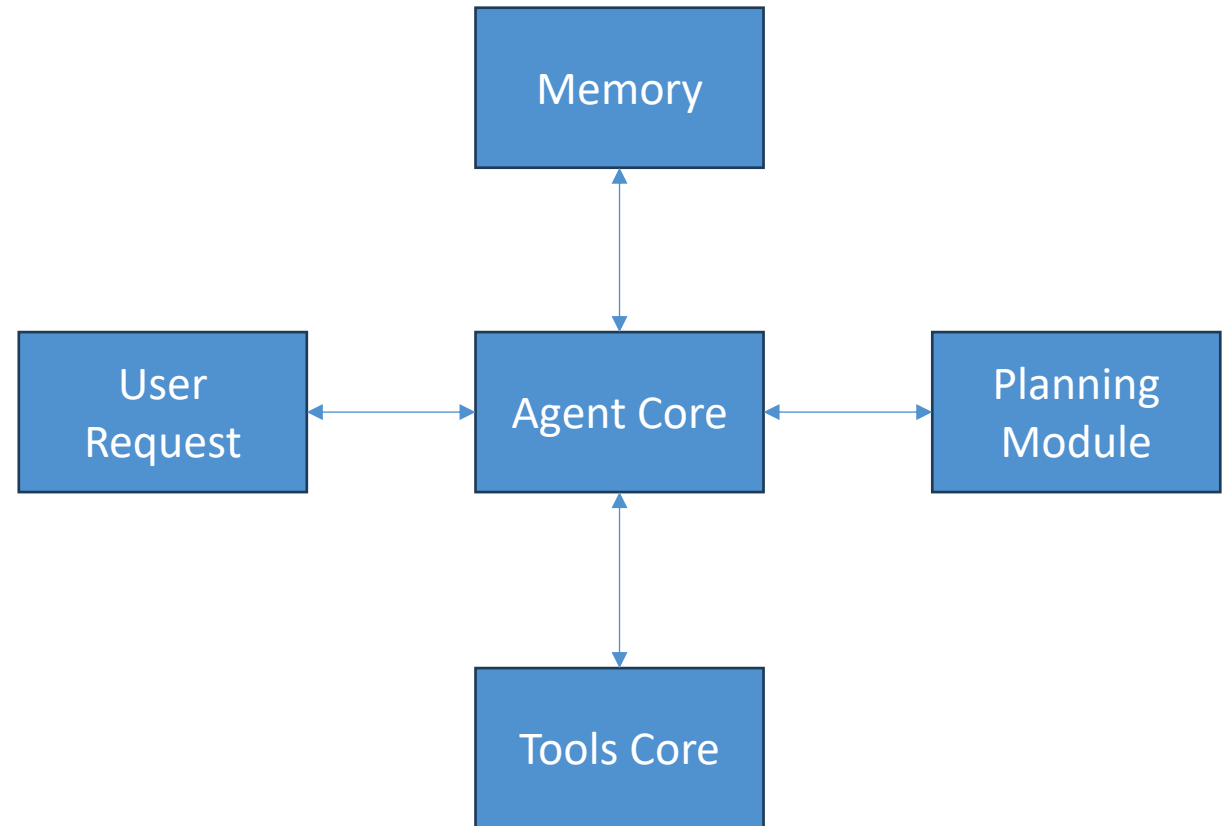
Test Prompt

Use Prompt

LLM Agents

LLM Agents

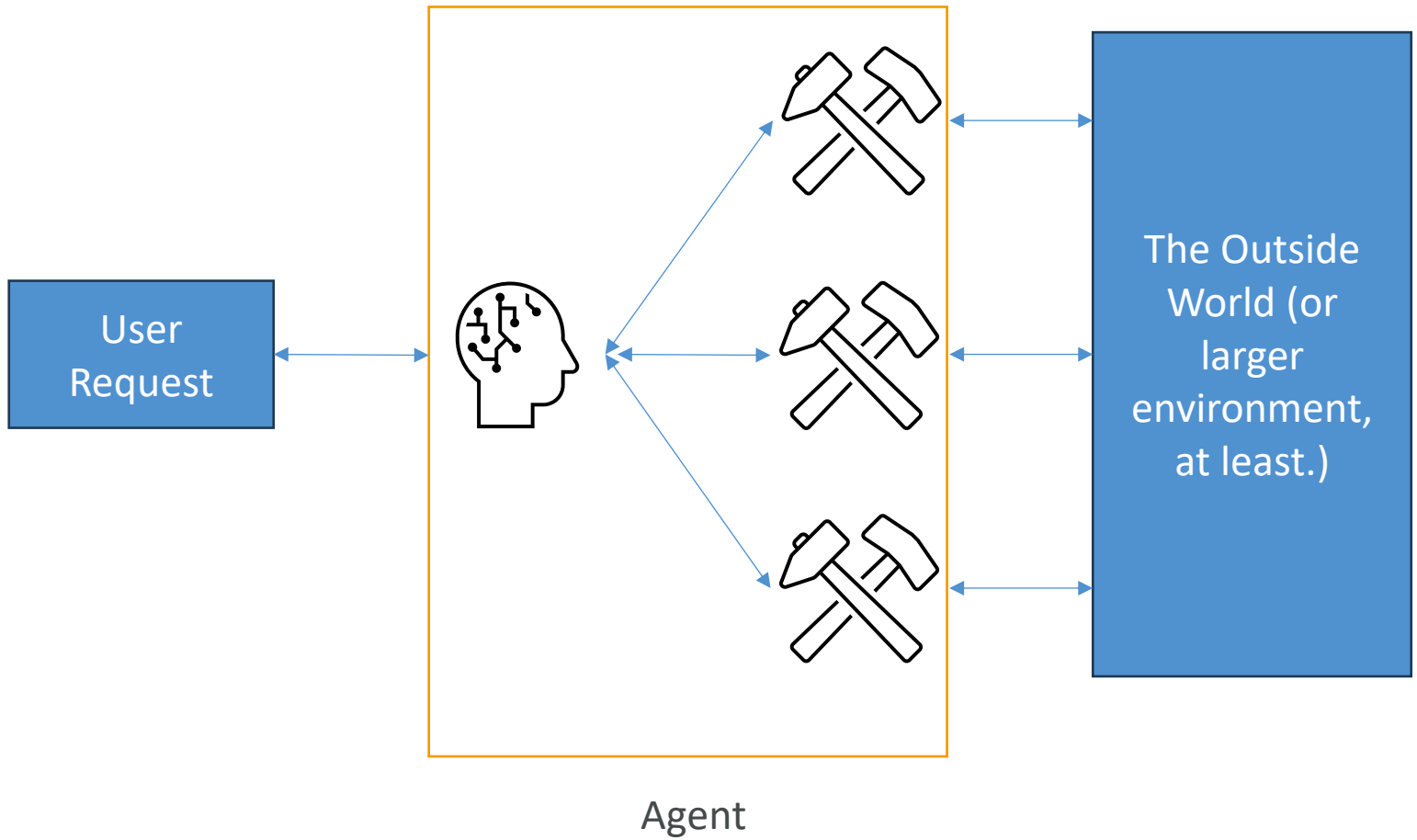
- Giving tools to your LLM!
- The LLM is given discretion on which tools to use for what purpose
- The agent has a memory, an ability to plan how to answer a request, and tools it can use in the process.
- In practice, the “memory” is just the chat history and external data stores, and the “planning module” is guidance given to the LLM on how to break down a question into sub-questions that the tools might be able to help with.
- Prompts associated with each tool are used to guide it on how to use its tools.



Conceptual diagram of an LLM agent, as described by Nvidia
(<https://developer.nvidia.com/blog/introduction-to-llm-agents>)

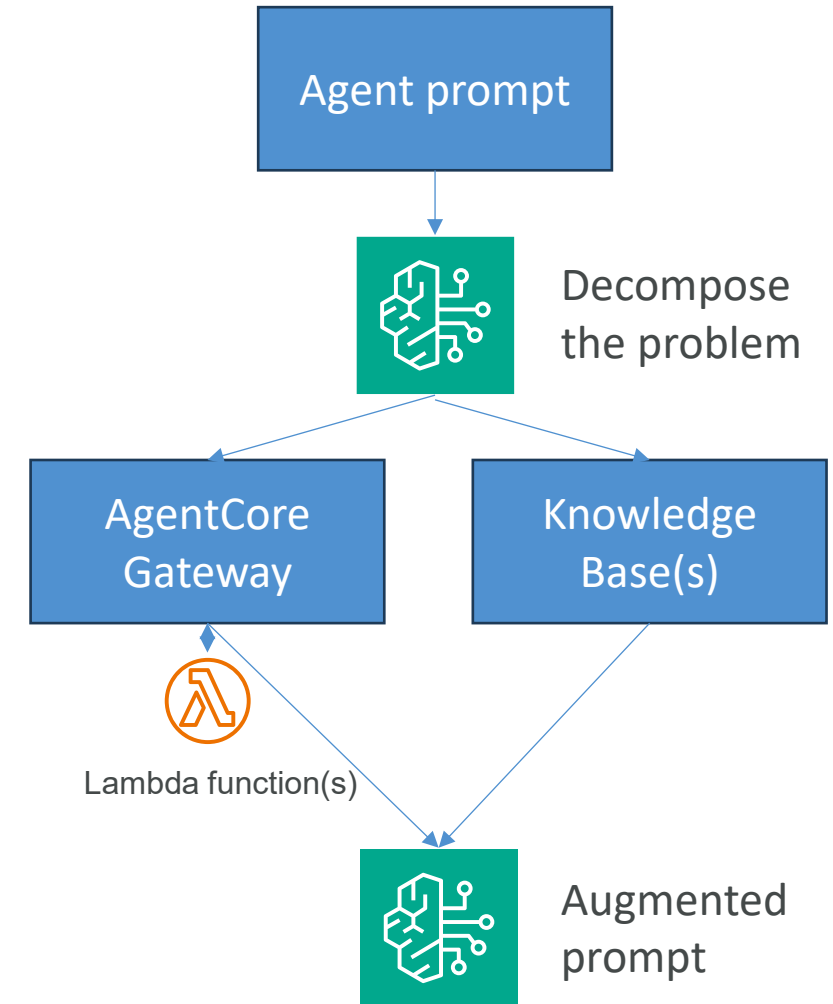
LLM Agents: A More Practical Approach

- “Tools” are just functions provided to a tools API.
 - In Bedrock AgentCore, the Gateway turns Lambda functions and APIs into MCP tools.
- Prompts guide the LLM on how to use them.
- Tools may access outside information, retrievers, other Python modules, services, etc.



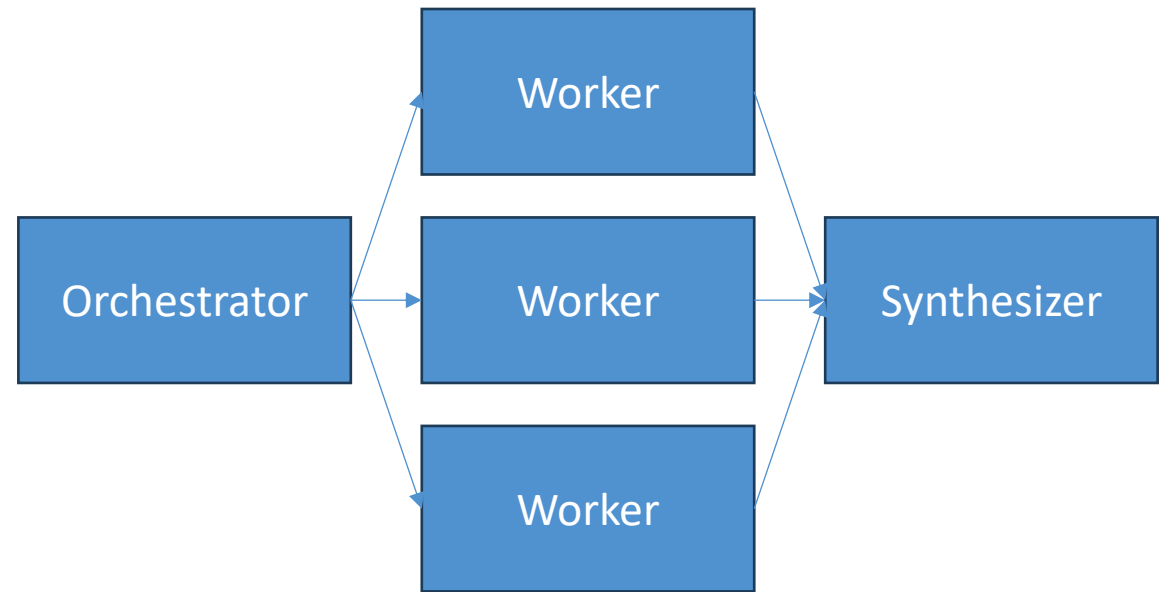
How do Agents Know Which Tools to Use?

- Start with a foundational model to work with
- In Bedrock AgentCore, the Gateway exposes your tools to agents via MCP (Model Context Protocol)
 - A prompt informs the FM when to use it
 - “Use this function to determine the current weather in a city”
 - You must define the parameters your tool (Lambda function) expects
 - Define name, description, type, and if it’s required or not
 - The description is actually important and will be used by the LLM to extract the required info from the user prompt
 - You can also allow the FM to go back and ask the user for missing information it needs for a tool
 - Tools are defined with JSON schemas – or OpenAPI / Smithy specs for API-based Gateway targets
- Agents may also have knowledge bases associated with them
 - Again, a prompt tells the LLM when it should use it
 - “Use this for answering questions about X”
 - This is “agentic RAG” – RAG is just another tool
- AgentCore’s built-in “Code Interpreter” tool allows the agent to write its own code to answer questions or produce charts.



Multi-agent systems: an example

- Similar to just giving tools to a single agent
 - But many tools may be used at once
 - For example, coding agents may need to operate on multiple files in different ways to achieve a larger task
 - Workflows that require complex decision making can benefit from these agents (otherwise, keep it simple with deterministic workflows.)
- The Orchestrator breaks down tasks and delegates them to worker LLMs
- A Synthesizer can then combine their results



For example, this could be a translation tool being asked to translate to multiple languages at once.

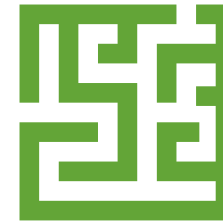
Each LLM here may have its own tools and memory.

Why use multiple agents (or augmented LLMs)?



Too many tools

If one agent has trouble choosing the appropriate tool, multiple specialized agents with specific tools might work better

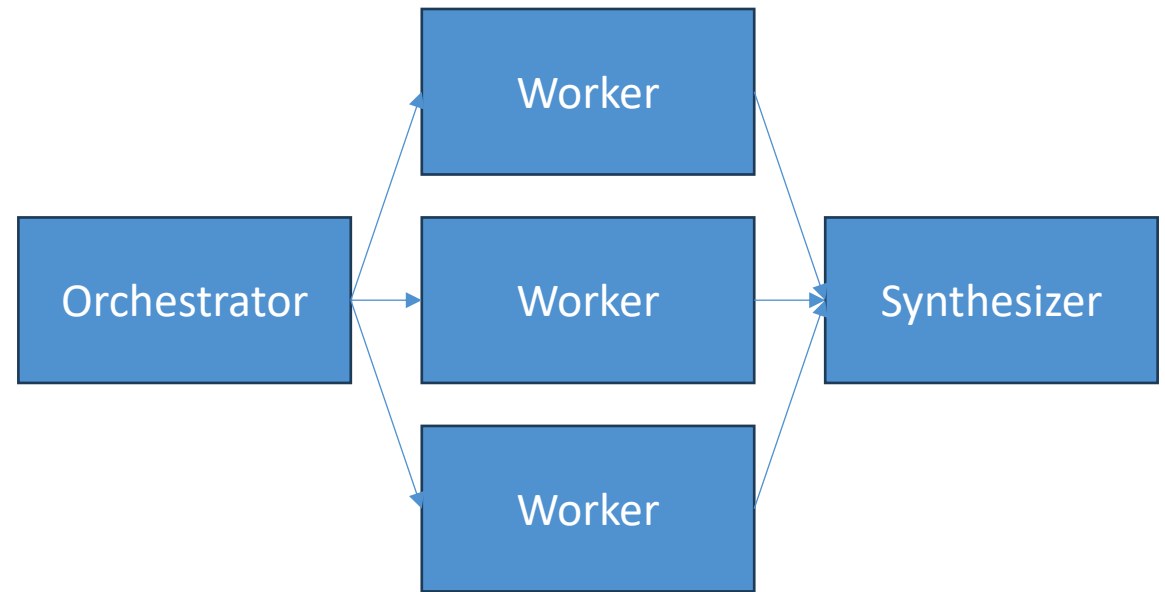


Complex logic

Prompts get too complicated with conditional statements, better to define the flow with specialized agents

Manager (Orchestrator – workers)

- Similar to just giving tools to a single agent
 - But many tools may be used at once
 - For example, coding agents may need to operate on multiple files in different ways to achieve a larger task
 - Workflows that require complex decision making can benefit from these agents (otherwise, keep it simple with deterministic workflows.)
- The Orchestrator breaks down tasks and delegates them to worker LLM's

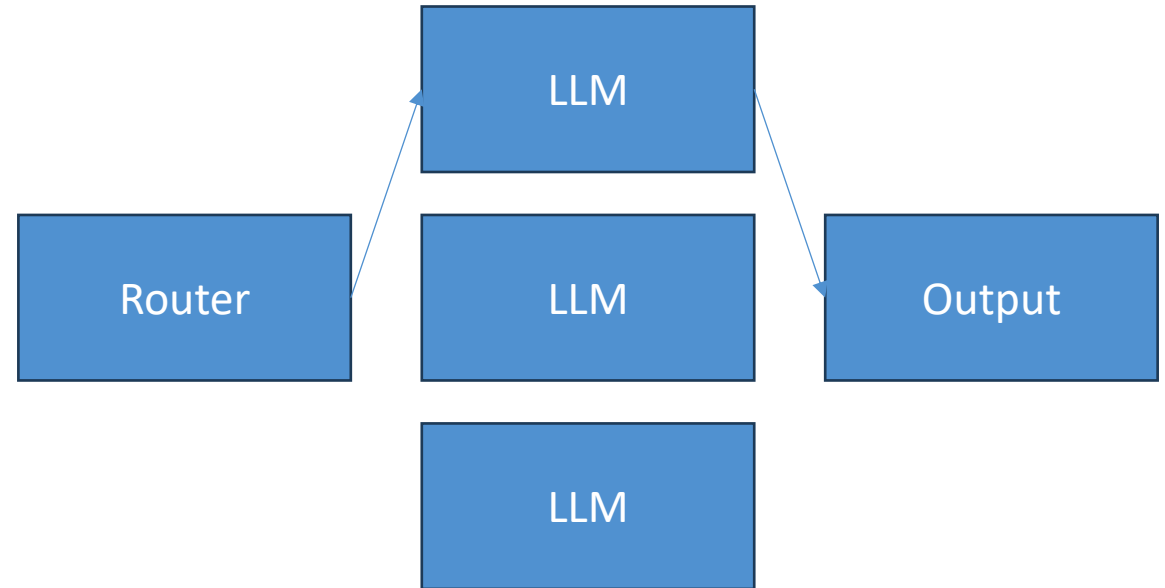


For example, this could be a translation tool being asked to translate to multiple languages at once.

Each LLM here may have its own tools and memory.

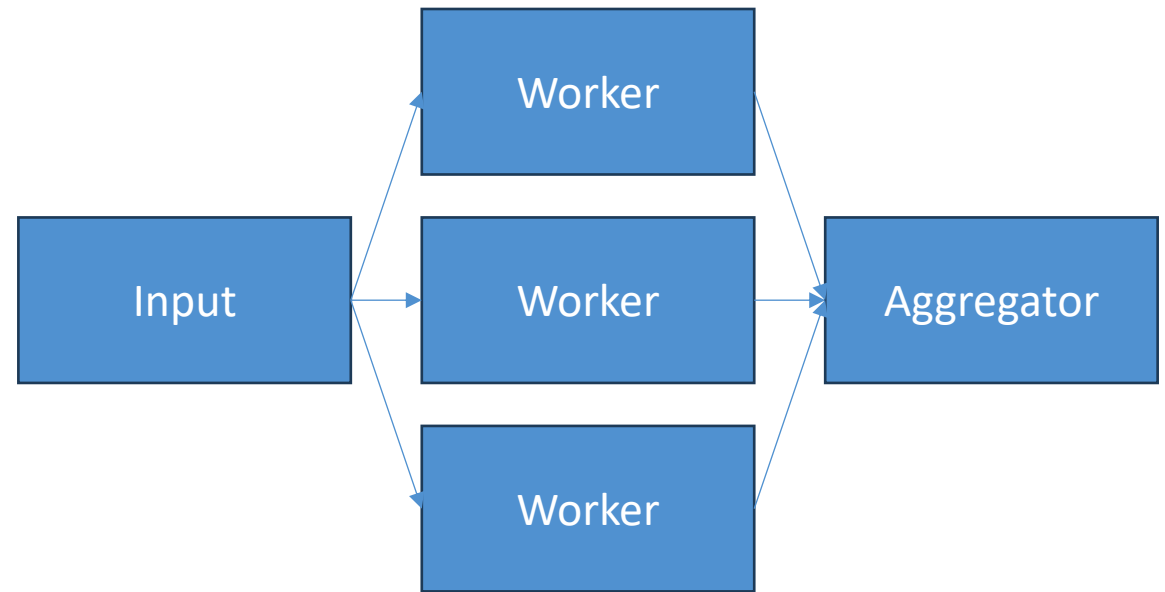
Routing

- A “router” LLM chooses just one of many specialized agents to call in turn
- Use for complex tasks where a single classification is needed
 - i.e., how large of a model is needed for the task
 - What kind of customer service query is being received



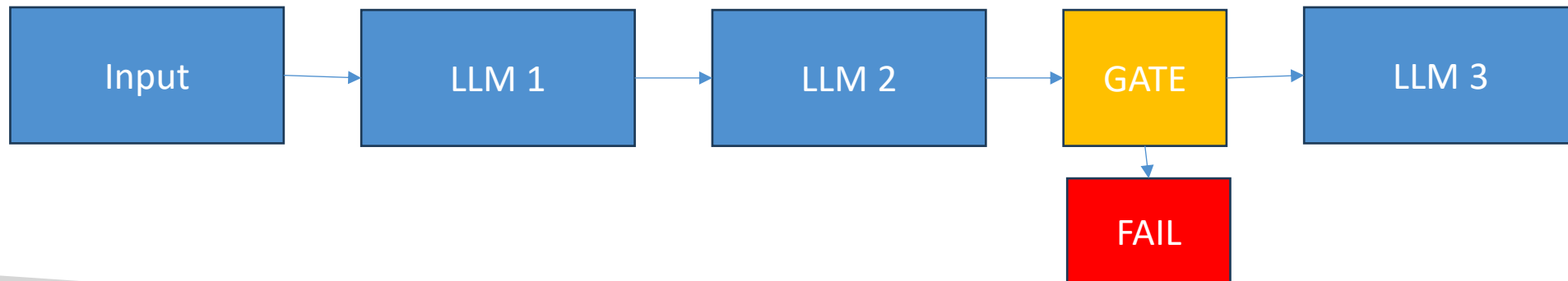
Parallelization

- A lot like the orchestrator
 - But without complex orchestration
- Sectioning: independent subtasks run in parallel
 - Running multiple guardrails at once
 - Running multiple evaluations at once
- Voting
 - Do the same thing with different models or prompts
 - Let them vote on the right result



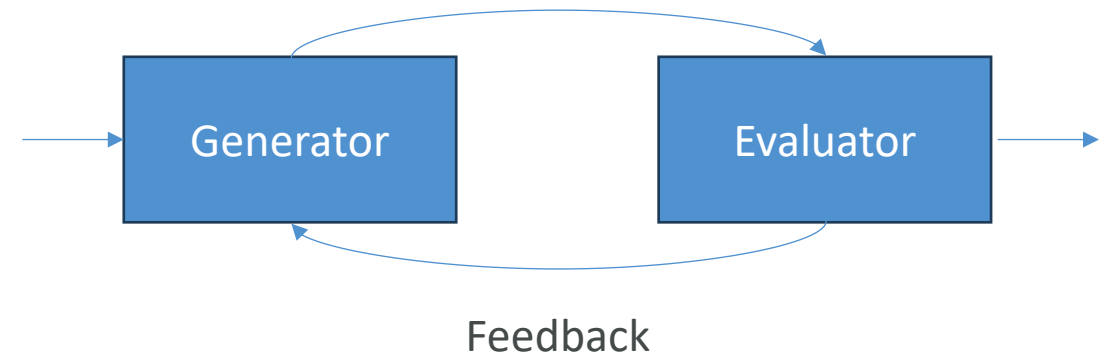
Prompt Chaining

- Use when a task has a discrete sequence of known steps
- Each LLM processes the output of the previous one
- Can add “gates” to keep things on track as you go, or exit early
- Examples
 - Write a document, then translate it to different languages
 - Write an outline, then make slides



Evaluator - Optimizer

- One LLM generates a response
- Another evaluates that response
- If it's not good enough, provide feedback and try again
- Examples
 - Code reviewing code and incorporating those recommendations
 - Complex search that requires multiple rounds



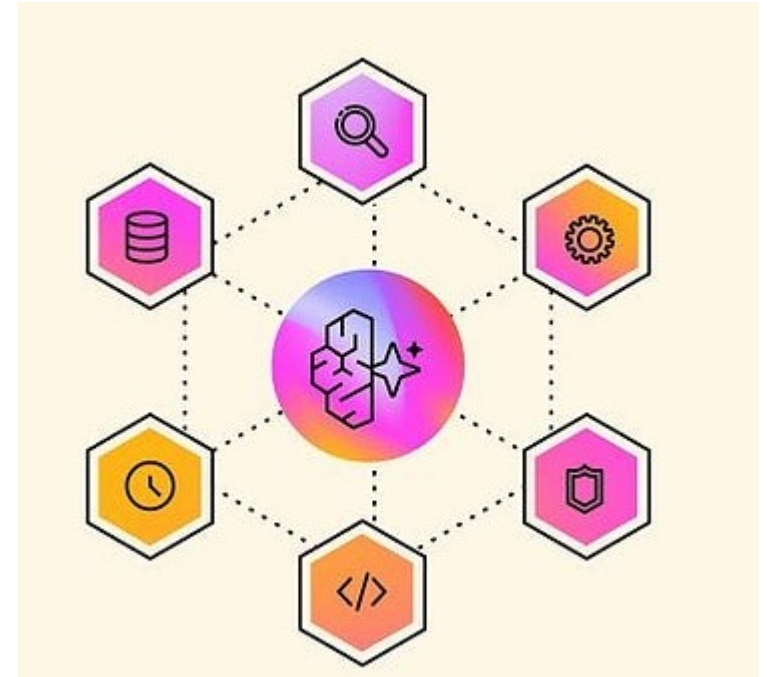
Adding Memory

- Short-term
 - Chat history within a session / immediate context
 - Enables conversations
 - Events within sessions
 - This could be in-memory or in a big distributed cache
 - Elastically, MemoryDB, even DynamoDB is OK
- Long-term
 - Stores “extracted insights”
 - Summaries of past sessions
 - Preferences (your coding style and favorite tools, for example)
 - Facts you gave it in the past
 - “Memory Records” that store structured information derived from agent interactions
 - “Strategies” for user preferences, semantic facts, session summaries
- This all needs to be stored somewhere!
 - Long-term: DynamoDB, SQLite, RDS, Aurora, etc.
 - AgentCore Memory (we’re getting there)
 - Mem0 (Strands uses this)
 - You could frame Knowledge Bases as a form of long-term memory too.

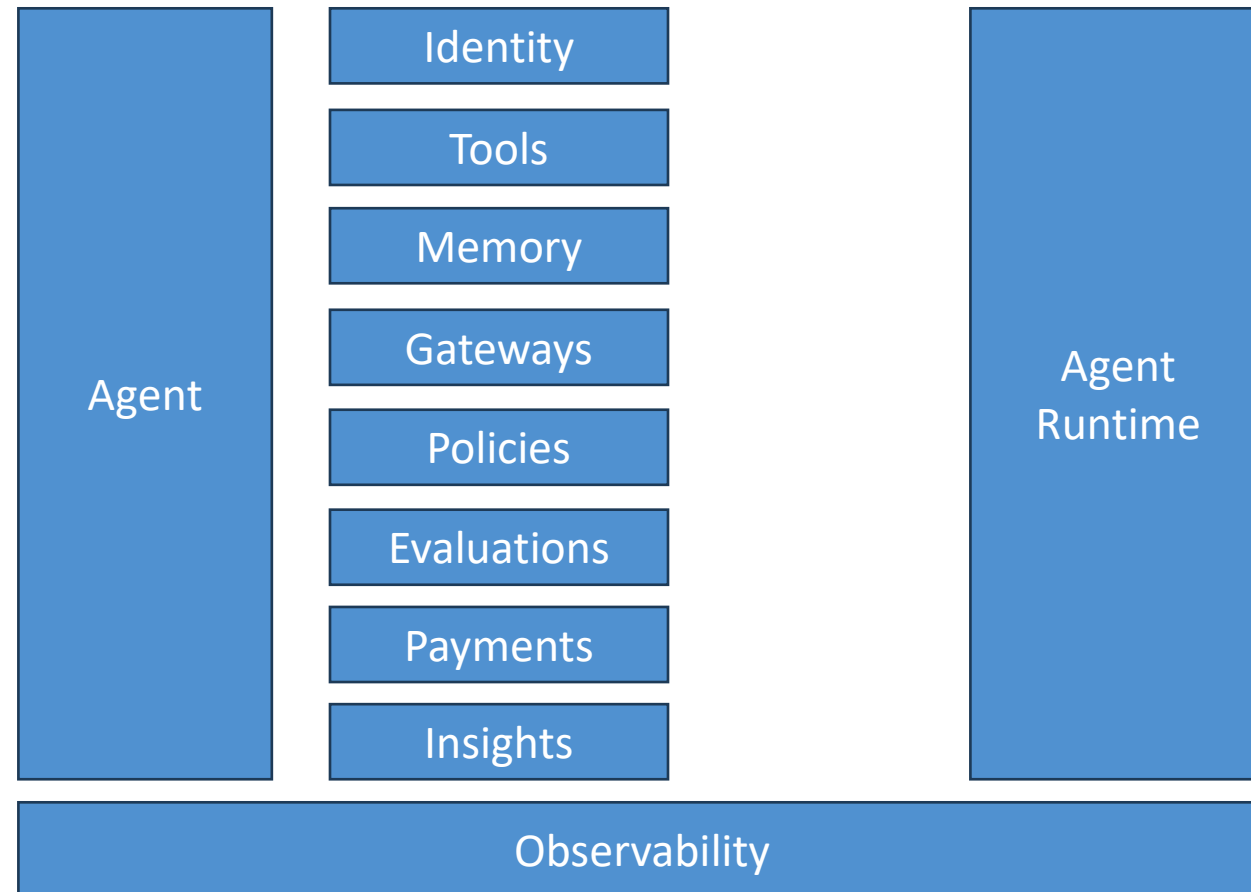


Amazon Bedrock AgentCore

- Hands deployment and operation of AI agents at scale
 - No fussing with Docker, ECR, ECS, etc
 - Serverless
- And it works with ANY agent framework
 - You aren't tied to Bedrock agents
 - You aren't even tied to AWS's Strands framework
 - OpenAI Agent SDK, LangGraph / LangChain, CrewAI, Google ADK, whatever
 - (although... Strands does get better support...)
- Includes a “agentcore CLI” to make deployment of agents to AWS super easy
- Includes several tools and capabilities you can include in your agents
 - These can be contained in a “harness” without writing code
- Integrates with Bedrock Knowledge Bases

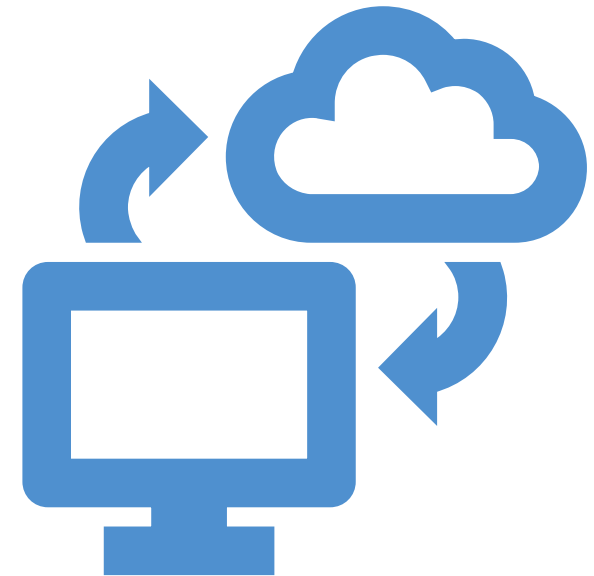


AgentCore Capabilities



AgentCore Agent Runtime

- Serverless endpoints
- Deploy your agent to ECR, enhanced with AgentCore capabilities
 - Agentcore CLI manages it all for you, using CodeBuild
 - But you can build your own Docker containers if you want
- Can have multiple endpoints
- Observability dashboard lets you track usage and performance
 - Through “GenAI Observability” feature in CloudWatch



Adding AgentCore Memory

- Short-term
 - Chat history within a session / immediate context
 - Enables conversations
 - API centered around Session objects that contain Events
- Long-term
 - Stores “extracted insights”
 - Summaries of past sessions
 - Preferences (your coding style and favorite tools, for example)
 - Facts you gave it in the past
 - API involves “Memory Records” that store structured information derived from agent interactions
 - “Strategies” for user preferences, semantic facts, session summaries
- This all needs to be stored somewhere!
 - The OpenAI Agents SDK gives you a SQLite implementation
 - But maybe you need something that scales better, and is serverless
 - Enter AgentCore Memory

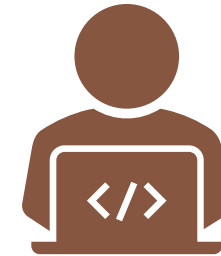


AgentCore Built-In Tools



Browser Tool

Allows control of your browser to interact with the web



Code Interpreter

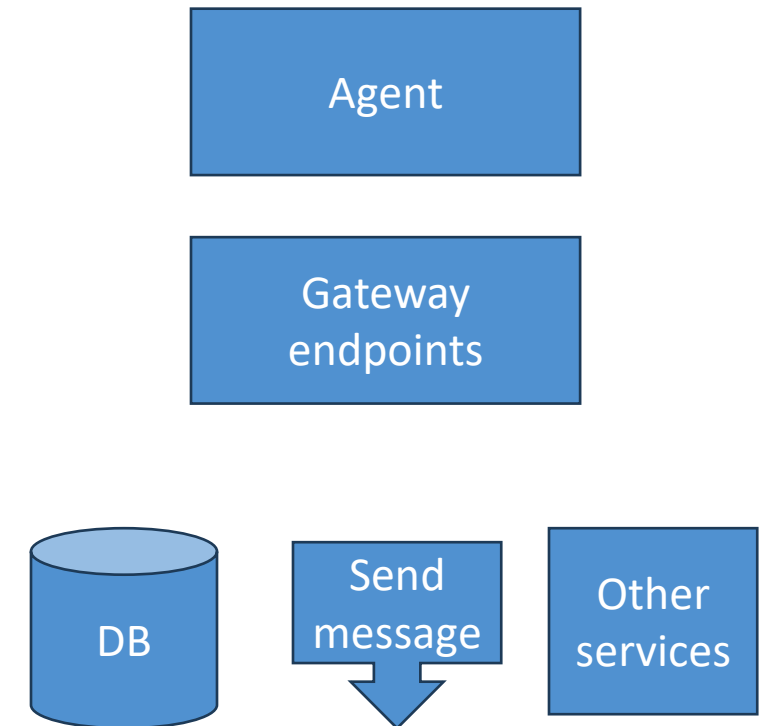
Lets you run code (in an isolated container)
Python, JavaScript, TypeScript

Importing Bedrock Agents

- Bedrock has its own system for endpoints...
 - But Bedrock Agents is deprecated for new customers.
 - AWS recommends migration to AgentCore
- Building agents in Bedrock is super easy
- Importing is just a matter of running `agentcore create -type import`
 - Specify your agent-id, agent-alias-id, region
 - This generates Strands code (or LangChain / LangGraph) in an output directory
 - From there you can test or deploy it like any other AgentCore agent

AgentCore Gateway

- Addresses the problem of using external tools at scale
- Convert APIs, Lambda functions, or other services into MCP tools
 - Targets can be OpenAPI (REST), Smithy models (that's an AWS thing), or Lambda
- Agents can then access them through Gateway endpoints
 - Manages OAuth security / credentials
- Semantic tool selection



AgentCore Identity

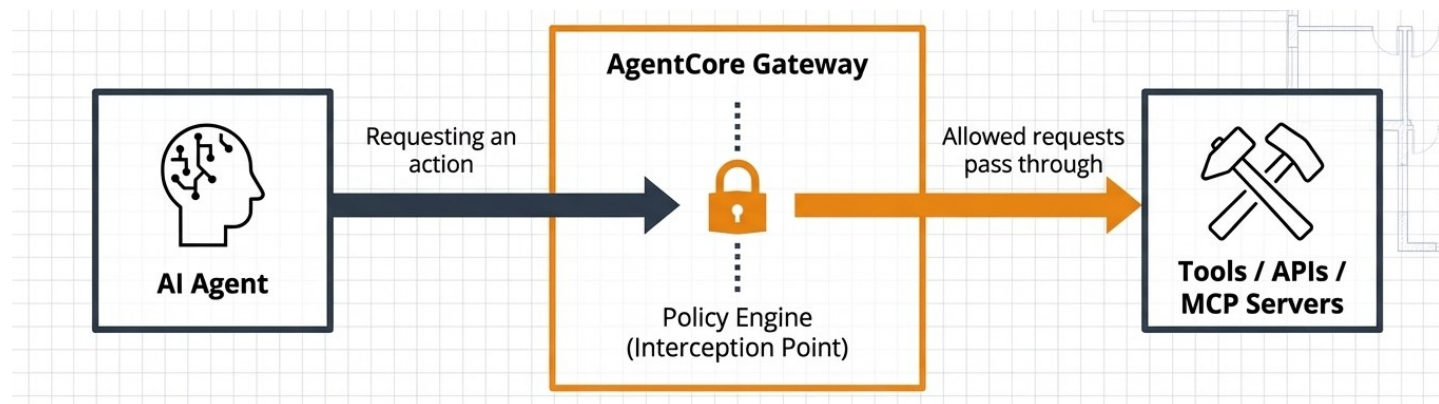
- This is different from OAuth identity for users and connecting to services we talked about earlier
- This is about your agent's identity / identities
- Secure access to external tools and AWS services
- Central repository for all of your agent identities
 - Similar to a Cognito user pool
- Secure credential storage
- OAuth 2.0 support
 - Built-in support for Google, GitHub, Slack, Salesforce, Atlassian
- There is a lot of depth to this, but you probably don't need it right away
 - Refer to <https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/identity.html>



Policy in AgentCore

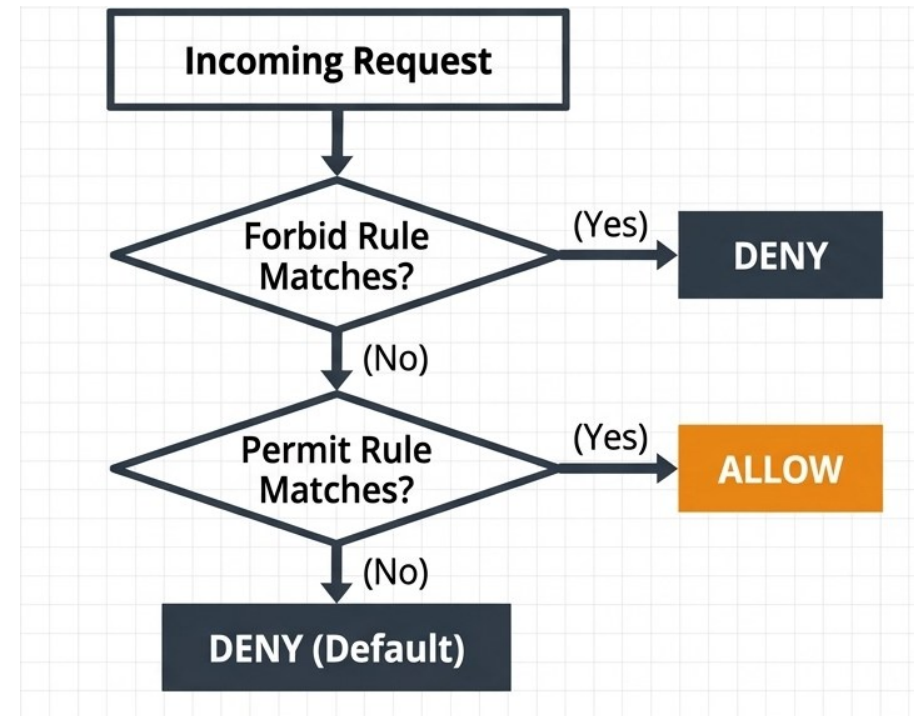
- Integrates with AgentCore Gateways
- Intercepts tool calls, evaluates against your defined policies
- Defined in the Cedar language
 - You can also construct policies using NLP or via a form
- Very fine-grained control – who can do what under what conditions
- Evaluated outside of the agent's code

```
permit (  
  principal == Role::"refund-agent",  
  action == Action::"process_refund",  
  resource  
) when { context.amount < 200 };
```



Policy Enforcement

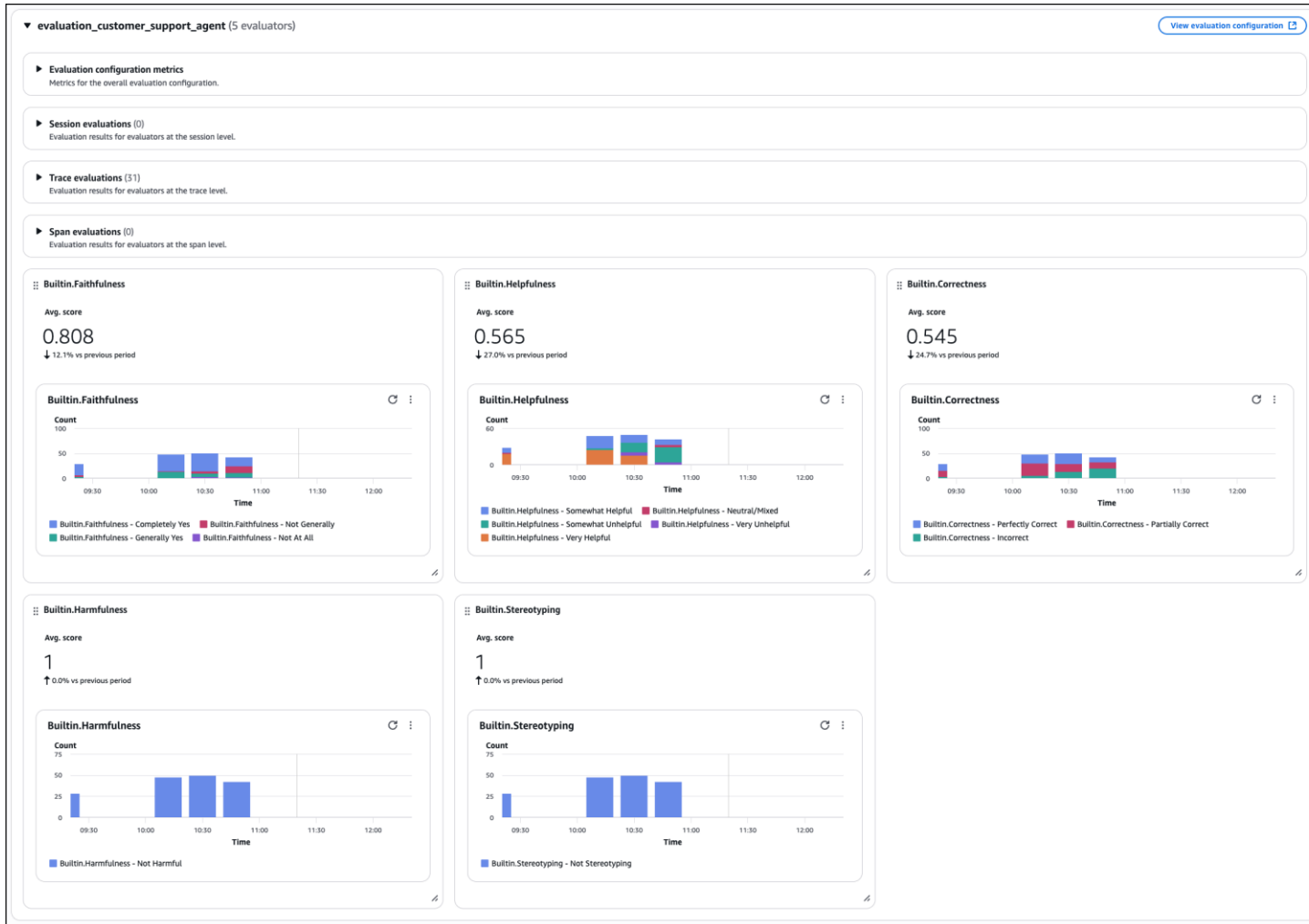
- Deny by default
 - All actions denied unless explicitly permitted
- Contextual validation
 - Validate full context, OAuth sub-claims, specific tool arguments
- Dual modes
 - Enforce mode to actively block actions
 - Log-only mode to test and validate policies



AgentCore Evaluations

- Assess how well your agent:
 - Performs tasks
 - Handles edge cases
 - Maintains consistency
- Integrates with:
 - Strands
 - LangGraph
 - OpenTelemetry
 - OpenInference
- Visualize results in CloudWatch
 - AgentCore Observability Insights
 - The usual CloudWatch alerts / alarms
- Cross-region inference
 - Automatically selects optimal region for processing
 - Data remains resident to where the request originated
 - But may be transmitted and processed – encrypted – to other regions.





Sample CloudWatch Visualization

Some Built-In Evaluations (prompt templates)

- Correctness (is it factually correct)
- Helpfulness (from the user's standpoint)
- Conciseness
- Instruction following
- Faithfulness (is response supported by context/sources)
- Response relevance (to user query)
- Coherence (logical structure)
- Refusal (is it evasive)
- Goal success rate
- Tool selection and parameter accuracy
- Harmfulness, Stereotyping
- Evaluation may be limited by filters and sampling rate you specify



Custom Evaluators

- Specify a model to use as a judge
 - And configure it
- And define it with a prompt
 - Maybe start with a built-in one
- Define scales
 - Maps ranges of results to labels
- Define evaluation level
 - Session, trace, tool call

The screenshot shows the 'Create custom evaluator' interface. It is divided into two main sections: 'Custom evaluator details' and 'Custom evaluator definition'.

Custom evaluator details:

- Evaluator name:** A text input field containing 'evaluator_test'. Below it, a note states: 'Must start with a letter (a-z, A-Z) and contain only letters, numbers, and underscores. Maximum 48 characters.'
- Additional configurations - optional:** A section for optional settings.

Custom evaluator definition:

- Instruction:** A text area for writing a tailored prompt. A blue notification box at the top says: 'The template for Faithfulness has been loaded. You can load a different template to get started.' with a 'Load template' button. The instruction text reads: 'You are an objective judge evaluating an AI assistant's response quality as to faithfulness, i.e. judge whether the AI assistant's response conflicts with the conversation history. The conversation history has multiple turns including:
- User messages (User:)
- Assistant responses (Assistant:)
- API calls/actions taken by the assistant (Action:)
- Tool outputs (Tool:)
Focus your evaluation on the last assistant message in the conversation history.'
- Configurations:**
 - Model:** A dropdown menu showing 'Nova Lite'.
 - Inference parameters:** A section with a 'Reset' button.
 - Randomness and diversity:** Includes 'Set temperature' and 'Set top P' with radio buttons.
 - Length:** Includes 'Set max. output tokens' and 'Set stop sequences' with radio buttons.

At the bottom left of the instruction text area, it says '532/5000 characters'.

Handling AgentCore Failure Modes



Coordination Failures

Multi-agent workflows stall or lose state mid-handoff

- Checkpoint state in AgentCore Memory
- Time out & retry idempotent handoffs
- Supervisor agent detects stalled workers



Truncated Streaming

Streamed responses cut off mid-generation

- Check the stop reason to detect truncation
- Buffer events; resume from the last chunk
- Fall back to a non-streaming retry



Tool Failures

Gateway tool calls error out or time out

- Return structured errors the LLM can act on
- Retry with exponential backoff
- Circuit breakers & fallback tools

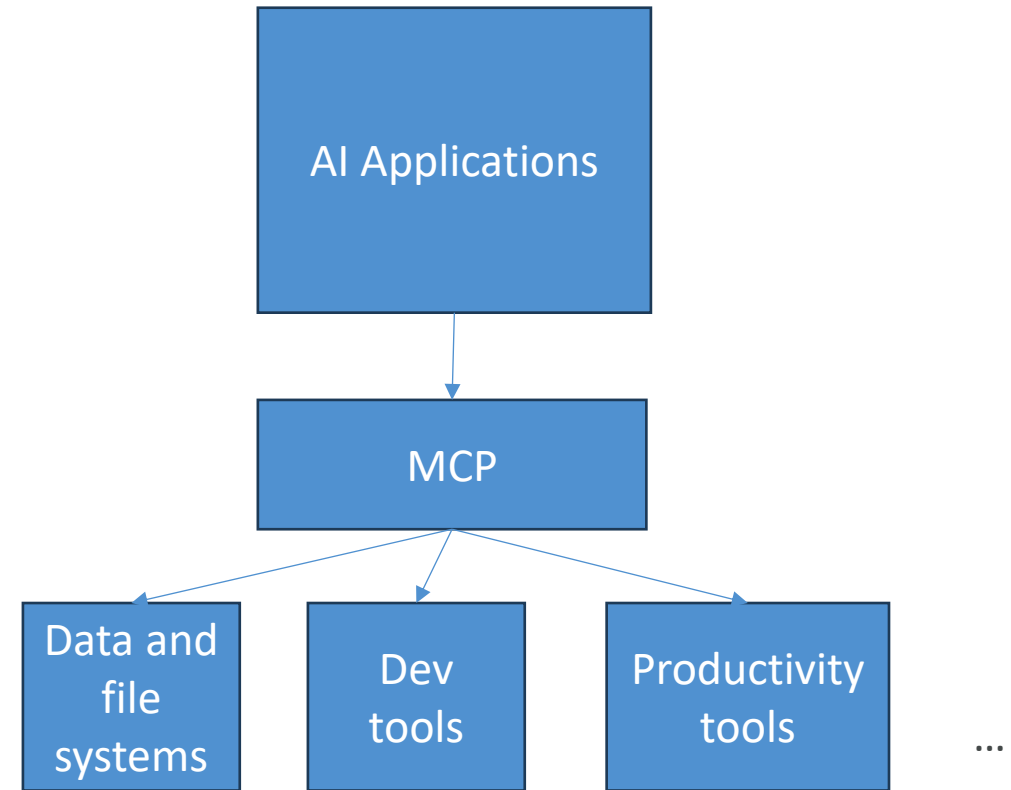
Observability is your safety net: trace every step (CloudWatch / OpenTelemetry), alarm on error rates, and keep tools idempotent so retries are always safe.

More Bedrock Features

- Imported models
 - Import models from S3 or SageMaker
- Model evaluation
 - Automatic: accuracy, toxicity, robustness, BERTScore, F1, etc.
 - Test against your own prompts or built-in prompt datasets
 - Human: Bring your own team
 - Human: AWS-managed team
- Provisioned throughput
- Watermark detection
 - Detects if an image was generated by Titan
- Bedrock Studio
 - Creates a web app workspace for Bedrock without AWS accounts
 - Uses Single Sign On integration with IAM and your own Identity Provider (IDP)
 - Users can collaborate on projects and components

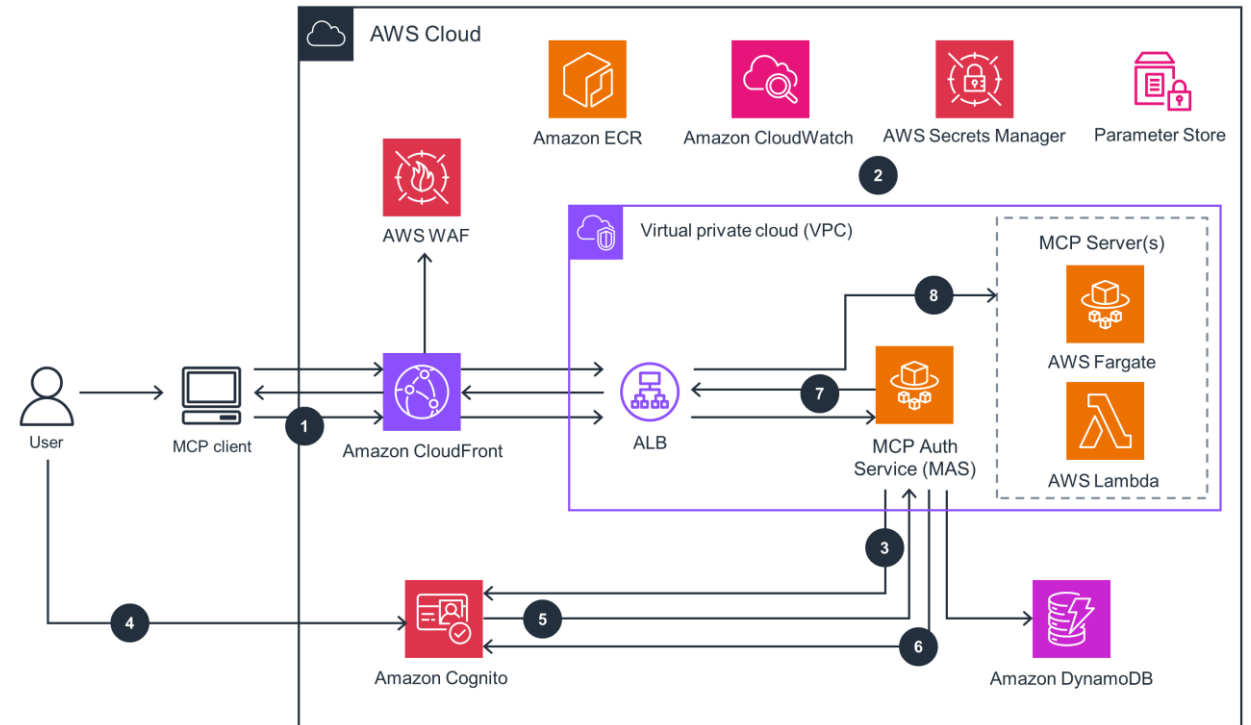
Model Context Protocol (MCP)

- Standardized interface for agent-tool interactions
 - “Like a USB-C port for AI Applications” – Anthropic
 - Technically the data layer is JSON-RPC 2.0
 - Transport is stdio or HTTP streaming
- Access external tools with consistent interaction patterns
 - Standardizes how apps provide context to LLMs
- Pretty much any agent SDK can call external MCP servers (including Strands)
- Servers may expose *tools*, *resources*, and *prompts*
- Some popular MCP servers:
 - GitHub, Atlassian, PostgreSQL, Slack, Google Maps... even Udemy!



Deploying your Own MCP Server

- For stateless, lightweight access:
 - AWS Lambda
- For complex tools:
 - Amazon ECS
 - Or Fargate
- API Gateway
 - Can also expose MCP endpoints
- Use MCP client libraries
 - Abstract away the complexities of the protocol, don't re-invent the wheel!
- AWS's official MCP architecture guidance is on the right here
 - But it's different from what the exam guide and SkillBuilder talk about :/
- You can also deploy MCP servers with AgentCore



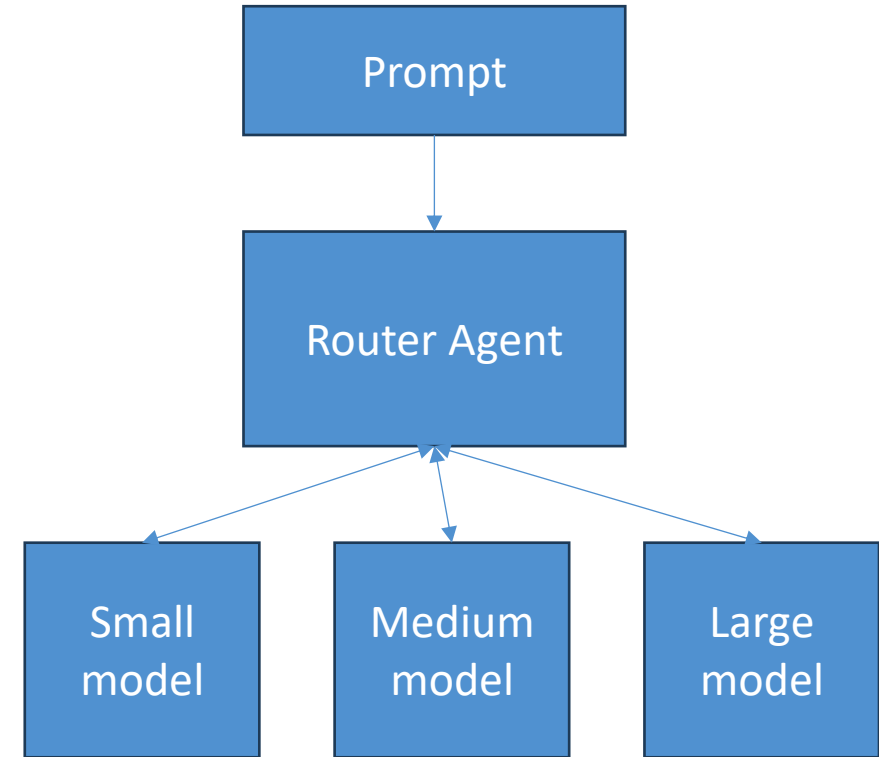
Humans in the Loop

- Can be low-tech
 - “Human augmentation pattern” – AI prepares drafts, humans refine
- AI systems can implement escalation criteria
 - Route complex or uncertain cases to human experts
 - Based on confidence scores or risk assessments
- Human feedback enables continuous improvement
 - Front feedback collection with API Gateway to just ask humans if they like the output
 - Store that feedback in DynamoDB
 - With indexing to allow later analysis
 - Use to measure which model / variant is preferred



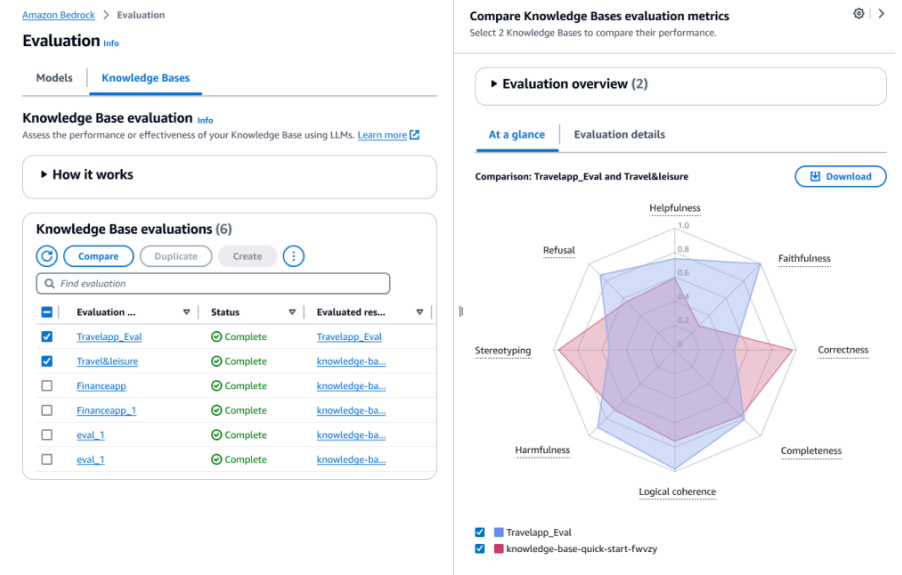
Cost-Effective Model Selection

- Cost / Capability Tradeoff
 - Do you really need the largest, most expensive model?
 - If the “smarts” are in RAG and tool usage, maybe a smaller model will do just fine
 - Smaller models may also play a role in pre-processing
 - Summarization, compression, classification, chunking – doesn’t need PhD-level smarts
- Tiered usage based on query complexity – Dynamic Routing
 - AKA **Intelligent Prompt Routing**
 - (This is now a feature in Bedrock as well)
 - Route to different models based on the complexity of the query
 - (This is how ChatGPT works under the hood...)
 - Many ways to route
 - A conditional Flow; Lambda; Agent Squad; Strands
 - This is a great application of agents – with a tool to score “complexity”



Measuring Price : Performance Ratio

- You can use tools like **Amazon Bedrock Evaluations** to measure Foundation Model performance against your prompt dataset
 - This can use human or LLM judges, or just predefined metrics
 - The **compare** feature in Evaluations lets you visualize the performance of different models with your prompts
- Pair that with token counting to estimate costs (or track actual costs)
- Now you can make informed tradeoffs...



Foundation model evaluation criteria

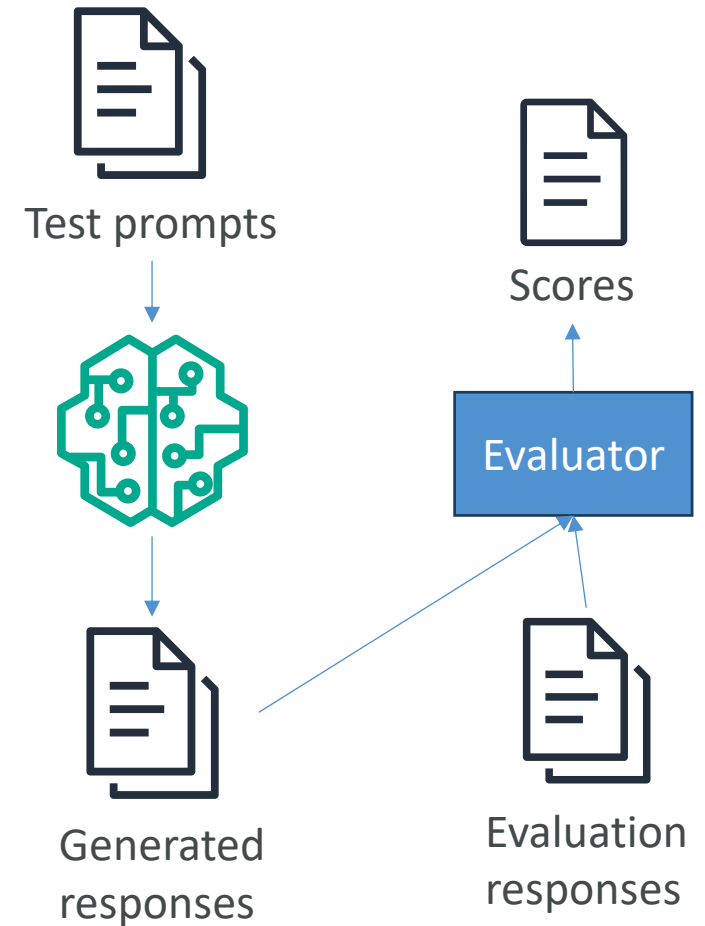
Human Evaluation

- The complex and non-deterministic nature of generative AI makes it hard to measure.
- Humans are often your best source of feedback
 - UX: How does it feel to use this model?
 - Contextual relevancy / sensitivity
 - Creativity & flexibility
 - Handling unexpected or complex problems



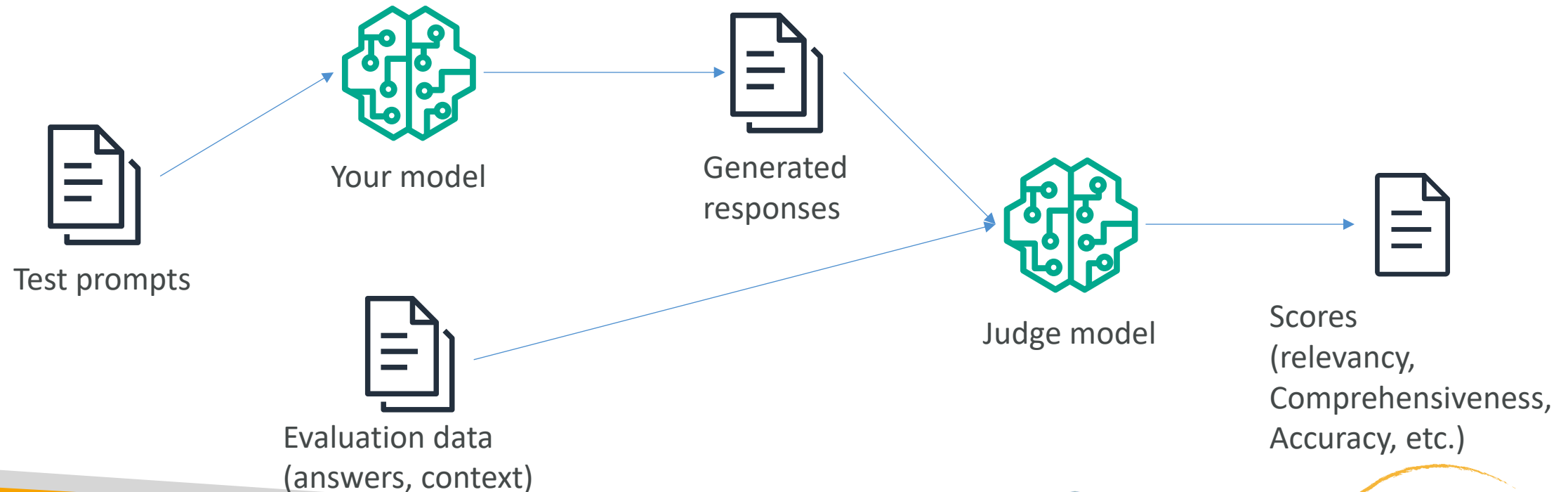
Evaluation against Benchmark Datasets

- Generally a set of sample questions (prompts) and answers generated by SME's
 - Might test performance for a specific application
 - Might test performance for specific kinds of problems (like reasoning ability)
- Can be useful as initial metrics before getting human feedback
- Often used for FM “leaderboards”
- Things you can measure
 - Accuracy
 - How close was the generated result to the “ideal” one?
 - Speed / Efficiency
 - How quickly was the result generated?
 - Scalability
 - How does it hold up to more data? More users? Longer prompts? More context?
 - Context retrieval
 - We’re getting there... but if the system retrieves external data, how relevant is that data?
- Only as good as your test and evaluation data... these metrics can be flawed and incomplete



Using another model as a “judge”

- A “trusted” model is used to evaluate responses from your test prompts
 - This assumes you have a “trusted” model (who watches the watchers?)
 - If your model and the judge model share some of the same problems, that could be a problem.
 - Perhaps better for testing “small” language models designed for efficiency





Hybrid approaches

- Why not both?
 - Comparing human and benchmark-based approaches can reveal the limitations of both

ROUGE

- Set of metrics for:
 - Text summarization
 - Machine translation
- Counts number of overlapping “units” between computer-generated output and evaluation ground-truth output.
 - Words, n-grams, sentence fragments
- ROUGE-N
 - Overlap on n-grams (how many words at a time are compared)
 - ROUGE-1: unigrams (one word,) ROUGE-2: bigrams (two words)
- ROUGE-L
 - Uses longest common subsequence between generated and reference text
 - Good at evaluating coherence and order of narrative

Certified AI Practitioner

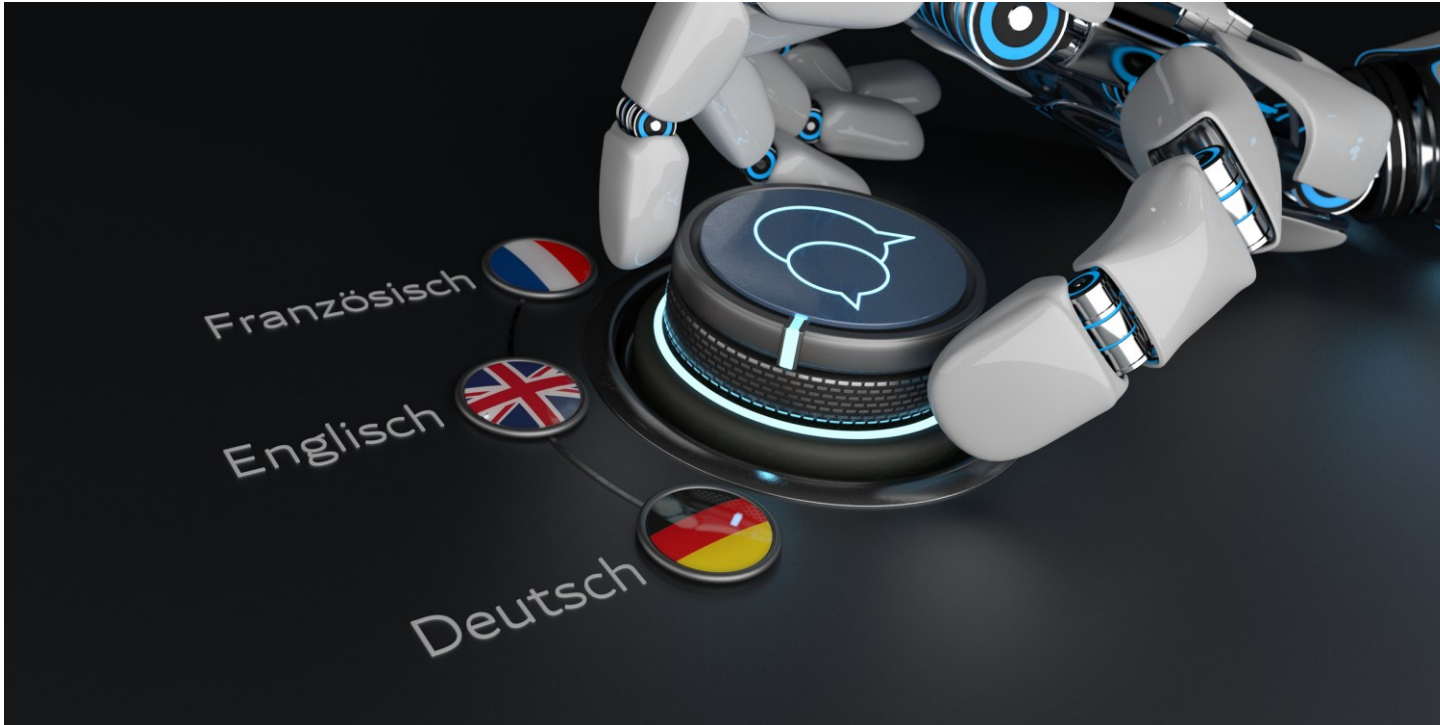
Unigrams: **Certified, AI, Practitioner**

Bigrams: **Certified AI, AI Practitioner**

Output: Certified ML Practitioner

ROUGE-1: 2/3

ROUGE-2: 0

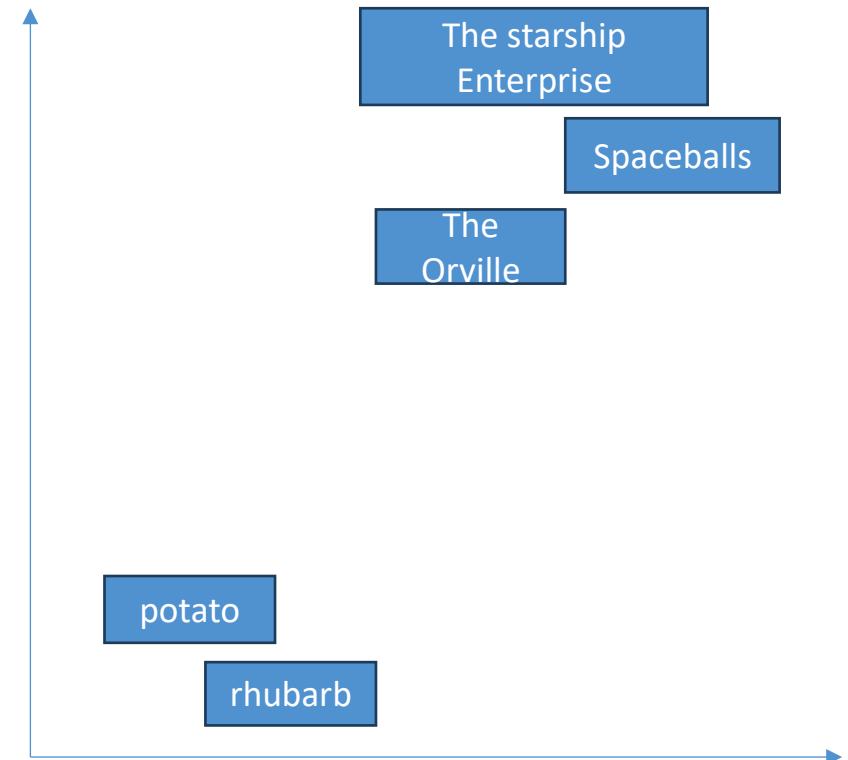


BLEU

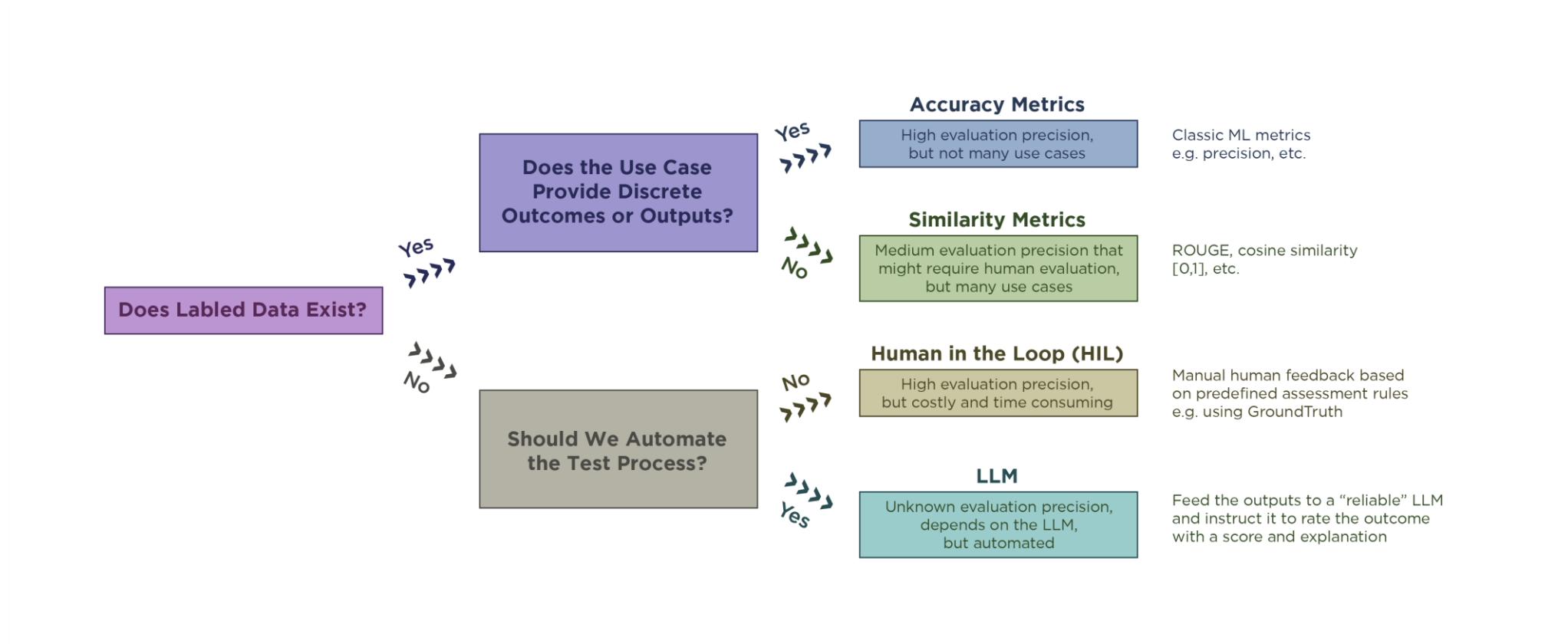
- Used for machine translation
- Compares machine translation to human translation
- Measure PRECISION of N-grams
 - As opposed to ROUGE, which is recall
 - Precision compares to everything, recall just to “relevant” results (overlapping results in the case of ROUGE)
 - Checks how many words / phrases appear in the reference translation
 - Works at the sentence level
- Brevity penalty
 - Overly short translations are penalized
- Limited use in assessing fluency and grammar

BERTscore

- LLMs rely on “embeddings” that store the meaning of a word as a vector
 - I am oversimplifying, but that’s the idea.
- BERTscore uses these embedding vectors to compare the semantic similarity between a model’s output and the ideal output
 - BERT is a language model that predated GPT
- Less sensitive to synonyms and paraphrasing that don’t really change the meaning of what’s being generated



Testing and evaluation: choosing a strategy



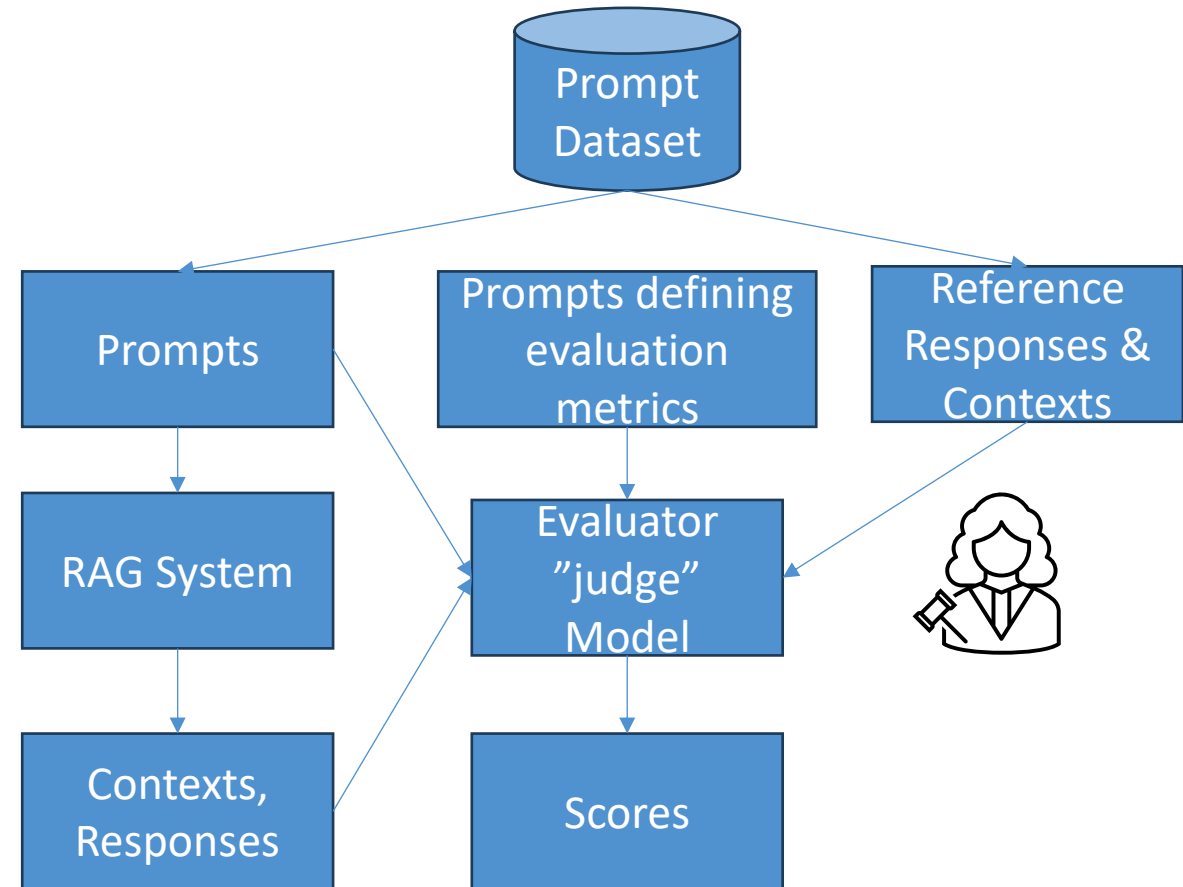
Bedrock Model Evaluations

- Automatic model evaluations
 - Tons of built-in task types, datasets, and metrics (seen here)
 - Text generation, summarization, Q&A, classification
- Human-based
 - Humans compare two models
 - Given your prompt dataset
 - Set up a work team in Cognito

Available built-in datasets for automatic model evaluation jobs in Amazon Bedrock			
Task type	Metric	Built-in datasets	Computed metric
General text generation	Accuracy	TREX	Real world knowledge (RWK) score
	Robustness	BOLD	Word error rate
		TREX	
		WikiText2	
	Toxicity	RealToxicityPrompts	Toxicity
Text summarization	Accuracy	Gigaword	BERTScore
	Toxicity	Gigaword	Toxicity
	Robustness	Gigaword	BERTScore and deltaBERTScore
Question and answer	Accuracy	BoolQ	NLP-F1
		NaturalQuestions	
		TriviaQA	
	Robustness	BoolQ	F1 and deltaF1
		NaturalQuestions	
		TriviaQA	
	Toxicity	BoolQ	Toxicity
		NaturalQuestions	
		TriviaQA	
Text classification	Accuracy	Women's Ecommerce Clothing Reviews	Accuracy (Binary accuracy from classification_accuracy_score)
	Robustness	Women's Ecommerce Clothing Reviews	classification_accuracy_score and delta_classification_accuracy_score

Bedrock Model Evaluations

- LLM as a judge
 - Evaluator model and generator model
 - You provide a prompt dataset
 - Many metrics available – we talked about this before
- RAG evaluation jobs
 - Retrieve only – measures retrieval from your RAG source
 - Relevance and coverage
 - Retrieve and generate – Measures the response as well
 - Correctness, completeness, helpfulness, logical coherence, etc.
 - Provide a prompt dataset in S3 in JSON line format
 - You can add your own metrics (based on prompts for the judge model)



Considerations for Deployment Validation

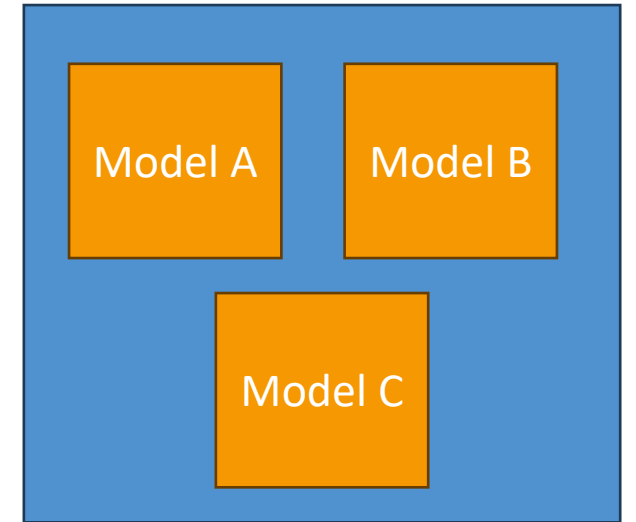
- Synthetic user workflows
 - Unit testing often doesn't cut it, because GenAI is non-deterministic
 - Simulate end to end usage of your application
 - Before and after deployments
 - Some useful tools
 - Cloudwatch synthetic monitoring (canaries,) Step Functions, EventBridge, Lambda
 - Store results in S3 / Athena, analyze in QuickSight
- AI-Specific output validation
 - Measure hallucination rate, semantic drift, faithfulness, compliance, etc.
 - Bedrock Model Evaluations does this
 - You could roll your own with Lambda + Step Functions
- Checking response consistency
 - Best you can do is make sure style is consistent, answers are similar for a test prompt dataset
 - Test that variability is within an accepted range



ML Implementation and Operations

Optimizing FM Deployments

- SageMaker AI offers single and multi-model endpoints
 - More generally, multi-container endpoints
 - Each endpoint supports deployment guardrails, VPC, network isolation
- You can train/tune a model in SageMaker AI, and deploy through Bedrock
 - You use Bedrock Custom Model Import for this
 - Now your inference is serverless 😊
- SageMaker AI Inference Components
 - Each model gets its own scaling policy



SageMaker multi-model endpoint

Optimizing FM Deployments

- Cross-region inference profiles
 - For endpoints on Bedrock
- EC2 Auto Scaling Groups
 - Load balancing in front of SageMaker AI endpoints
- Available model servers with SageMaker AI
 - TorchServe
 - DJL Serving (Deep Java Library)
 - Deep Learning Containers
 - DJL was created by Amazon, so more likely to show up on exam
 - Triton Inference Server

Optimizing FM Deployments

- Use asynchronous inference if latency isn't important
 - SageMaker AI async endpoints
 - Your own queue with SNS / SQS
- Model compression
 - Quantization (of model weights)
 - Pruning
 - Knowledge distillation
 - A smaller model is trained from a larger model
- Avoid premature optimization
 - Measure your performance, costs, resource utilization
 - Don't solve problems that don't exist!



Deployment Safeguards

- Deployment Guardrails
 - For asynchronous or real-time inference endpoints
 - Controls shifting traffic to new models
 - “Blue/Green Deployments”
 - All at once: shift everything, monitor, terminate blue fleet
 - Canary: shift a small portion of traffic and monitor
 - Linear: Shift traffic in linearly spaced steps
 - Auto-rollbacks
- Shadow Tests
 - Compare performance of shadow variant to production
 - You monitor in SageMaker console and decide when to promote it



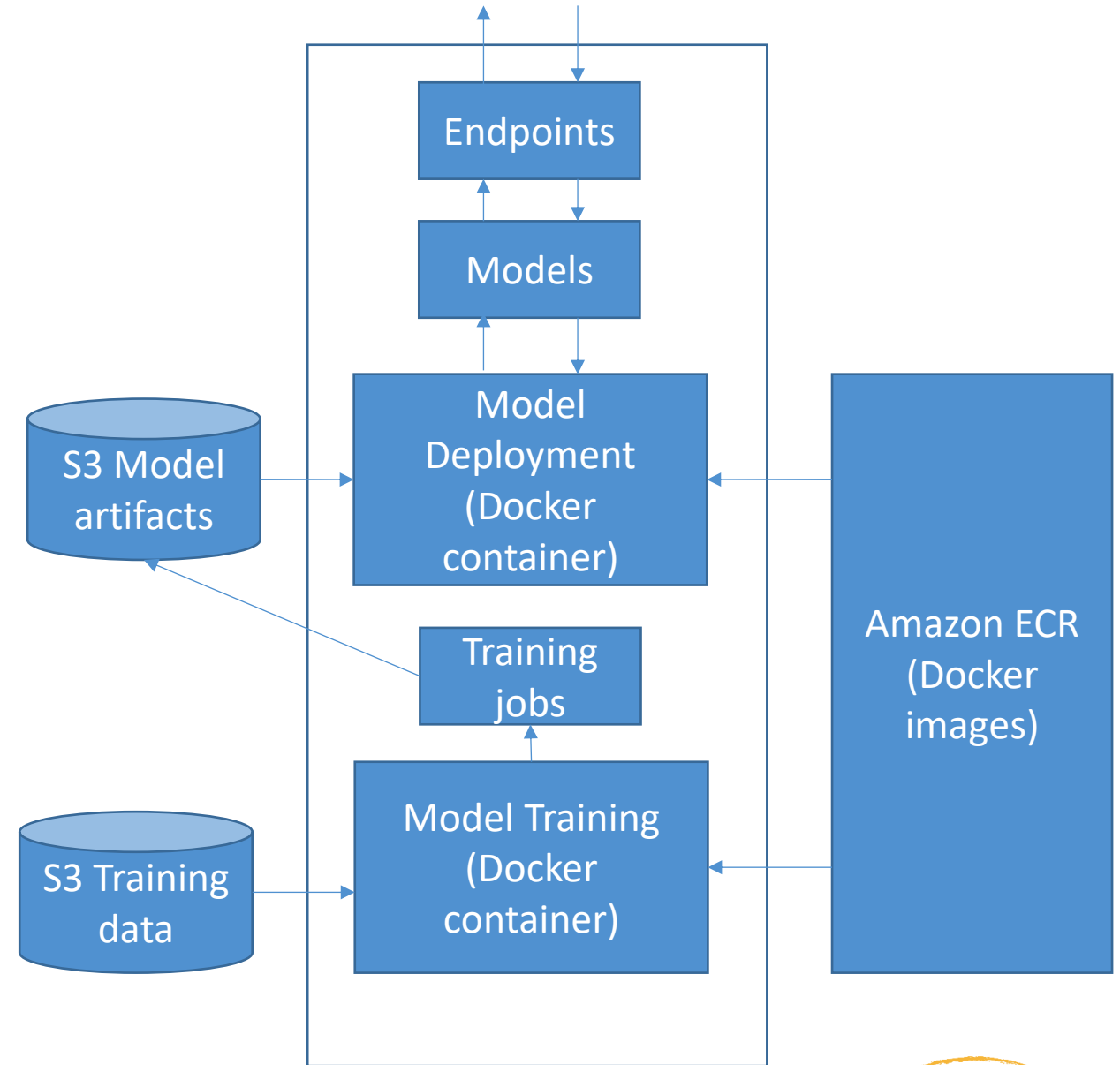
SageMaker AI and Docker Containers

SageMaker AI + Docker

- All models in SageMaker are hosted in Docker containers
 - Pre-built deep learning
 - Pre-built scikit-learn and Spark ML
 - Pre-built Tensorflow, MXNet, Chainer, PyTorch
 - Distributed training via Horovod or Parameter Servers
 - Your own training and inference code! Or extend a pre-built image.
- This allows you to use any script or algorithm within SageMaker, regardless of runtime or language
 - Containers are isolated, and contain all dependencies and resources needed to run

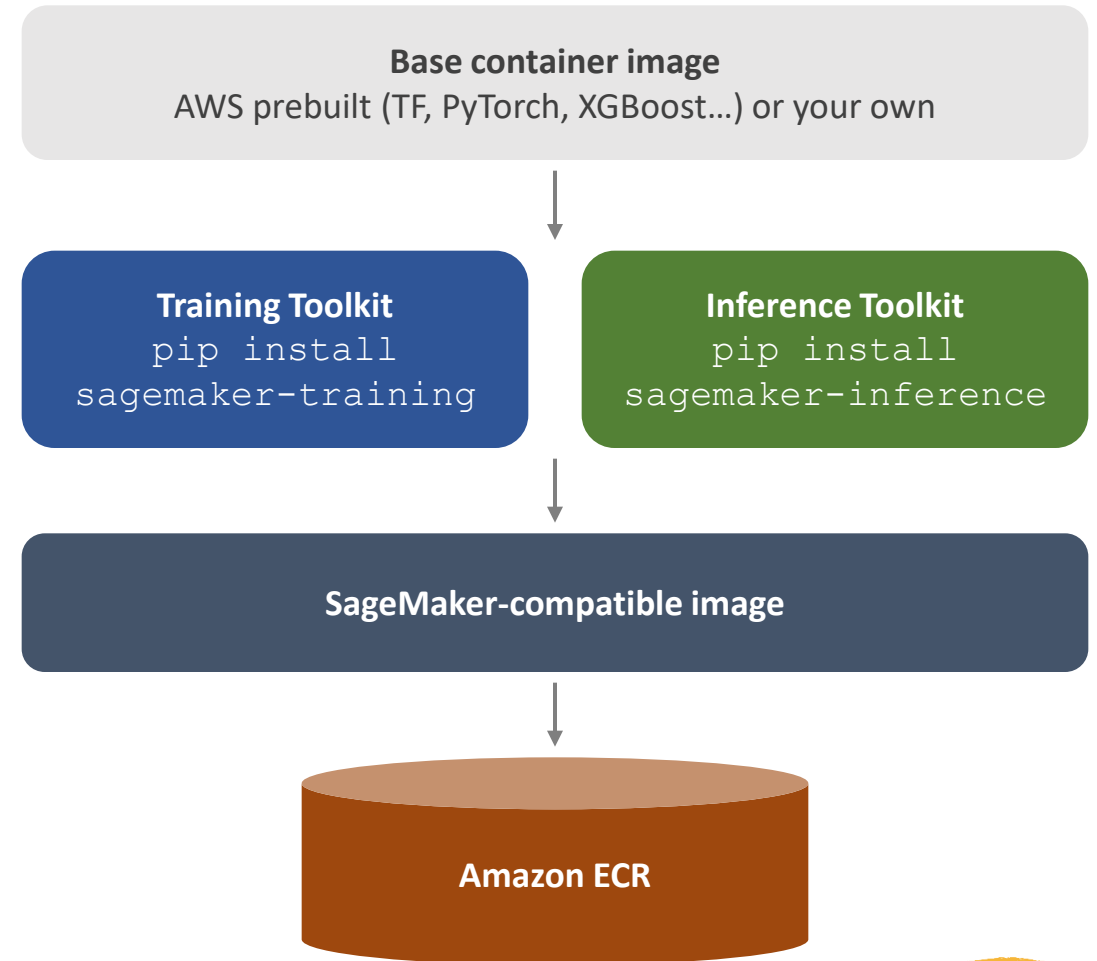
Using Docker

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



SageMaker AI Container Toolkits

- **SageMaker Training Toolkit** — adapt your image to run training on SageMaker
- **SageMaker AI Inference Toolkit** — adapt your image for model hosting / inference
- AWS provides prebuilt images for common frameworks (TensorFlow, PyTorch, XGBoost, Scikit-learn...)
- Often you can use or extend a prebuilt image instead of building one from scratch
- Container images live in Amazon ECR — SageMaker pulls them for training & hosting



Structure of a training container

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

Structure of a Deployment Container

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


Structure of your Docker image

- WORKDIR
 - nginx.conf
 - predictor.py
 - serve/
 - train/
 - wsgi.py

Assembling it all in a Dockerfile

```
FROM tensorflow/tensorflow:2.0.0a0

RUN pip3 install sagemaker-training

# Copies the training code inside the
container
COPY train.py /opt/ml/code/train.py

# Defines train.py as script entrypoint
ENV SAGEMAKER_PROGRAM train.py
```

Environment variables

- SAGEMAKER_PROGRAM
 - Run a script inside /opt/ml/code
- SAGEMAKER_TRAINING_MODULE
- SAGEMAKER_SERVICE_MODULE
- SM_MODEL_DIR
- SM_CHANNELS / SM_CHANNEL_*
- SM_HPS / SM_HP_*
- SM_USER_ARGS
- ...and many more

Using your own image

- `cd dockerfile`
- `!docker build -t foo .`
- `from sagemaker.estimator import Estimator`

```
estimator = Estimator(image_name='foo',  
                        role='SageMakerRole',  
                        train_instance_count=1,  
                        train_instance_type='local')
```

```
estimator.fit()
```

Production Variants

- You can test out multiple models on live traffic using Production Variants
 - Variant Weights tell SageMaker how to distribute traffic among them
 - So, you could roll out a new iteration of your model at say 10% variant weight
 - Once you're confident in its performance, ramp it up to 100%
- This lets you do A/B tests, and to validate performance in real-world settings
 - Offline validation isn't always useful
- Shadow Variants
- Deployment Guardrails

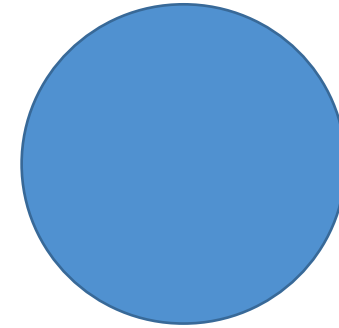
Managing SageMaker AI Resources

Choosing your instance types

- We covered this under “modeling”, even though it’s an operations concern
- In general, algorithms that rely on deep learning will benefit from GPU instances (P3, g4dn) for training
- Inference is usually less demanding and you can often get away with compute instances there (C5)
- GPU instances can be really pricey

Managed Spot Training

- Can use EC2 Spot instances for training
 - Save up to 90% over on-demand instances
- Spot instances can be interrupted!
 - Use checkpoints to S3 so training can resume
- Can increase training time as you need to wait for spot instances to become available



Automatic Scaling

- You set up a scaling policy to define target metrics, min/max capacity, cooldown periods
- Works with CloudWatch
- Dynamically adjusts number of instances for a production variant
- Load test your configuration before using it!



Token Efficiency

- Token Estimation and Tracking
 - Bedrock has a **CountTokens** API
 - Returns token count for a given request without actually running it
 - Different models behave differently
 - Costs nothing
 - Estimate costs prior to inference
 - Optimize prompts to fit within token limits
- CloudWatch tracks this too – also on the output side
 - InputTokenCount
 - outputTokenCount

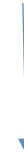


Other Stuff CloudWatch Monitors

- You can use these for additional optimization opportunities
- Count, time for requests
- TTFT Time to First Token (in streaming)
- Throttles / server errors / client errors on invocation
- Model latency, invocation count, throttles
- And again – input / output token counts



Amazon Bedrock



Amazon CloudWatch

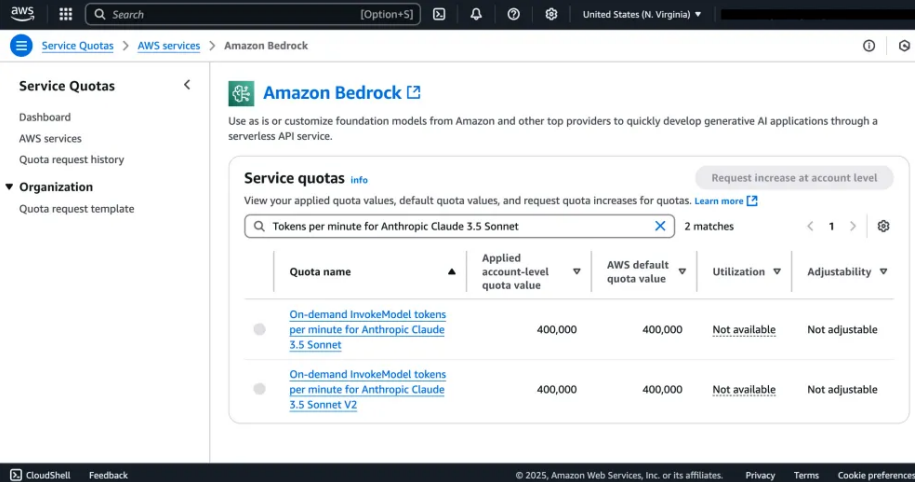
More Token Efficiency Techniques

- Context Window Optimization / Context Pruning
 - Chunking RAG data, limiting # of chunks retrieved
 - Filtering chunks by metadata to toss irrelevant stuff
 - Summarize (or even toss) older parts of conversation history
- Response Size Controls / Response Limiting
 - Use maxTokens
 - Bake desired length in the prompt (“respond in 50 words or less”)
 - Use few-shot examples to illustrate the desired verbosity
 - Use JSON output to force a given format / length
- Prompt Compression
 - Use a small model to summarize large chats, documents before sending to larger model
 - Use Knowledge Bases instead of complete documents in the prompt



Maximizing Resource Utilization

- Batching strategies
 - Batch embeddings up for vector stores
 - Bedrock Batch Inference
 - Submit many prompts together in S3, get responses in S3
- Capacity planning with Bedrock
 - Work from desired tokens per minute (TPM), requests per minute (RPM)
 - **AWS Service Quotas** has a Bedrock tool (seen here)
 - Request a quota increase if necessary, or provisioned throughput
 - **AWS CloudFormation** templates can help with capacity planning
- Tensor parallelism
 - Shards LLM weights across GPUs
 - Better memory utilization per container



Quota name	Applied account-level quota value	AWS default quota value	Utilization	Adjustability
On-demand InvokeModel tokens per minute for Anthropic Claude 3.5 Sonnet	400,000	400,000	Not available	Not adjustable
On-demand InvokeModel tokens per minute for Anthropic Claude 3.5 Sonnet V2	400,000	400,000	Not available	Not adjustable

Maximizing Resource Utilization

- Provisioned throughput
 - Bedrock allows you to provision by **Tokens** or by **Model Units** (MU's) which corresponds to token throughput
 - You must provision for customized models
 - Useful for consistent performance with high volume
 - You provision for a given model, and must **provide the provisioned model ARN as the modelID** when running inference on it
- Utilization monitoring
 - CloudWatch is your friend, with dashboards
 - AWS Cost Explorer to attribute model costs to business functions
- Auto-scaling
 - Serverless solutions like Lambda, Bedrock, OpenSearch serverless, AgentCore

Purchase Provisioned Throughput [Info](#)

Configuration options

Provisioned Throughput name

The name can have up to 40 characters. Valid characters A-Z,a-z,0-9, and - (hyphen).

Select model

[Select model](#)

► **Tags - optional**

Commitment term & model units [Info](#)

To purchase provisioned throughput, select a commitment term and choose a number of model units.

ⓘ Your Provisioned Throughput will automatically renew at the end of the commitment term. If you cancel your Provisioned Throughput before the end of the commitment term, you will be charged for the full term. You can cancel at any time.

Commitment term

Select a duration for which to keep the provisioned throughput.

Model units

A model unit delivers a specific throughput level for the specified model. Model unit quotas depend on the level of commitment you specify for the provisioned throughput. To request an increase, use the [limit increase form](#).

ⓘ **Edits to model, model units and commitment term will be restricted**

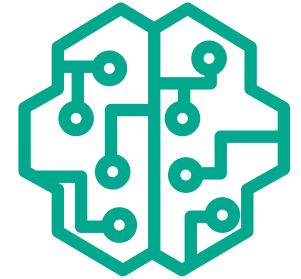
Once provisioned throughput is purchased

1. You can't edit the model units or commitment term.
2. If you select a base model, you can't change it later.

[Learn more](#)

Deploying Models for Inference

- SageMaker JumpStart
 - Deploying pre-trained models to pre-configured endpoints
- ModelBuilder from the SageMaker Python SDK
 - Configure deployment settings from code
- AWS CloudFormation
 - For advanced users who need consistent and repeatable deployments
 - `AWS::SageMaker::Model` resources creates a model to host at an endpoint



Deploying Models for Inference

- Real-time Inference
 - For interactive workloads with low latency requirements
- Amazon SageMaker Serverless Inference
 - No management of infrastructure
 - Ideal if workload has idle periods and uneven traffic over time, and can tolerate cold starts
- Asynchronous Inference
 - Queues requests and processes them asynchronously
 - Example: large payload sizes (up to 1GB) with long processing times, but near-real-time latency requirements
- Autoscaling
 - Dynamically adjust compute resources for endpoints based on traffic
- SageMaker Neo
 - Optimizes models for AWS Inferentia chips (for example)



Serverless Inference

- Introduced for 2022
- Specify your container, memory requirement, concurrency requirements
- Underlying capacity is automatically provisioned and scaled
- Good for infrequent or unpredictable traffic; will scale down to zero when there are no requests.
- Charged based on usage
- Monitor via CloudWatch
 - ModelSetupTime, Invocations, MemoryUtilization



Amazon SageMaker Serverless Inference

Fully managed serverless endpoints for machine learning inference with pay-per-use pricing

Amazon SageMaker Inference Recommender

- Recommends best instance type & configuration for your models
- Automates load testing model tuning
- Deploys to optimal inference endpoint
- How it works:
 - Register your model to the model registry
 - Benchmark different endpoint configurations
 - Collect & visualize metrics to decide on instance types
 - Existing models from zoos may have benchmarks already
- Instance Recommendations
 - Runs load tests on recommended instance types
 - Takes about 45 minutes
- Endpoint Recommendations
 - Custom load test
 - You specify instances, traffic patterns, latency requirements, throughput requirements
 - Takes about 2 hours

```
'InferenceRecommendations': [{
  'Metrics': {
    'CostPerHour': 0.20399999618530273,
    'CostPerInference': 5.246913588052848e-06,
    'MaximumInvocations': 648,
    'ModelLatency': 263596
  },
  'EndpointConfiguration': {
    'EndpointName': 'endpoint-name',
    'VariantName': 'variant-name',
    'InstanceType': 'ml.c5.xlarge',
    'InitialInstanceCount': 1
  },
  'ModelConfiguration': {
    'Compiled': False,
    'EnvironmentParameters': []
  }
},
```

SageMaker AI and Availability Zones

- SageMaker automatically attempts to distribute instances across availability zones
- But you need more than one instance for this to work!
- Deploy multiple instances for each production endpoint
- Configure VPC's with at least two subnets, each in a different AZ

AI-Specific Cost Drivers



FM Inference

- Billed per input + output tokens; model choice sets the rate
- Provisioned vs. on-demand / batch — economics follow traffic

Optimize: right-size the model; match pricing mode to traffic



Token Usage

- Prompts, history, retrieved context, long responses all add tokens
- CountTokens estimates usage before inference (supported models)

Optimize: trim prompts and context; cap response length



Embeddings

- Every embedding call consumes model resources
- High-volume re-embedding and oversized vector dimensions raise cost

Optimize: embed once, batch jobs, right-size dimensions



Vector Stores

- Storage size, vector dimensions, and index / query capacity all bill
- More indexed content = more RAG infrastructure

Optimize: index only what retrieval actually needs



Agents multiply cost: one user request can trigger many model calls, retrievals, and tool interactions

Optimize cost together with quality and latency — cheapest is not useful if it fails the workload.

Inference Pipelines

Inference Pipelines

- Linear sequence of 2-15 containers
- Any combination of pre-trained built-in algorithms or your own algorithms in Docker containers
- Combine pre-processing, predictions, post-processing
- Spark ML and scikit-learn containers OK
 - Spark ML can be run with Glue or EMR
 - Serialized into MLeap format
- Can handle both real-time inference and batch transforms



Model Performance Baselines & Drift

- A baseline captures expected / reference behavior, so production can be compared against it
 - Data baseline: feature distributions, ranges, missing-value rates, categorical proportions, means & std devs
 - Performance baseline: accuracy, precision, recall, F1, AUC, RMSE, MAE – whatever fits the workload
- Data drift: the distribution of production inputs shifts away from the reference / training data
- Model quality drift: prediction quality degrades over time
 - Measuring it may require real-world labels or feedback that arrive later
- Data drift doesn't automatically mean accuracy dropped – but the model is now seeing data unlike what it learned from

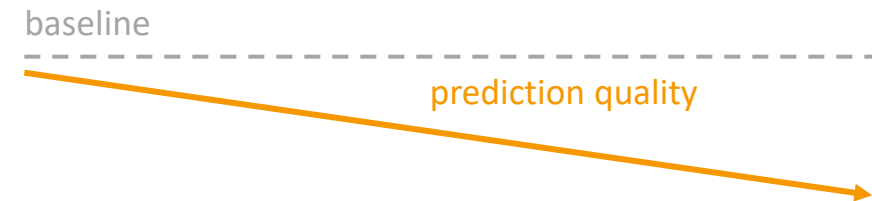
Data drift

The input distribution changed

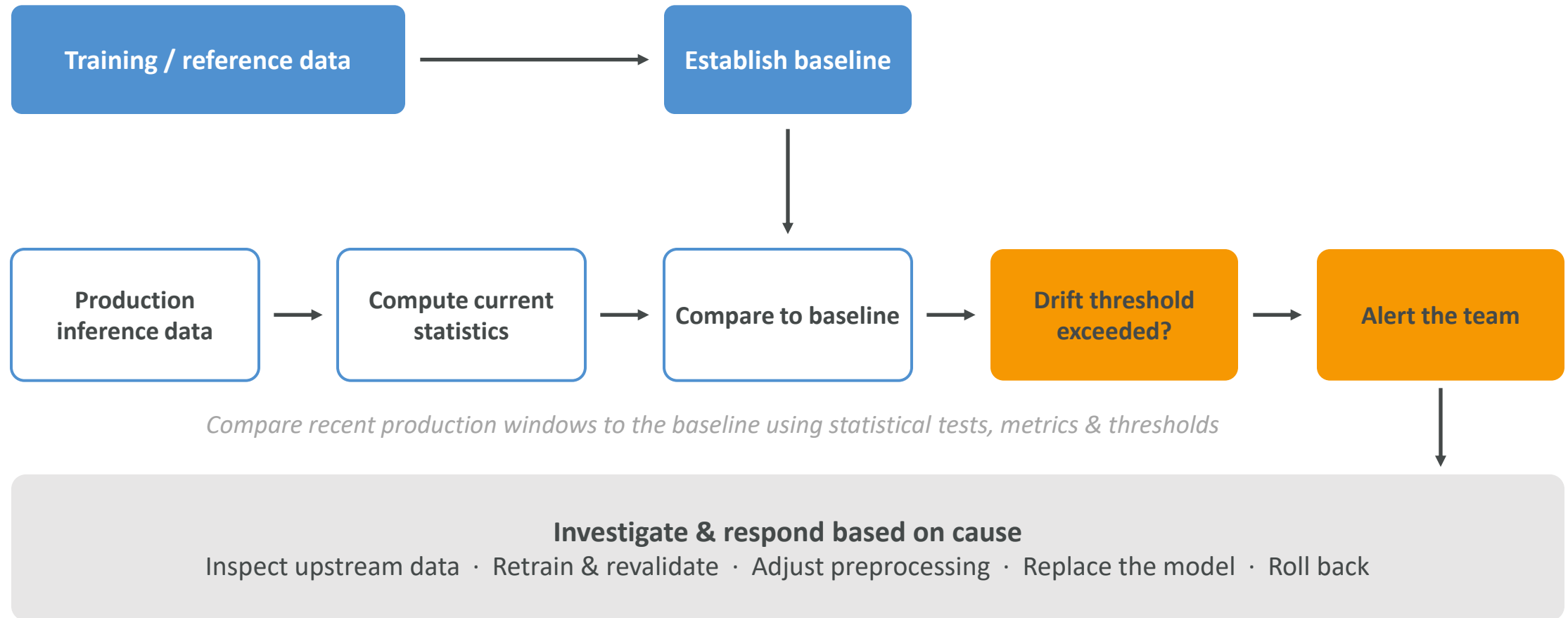


Model quality drift

The prediction quality changed



Detecting Drift: Production vs. Baseline



Drift Monitoring on AWS Today

- AWS publishes SageMaker AI monitoring reference solutions built on open-source tooling
 - MLflow Apps and Evidently AI generate drift & model-quality reports
- Amazon CloudWatch: publish custom drift / quality metrics; dashboards, alarms & anomaly detection
- Amazon QuickSight: monitoring & governance dashboards
- MLflow retains metrics + experiment / model context, making comparisons & root-cause analysis easier
- Exam tip: know the concepts & choose sensible tools – you won't be tested on Evidently APIs or implementation details

Heads up!

SageMaker Model Monitor is now in **maintenance mode** – no new features are planned.

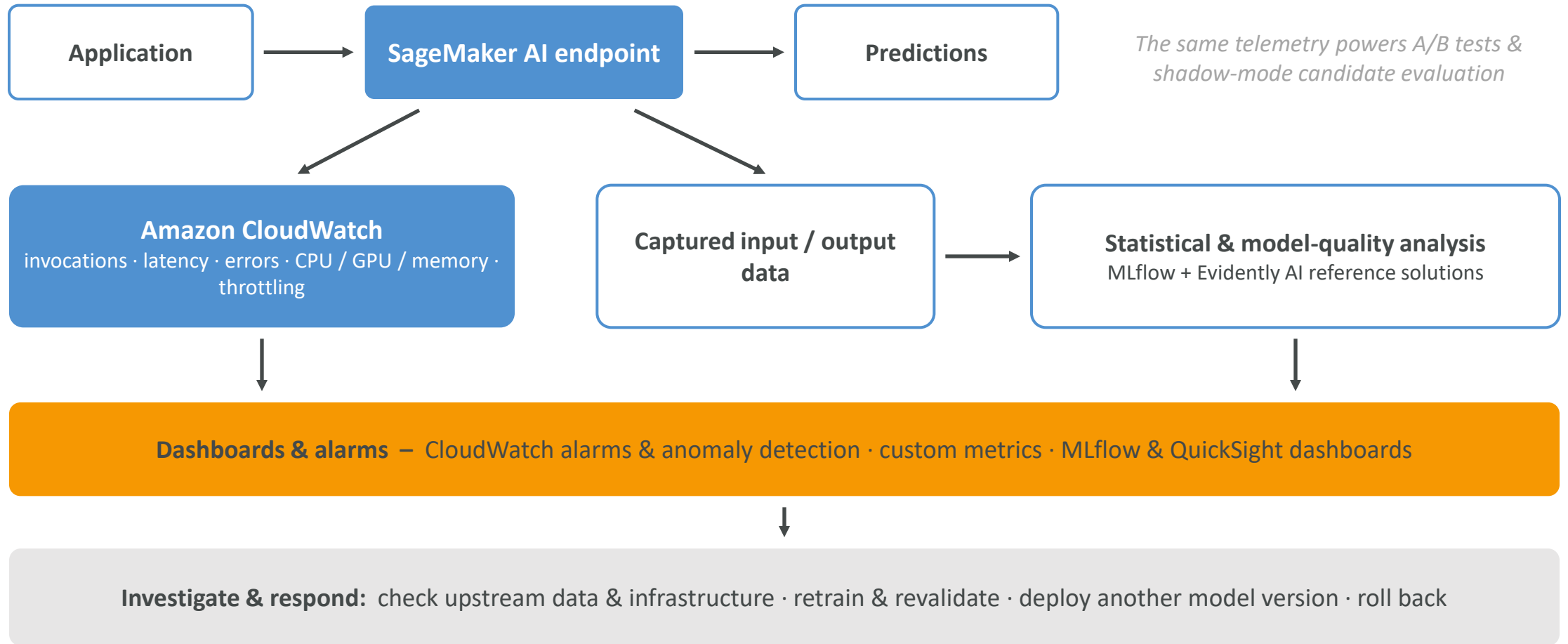
AWS publishes replacement monitoring approaches – don't build new monitoring on Model Monitor.

What Should We Monitor?

1 · Infrastructure	2 · Input Data	3 · Model Quality	4 · Response
• Endpoint invocations & traffic • Latency • Errors & failures • CPU / GPU / memory utilization • Capacity & throttling • Resource saturation	• Missing or malformed features • Unexpected ranges or categories • Changes in input distributions • Data drift vs. a baseline	• Prediction / output distributions • Accuracy, precision, recall, F1, RMSE... when real labels arrive • Ground truth is often delayed in production	• Dashboards & alerts • Investigate upstream data & infrastructure • Retrain & revalidate • Deploy another model version • Roll back when necessary

Monitoring is workload-specific – no universal metric or drift threshold proves every model is healthy

A Production ML Monitoring Architecture

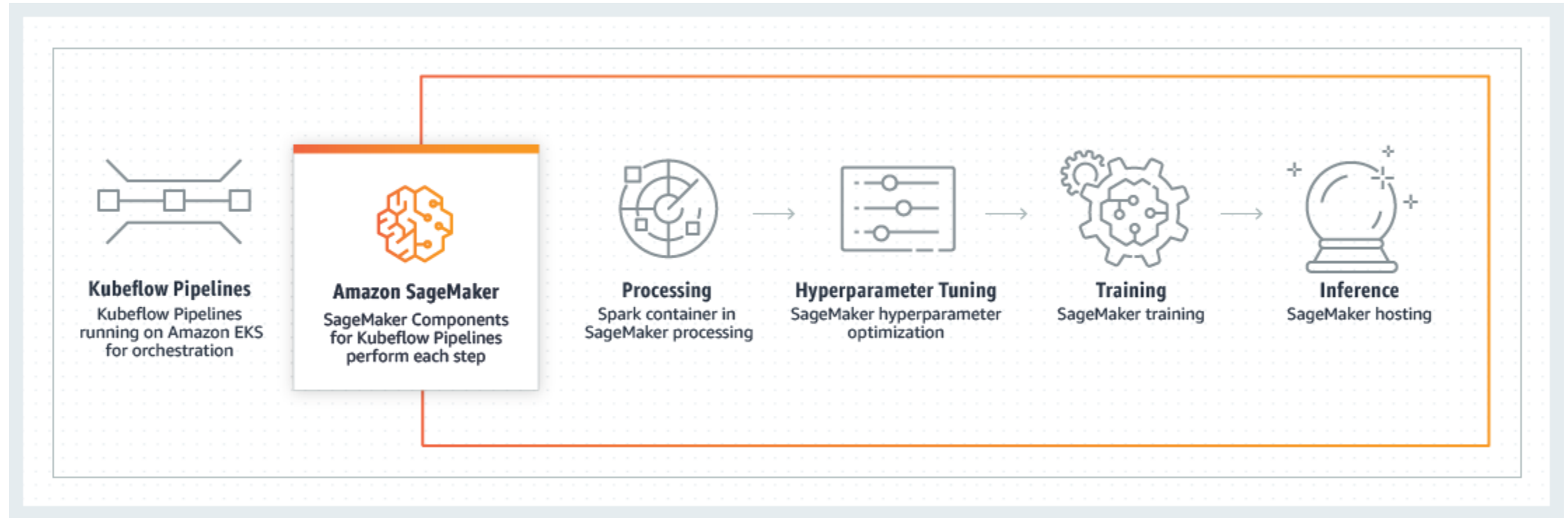


MLOps with SageMaker AI

MLOps with SageMaker AI and Kubernetes

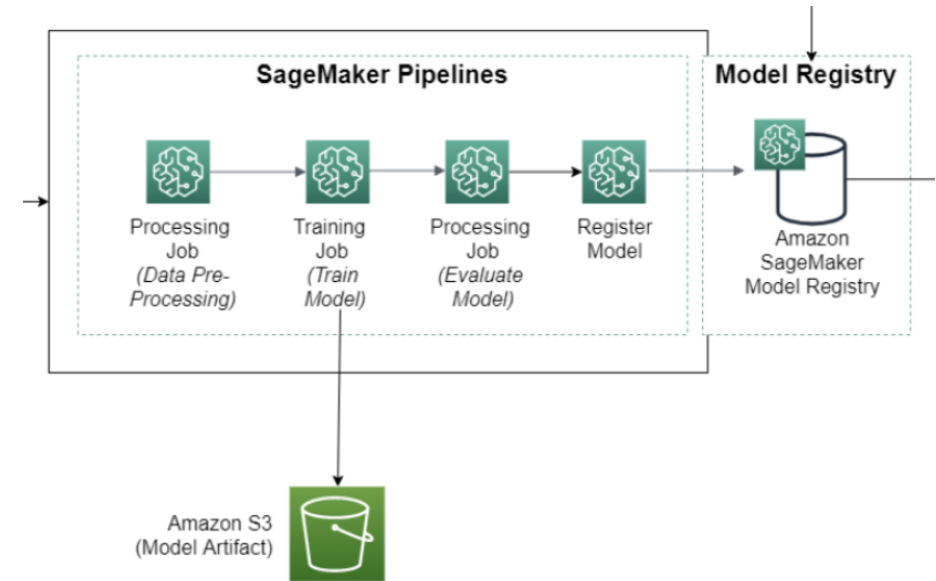
- Integrates SageMaker with Kubernetes-based ML infrastructure
- Components for Kubeflow Pipelines
- Enables hybrid ML workflows (on-prem + cloud)
- Enables integration of existing ML platforms built on Kubernetes / Kubeflow

SageMaker Components for Kubeflow Pipelines

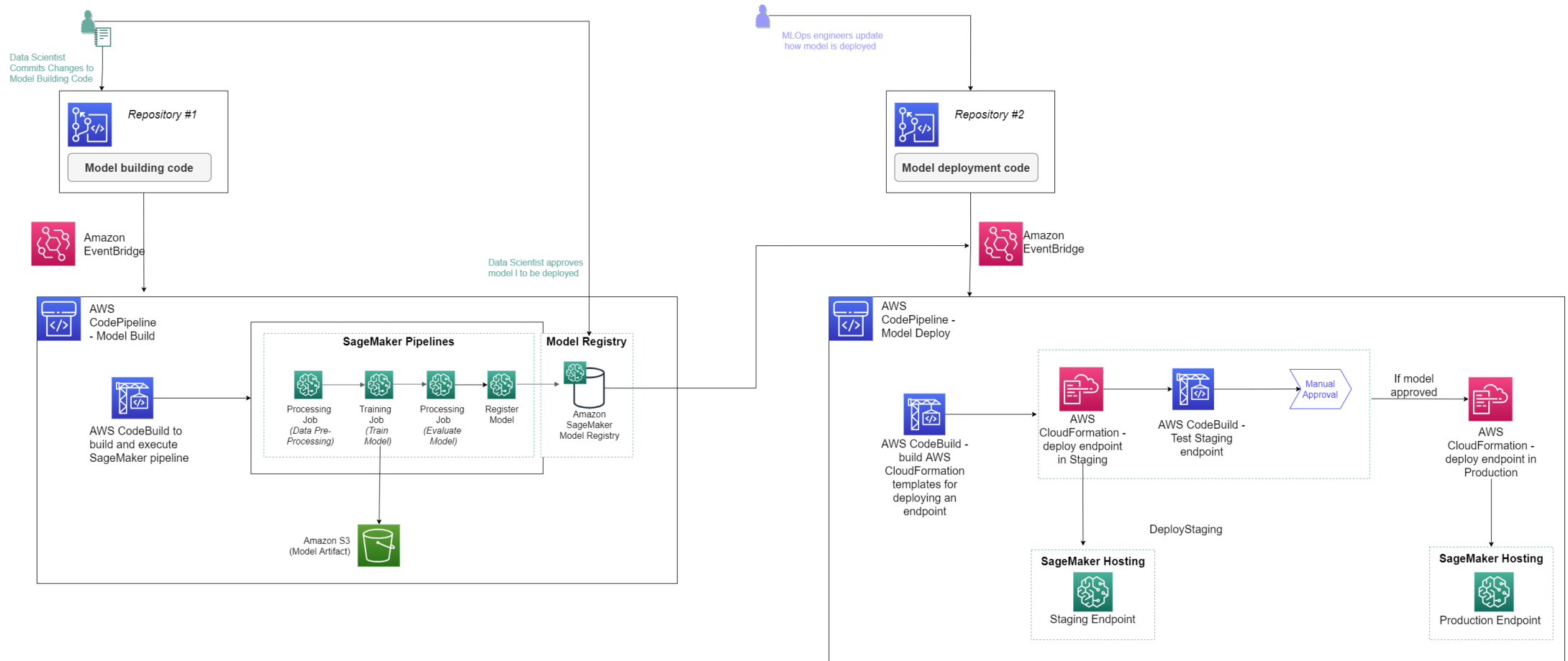


SageMaker Projects

- SageMaker Studio's native MLOps solution with CI/CD
 - Build images
 - Prep data, feature engineering
 - Train models
 - Evaluate models
 - Deploy models
 - Monitor & update models
- Uses code repositories for building & deploying ML solutions
- Uses SageMaker Pipelines defining steps

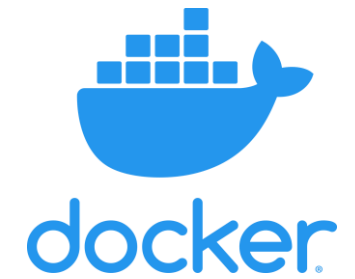


SageMaker Projects



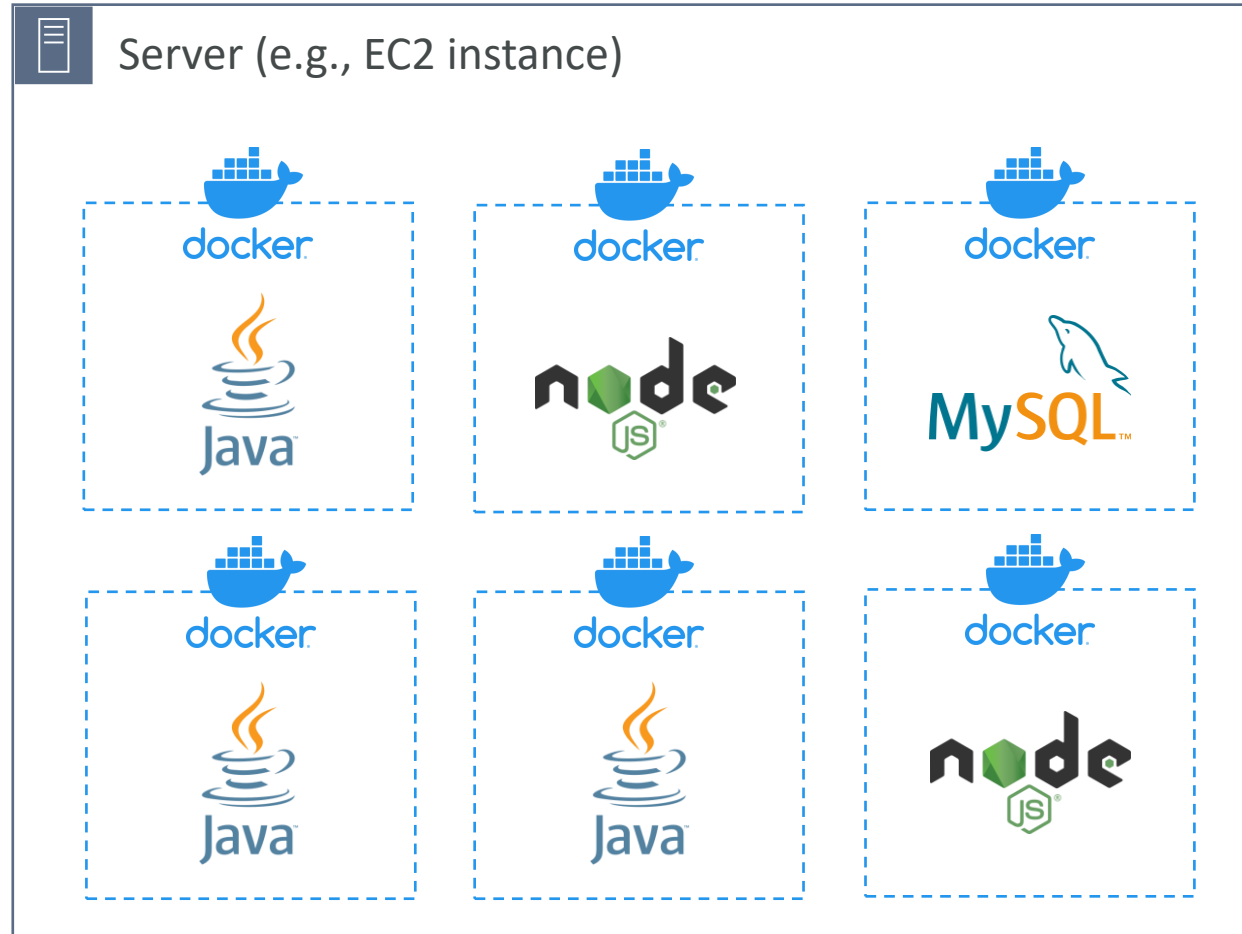
Containers on AWS

What is Docker?



- Docker is a software development platform to deploy apps
- Apps are packaged in **containers** that can be run on any OS
- Apps run the same, regardless of where they're run
 - Any machine
 - No compatibility issues
 - Predictable behavior
 - Less work
 - Easier to maintain and deploy
 - Works with any language, any OS, any technology
- Use cases: microservices architecture, lift-and-shift apps from on-premises to the AWS cloud, ...

Docker on an OS

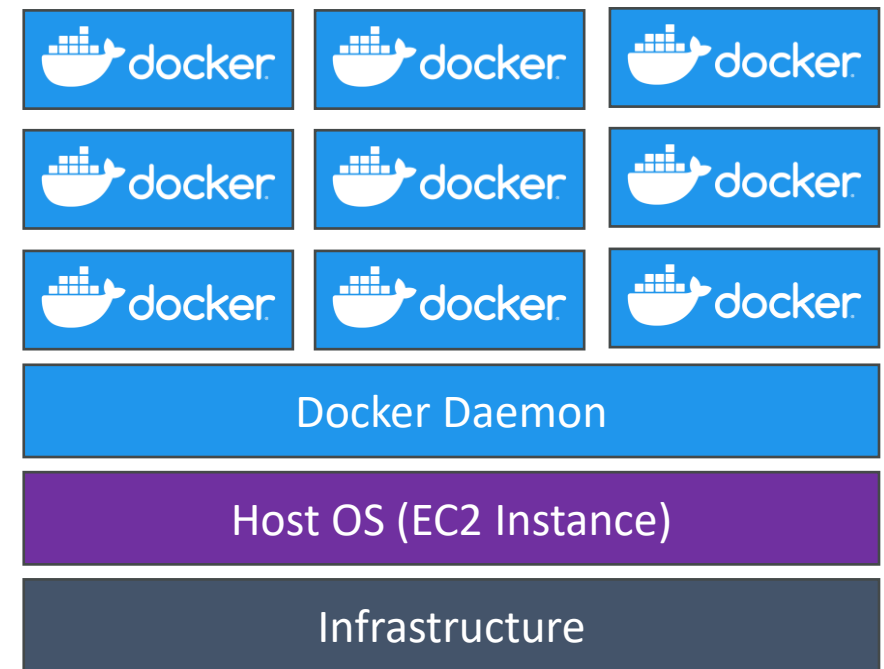
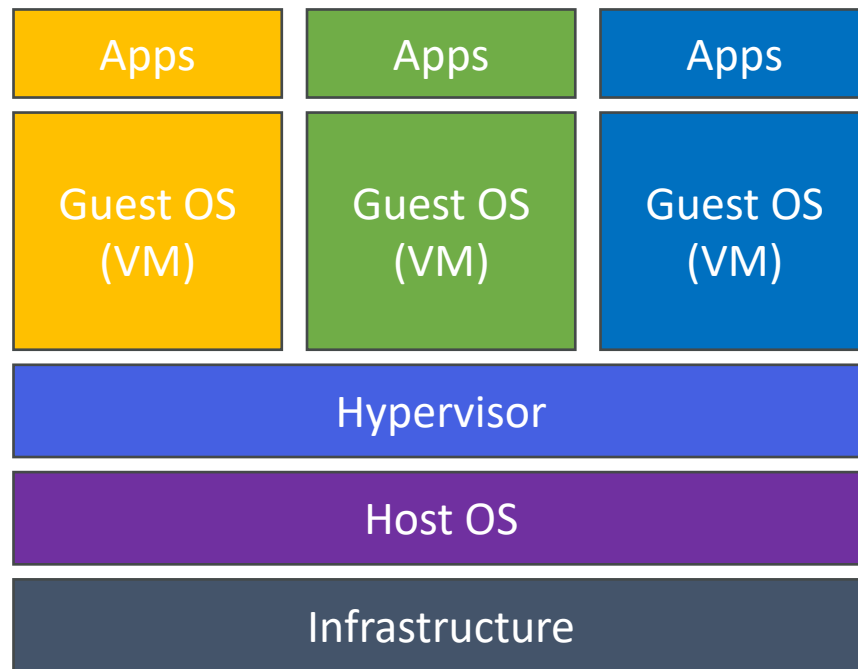


Where are Docker images stored?

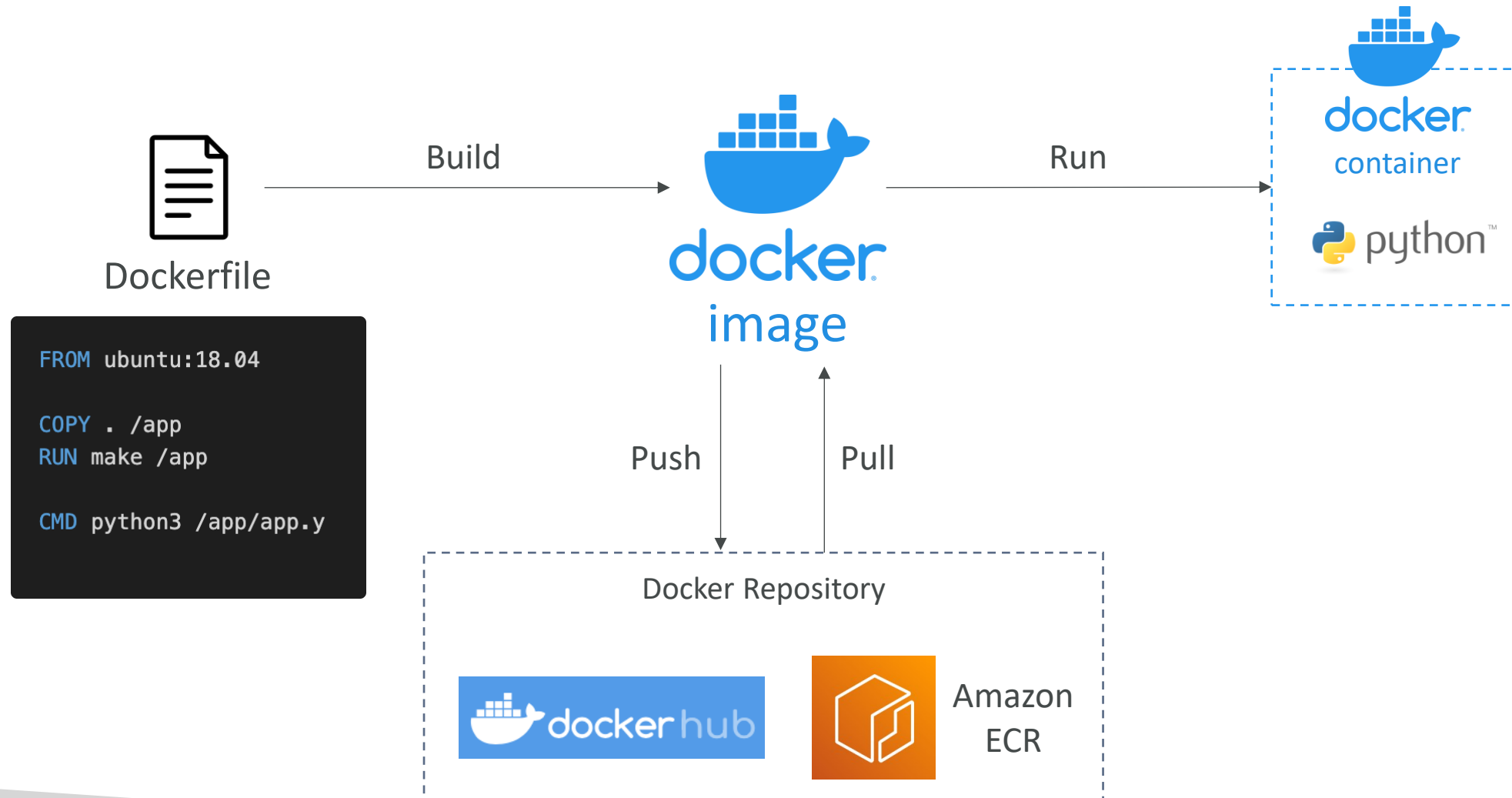
- Docker images are stored in Docker Repositories
- **Docker Hub** (<https://hub.docker.com>)
 - **Public** repository
 - Find base images for many technologies or OS (e.g., Ubuntu, MySQL, ...)
- **Amazon ECR (Amazon Elastic Container Registry)**
 - **Private** repository
 - **Public** repository (**Amazon ECR Public Gallery** <https://gallery.ecr.aws>)

Docker vs. Virtual Machines

- Docker is "sort of" a virtualization technology, but not exactly
- Resources are shared with the host => many containers on one server



Getting Started with Docker



Docker Containers Management on AWS

- **Amazon Elastic Container Service (Amazon ECS)**

- Amazon's own container platform



Amazon ECS

- **Amazon Elastic Kubernetes Service (Amazon EKS)**

- Amazon's managed Kubernetes (open source)



Amazon EKS

- **AWS Fargate**

- Amazon's own Serverless container platform
- Works with ECS and with EKS



AWS Fargate

- **Amazon ECR:**

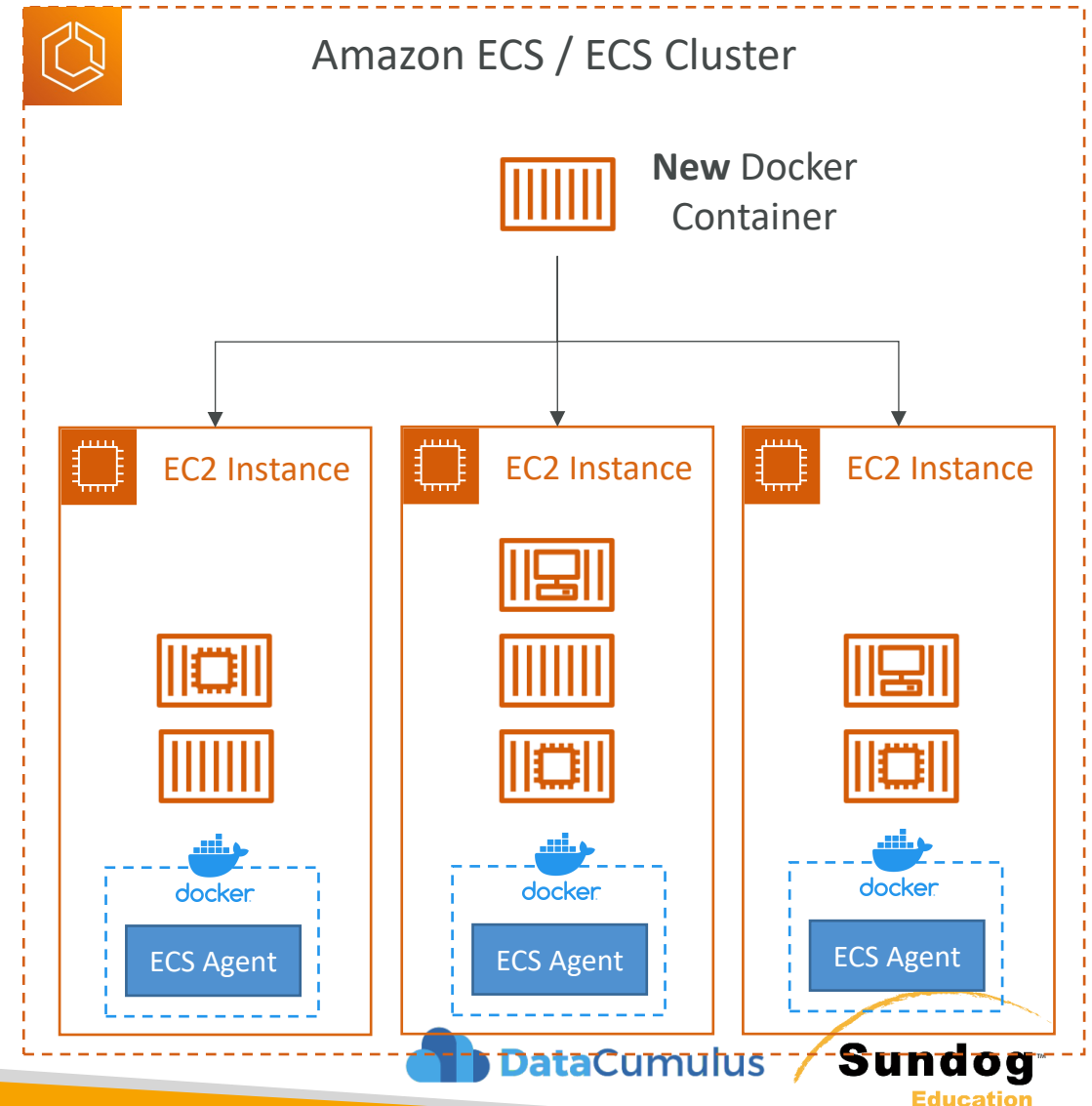
- Store container images



Amazon ECR

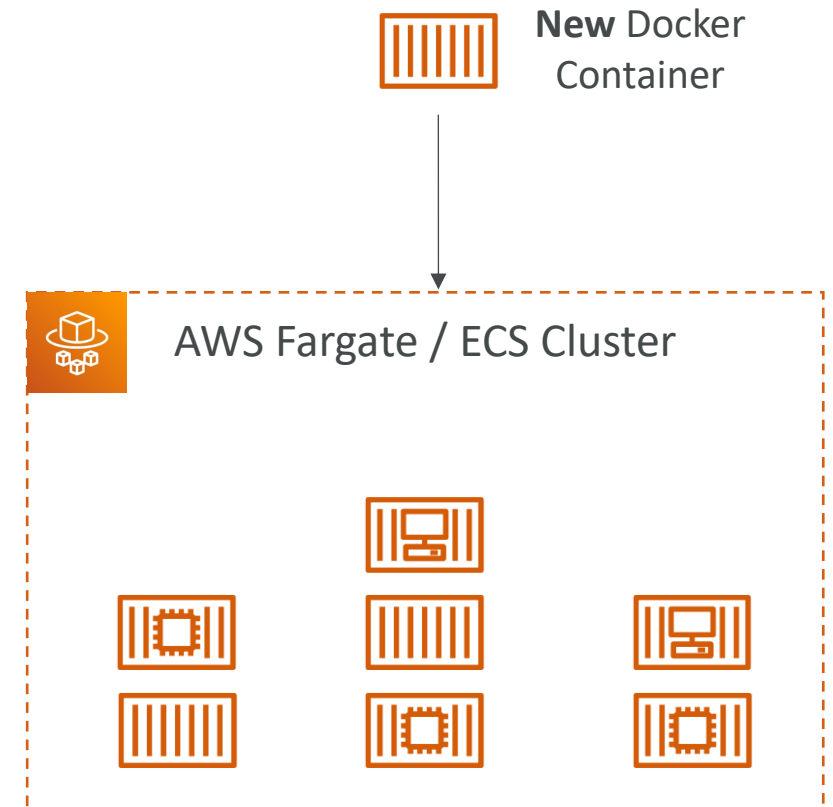
Amazon ECS - EC2 Launch Type

- ECS = Elastic Container Service
- Launch Docker containers on AWS = Launch **ECS Tasks** on ECS Clusters
- **EC2 Launch Type: you must provision & maintain the infrastructure (the EC2 instances)**
- Each EC2 Instance must run the ECS Agent to register in the ECS Cluster
- AWS takes care of starting / stopping containers



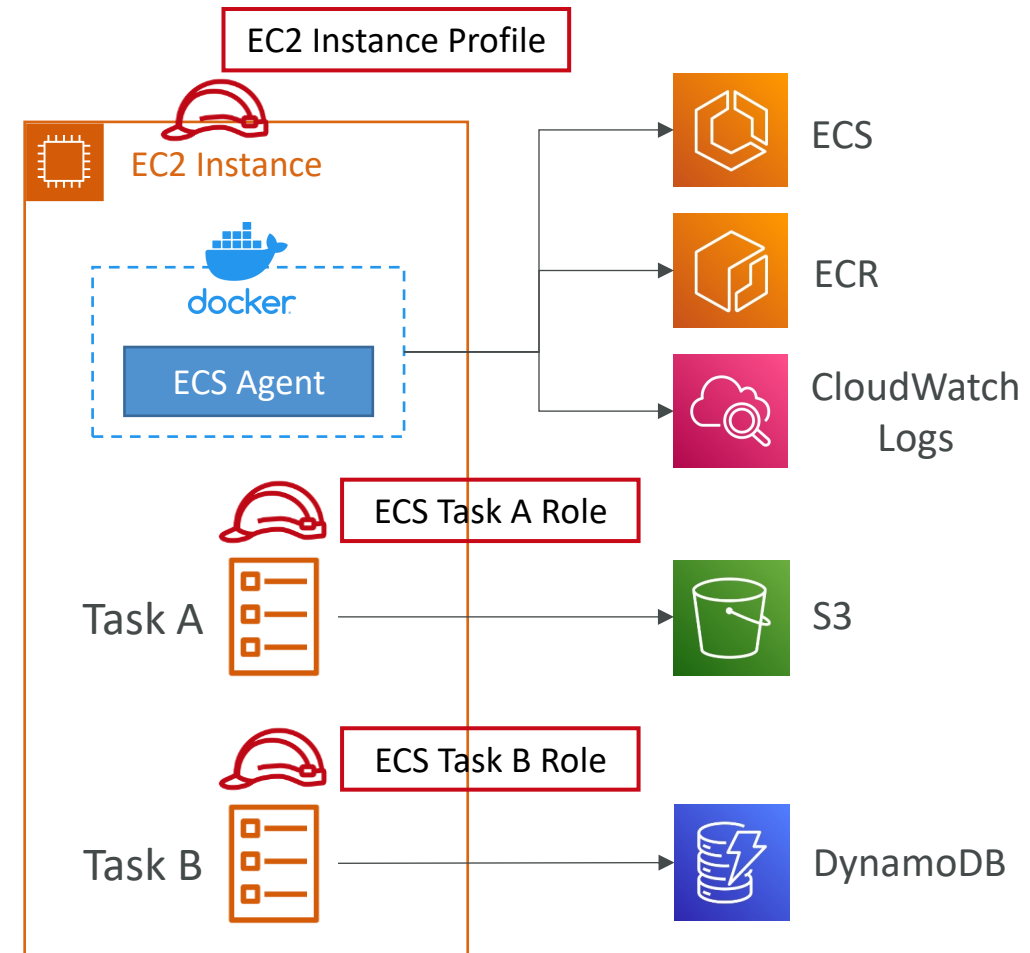
Amazon ECS – Fargate Launch Type

- Launch Docker containers on AWS
- **You do not provision the infrastructure (no EC2 instances to manage)**
- **It's all Serverless!**
- You just create task definitions
- AWS just runs ECS Tasks for you based on the CPU / RAM you need
- To scale, just increase the number of tasks. Simple - no more EC2 instances



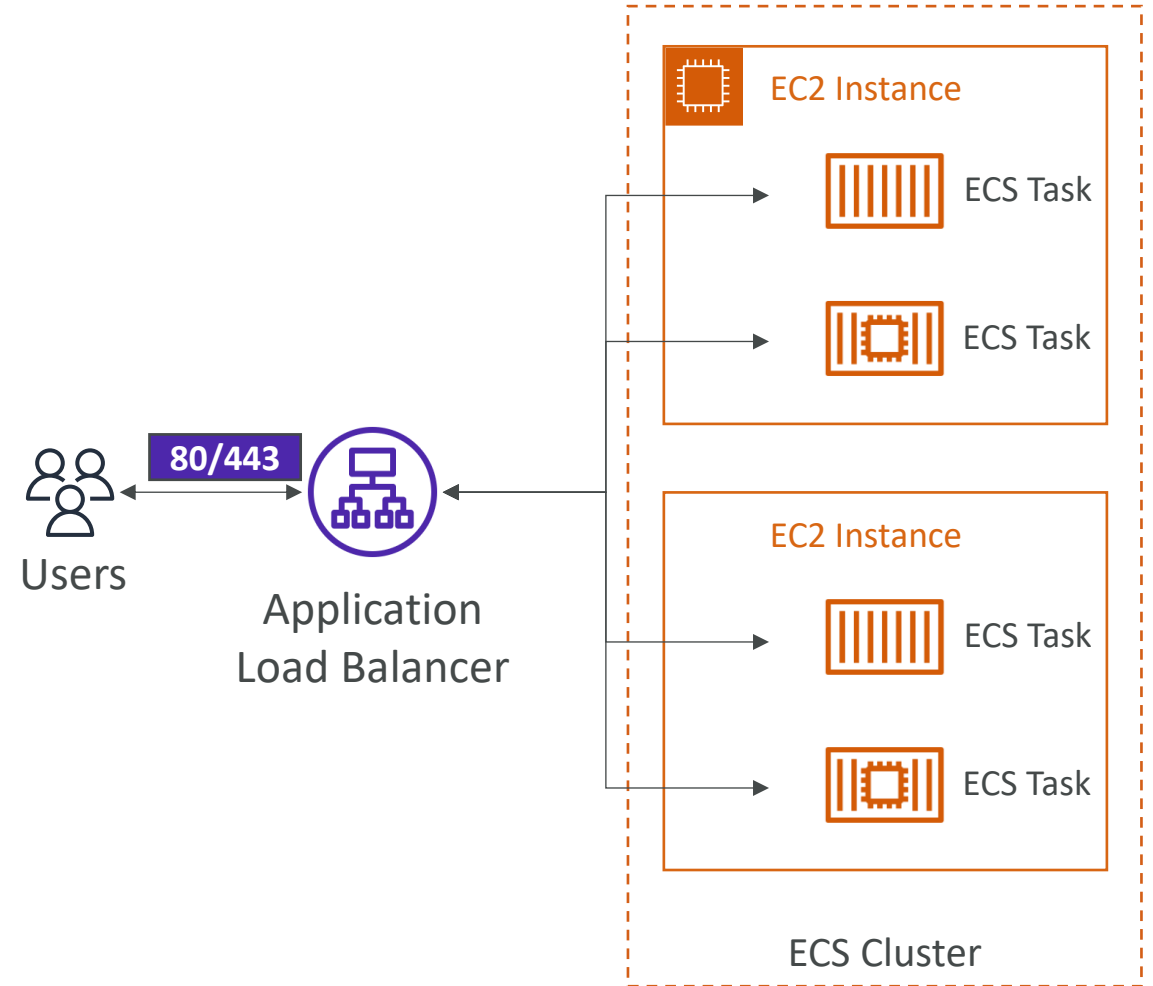
Amazon ECS – IAM Roles for ECS

- **EC2 Instance Profile (EC2 Launch Type only):**
 - Used by the ECS agent
 - Makes API calls to ECS service
 - Send container logs to CloudWatch Logs
 - Pull Docker image from ECR
 - Reference sensitive data in Secrets Manager or SSM Parameter Store
- **ECS Task Role:**
 - Allows each task to have a specific role
 - Use different roles for the different ECS Services you run
 - Task Role is defined in the task definition



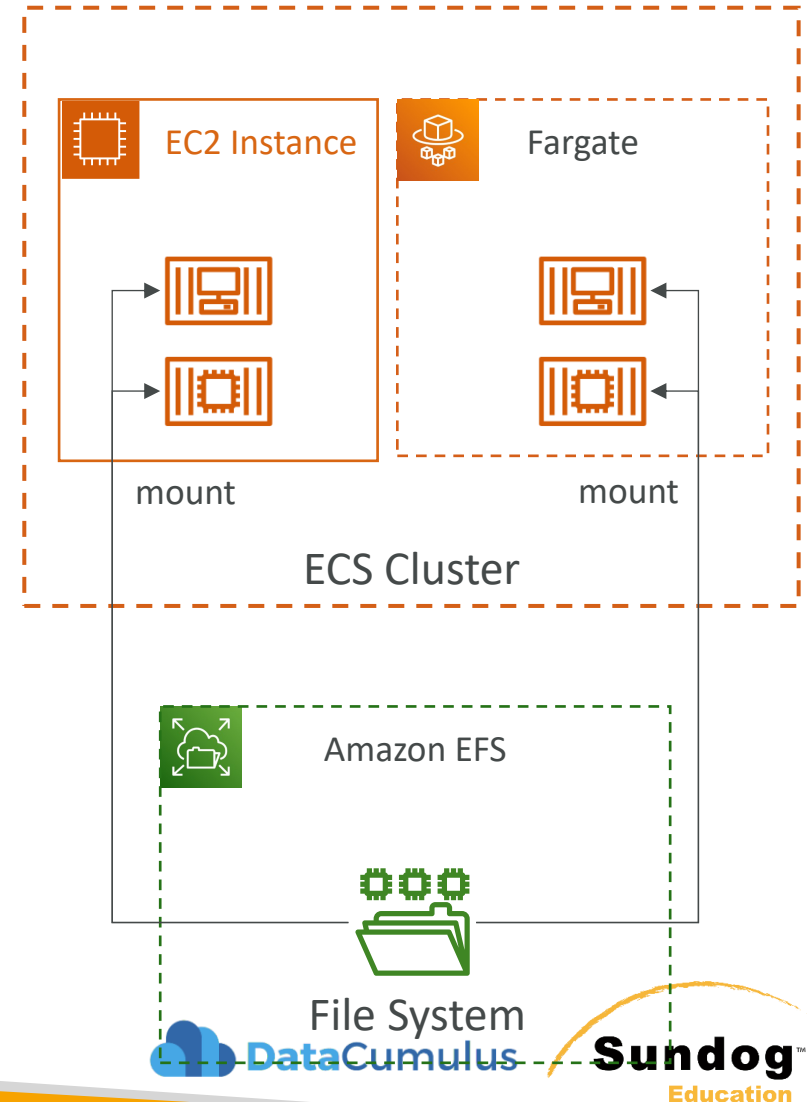
Amazon ECS – Load Balancer Integrations

- **Application Load Balancer**
supported and works for most use cases
- **Network Load Balancer**
recommended only for high throughput / high performance use cases, or to pair it with AWS Private Link
- **Classic Load Balancer** supported but not recommended (no advanced features – no Fargate)



Amazon ECS – Data Volumes (EFS)

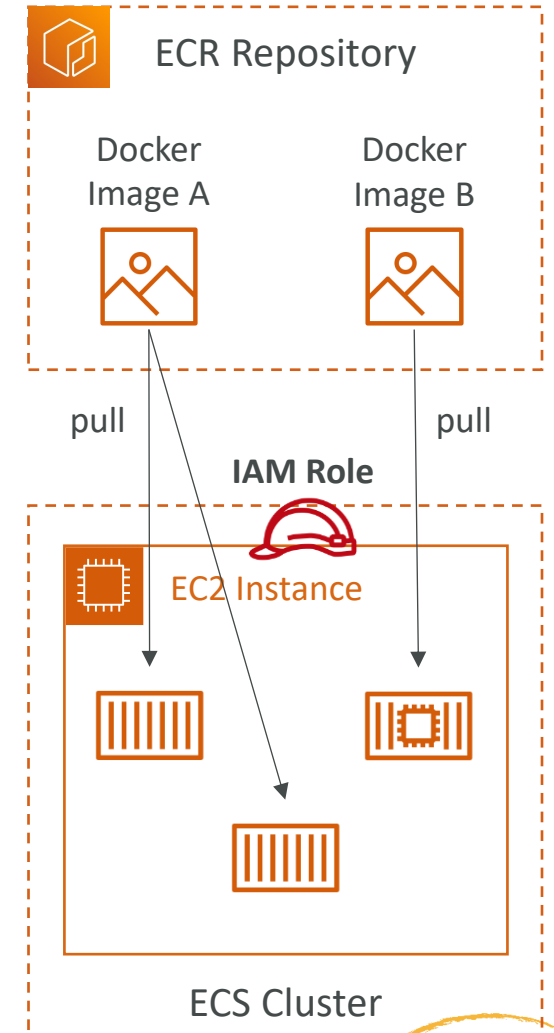
- Mount EFS file systems onto ECS tasks
- Works for both **EC2** and **Fargate** launch types
- Tasks running in any AZ will share the same data in the EFS file system
- **Fargate + EFS = Serverless**
- Use cases: persistent multi-AZ shared storage for your containers
- Note:
 - Amazon S3 cannot be mounted as a file system





Amazon ECR

- ECR = Elastic Container Registry
- Store and manage Docker images on AWS
- **Private** and **Public** repository (**Amazon ECR Public Gallery** <https://gallery.ecr.aws>)
- Fully integrated with ECS, backed by Amazon S3
- Access is controlled through IAM (permission errors => policy)
- Supports image vulnerability scanning, versioning, image tags, image lifecycle, ...

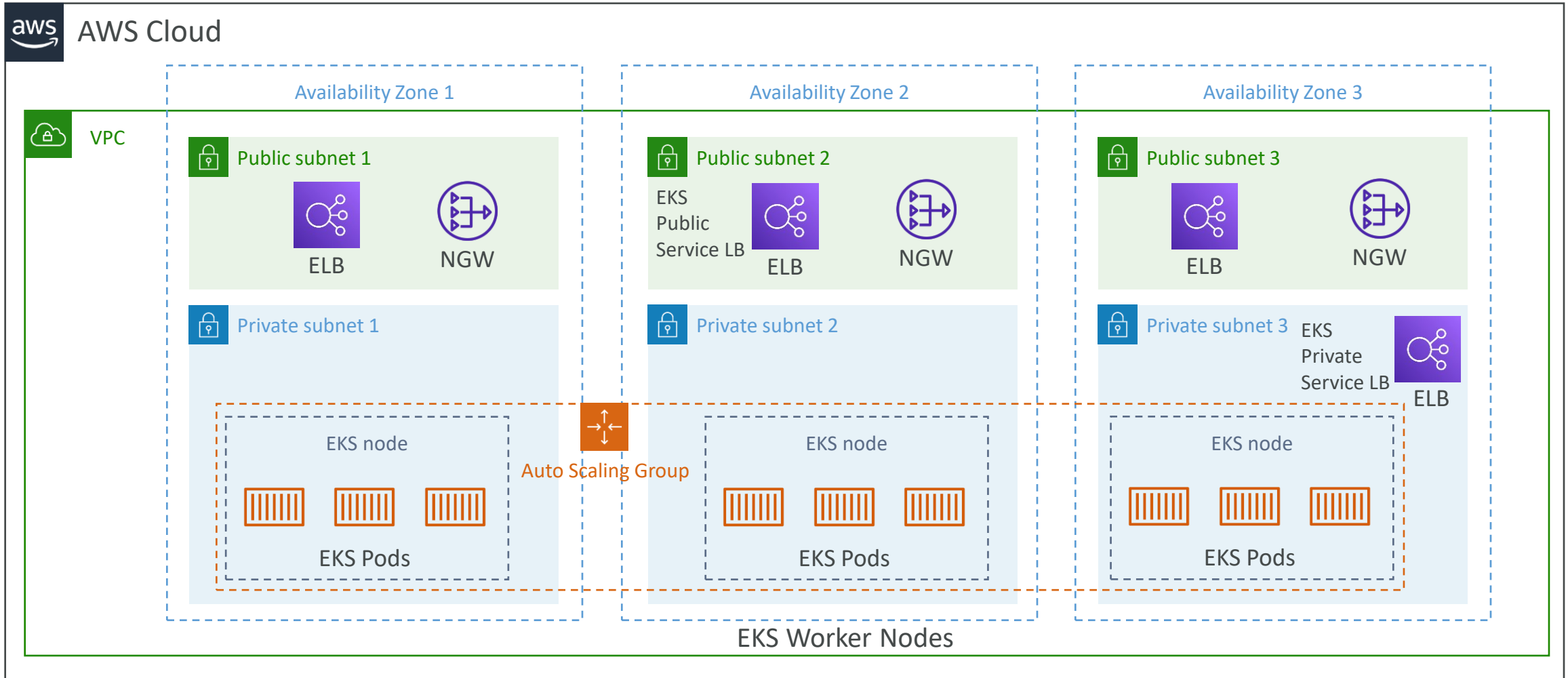


Amazon EKS Overview



- Amazon EKS = Amazon Elastic **Kubernetes** Service
- It is a way to launch **managed Kubernetes clusters on AWS**
- Kubernetes is an **open-source system** for automatic deployment, scaling and management of containerized (usually Docker) application
- It's an alternative to ECS, similar goal but different API
- EKS supports **EC2** if you want to deploy worker nodes or **Fargate** to deploy serverless containers
- **Use case:** if your company is already using Kubernetes on-premises or in another cloud, and wants to migrate to AWS using Kubernetes
- **Kubernetes is cloud-agnostic** (can be used in any cloud – Azure, GCP...)
- For multiple regions, deploy one EKS cluster per region
- Collect logs and metrics using **CloudWatch Container Insights**

Amazon EKS - Diagram



Amazon EKS – Node Types

- **Managed Node Groups**

- Creates and manages Nodes (EC2 instances) for you
- Nodes are part of an ASG managed by EKS
- Supports On-Demand or Spot Instances

- **Self-Managed Nodes**

- Nodes created by you and registered to the EKS cluster and managed by an ASG
- You can use prebuilt AMI - Amazon EKS Optimized AMI
- Supports On-Demand or Spot Instances

- **AWS Fargate**

- No maintenance required; no nodes managed

Amazon EKS – Data Volumes

- Need to specify **StorageClass** manifest on your EKS cluster
- Leverages a **Container Storage Interface (CSI)** compliant driver
- Support for...
- Amazon EBS
- Amazon EFS (works with Fargate)
- Amazon FSx for Lustre
- Amazon FSx for NetApp ONTAP



AWS Batch

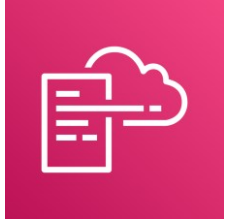


- Run batch jobs as Docker images
 - Dynamic provisioning of the instances (EC2 & Spot Instances)
 - Optimal quantity and type based on volume and requirements
 - No need to manage clusters, fully **serverless**
 - You just pay for the underlying EC2 instances
-
- Schedule Batch Jobs using CloudWatch Events
 - Orchestrate Batch Jobs using AWS

AWS Batch vs Glue

- Glue:
 - Glue ETL - Run Apache Spark code, Scala or Python based, focus on the ETL
 - Glue ETL - Do not worry about configuring or managing the resources
 - Data Catalog to make the data available to Athena or Redshift Spectrum
- Batch:
 - For any computing job regardless of the job (must provide Docker image)
 - Resources are created in your account, managed by Batch
 - For any non-ETL related work, Batch is probably better

What is CloudFormation



- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
 - I want a security group
 - I want two EC2 instances using this security group
 - I want an S3 bucket
 - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

Benefits of AWS CloudFormation (1/2)

- **Infrastructure as code**

- No resources are manually created, which is excellent for control
- Changes to the infrastructure are reviewed through code

- **Cost**

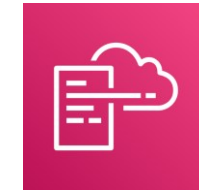
- Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
- You can estimate the costs of your resources using the CloudFormation template
- Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

Benefits of AWS CloudFormation (2/2)

- Productivity
 - Ability to destroy and re-create an infrastructure on the cloud on the fly
 - Automated generation of Diagram for your templates!
 - Declarative programming (no need to figure out ordering and orchestration)
- Don't re-invent the wheel
 - Leverage existing templates on the web!
 - Leverage the documentation
- **Supports (almost) all AWS resources:**
 - Everything we'll see in this course is supported
 - You can use "custom resources" for resources that are not supported

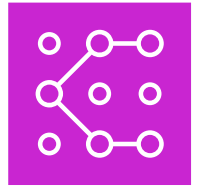
CloudFormation + Infrastructure Composer

- Example: WordPress CloudFormation Stack



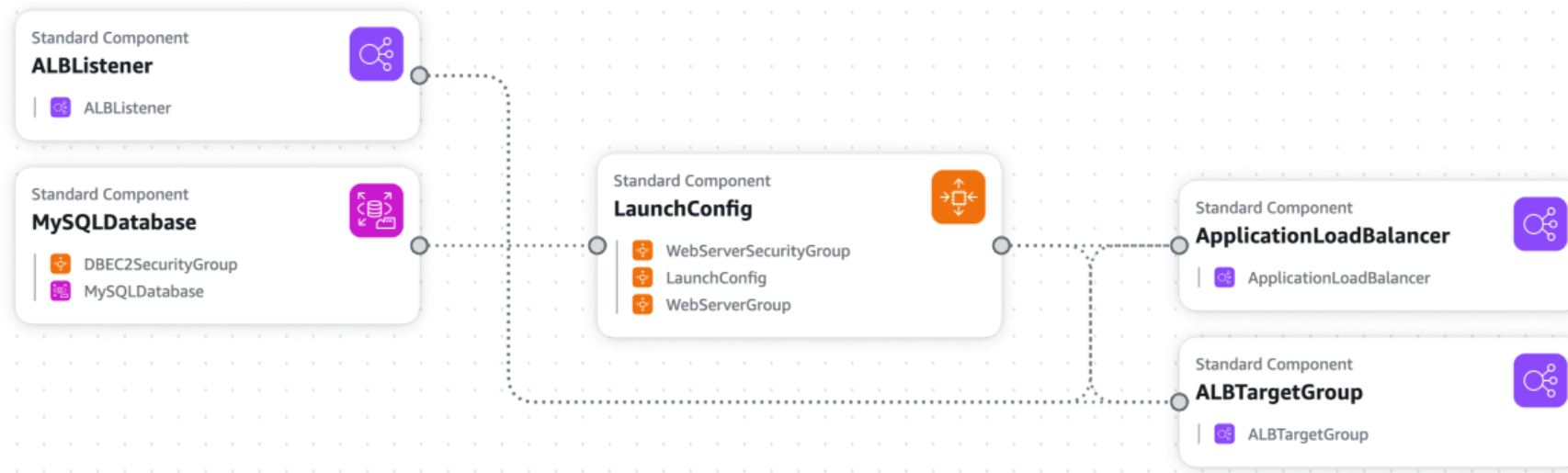
CloudFormation

+



Infrastructure
Composer

- We can see all the **resources**
- We can see the **relations** between the components



AWS Cloud Development Kit (CDK)



- Define your cloud infrastructure using a familiar language:
 - JavaScript/TypeScript, Python, Java, and .NET
- Contains high level components called **constructs**
- The code is “compiled” into a CloudFormation template (JSON/YAML)
- **You can therefore deploy infrastructure and application runtime code together**
 - Great for Lambda functions
 - Great for Docker containers in ECS / EKS

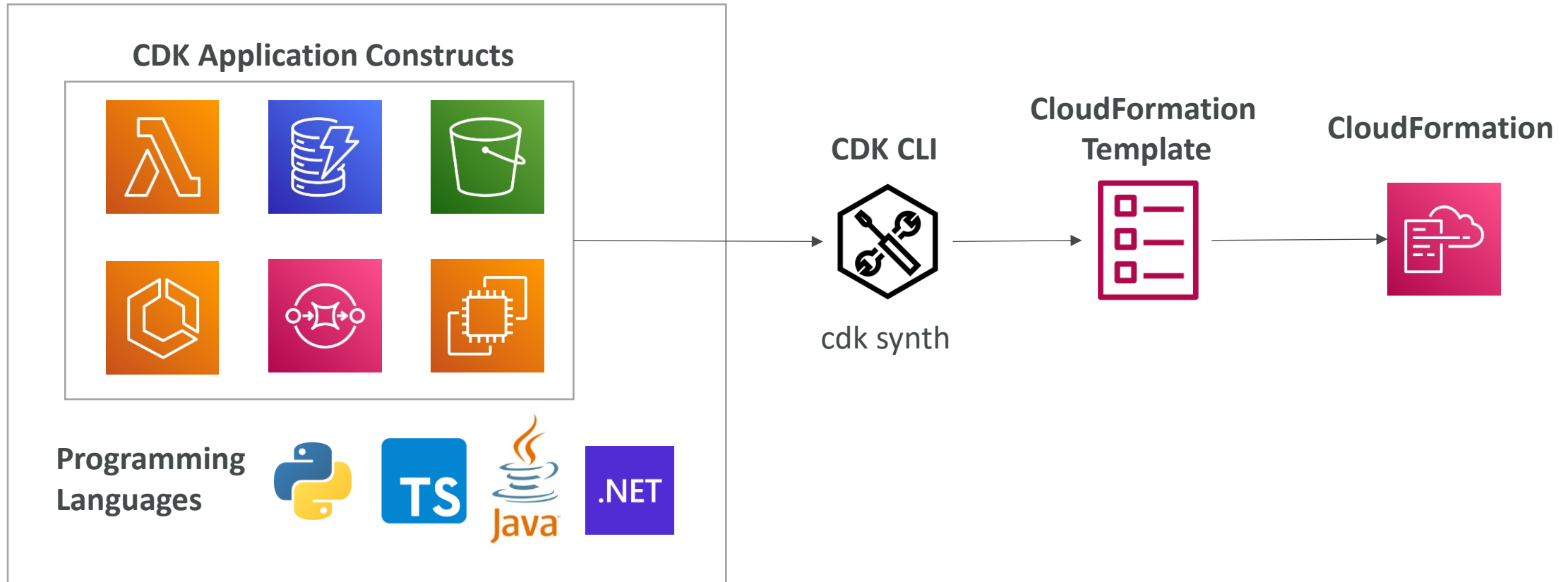
```
export class MyEcsConstructStack extends core.Stack {
  constructor(scope: core.App, id: string, props?: core.StackProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, "MyVpc", {
      maxAzs: 1 // Default is all AZs in region
    });

    const cluster = new ecs.Cluster(this, "MyCluster", {
      vpc: vpc
    });

    // create a load-balanced Fargate service and make it public
    new ecs_patterns.ApplicationLoadBalancedFargateService(this, "My
      cluster: cluster, // Required
      cpu: 512, // Default is 256
      desiredCount: 6, // Default is 1
      taskImageOptions: { image: ecs.ContainerImage.fromRegistry("an
      memoryLimitMiB: 2048, // Default is 512
      publicLoadBalancer: true // Default is false
    });
  }
}
```


CDK in a diagram



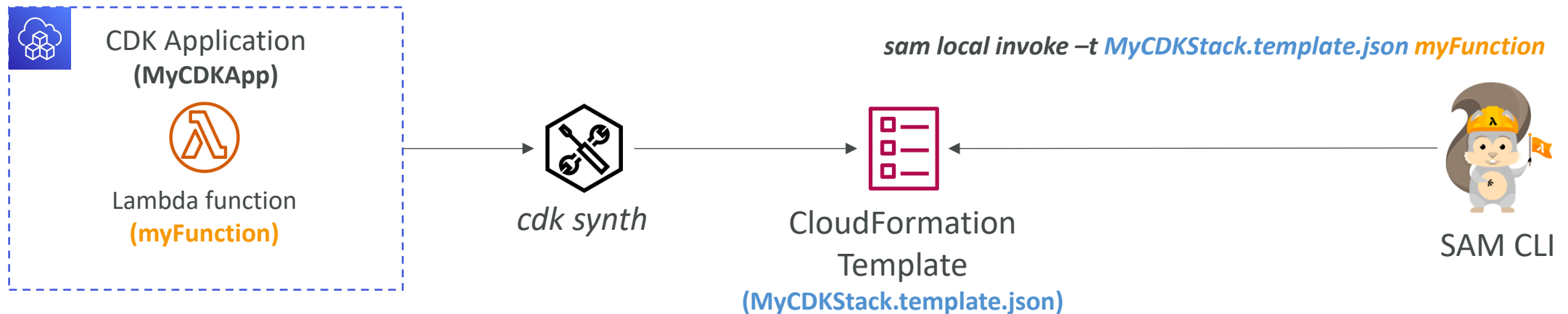
CDK vs SAM



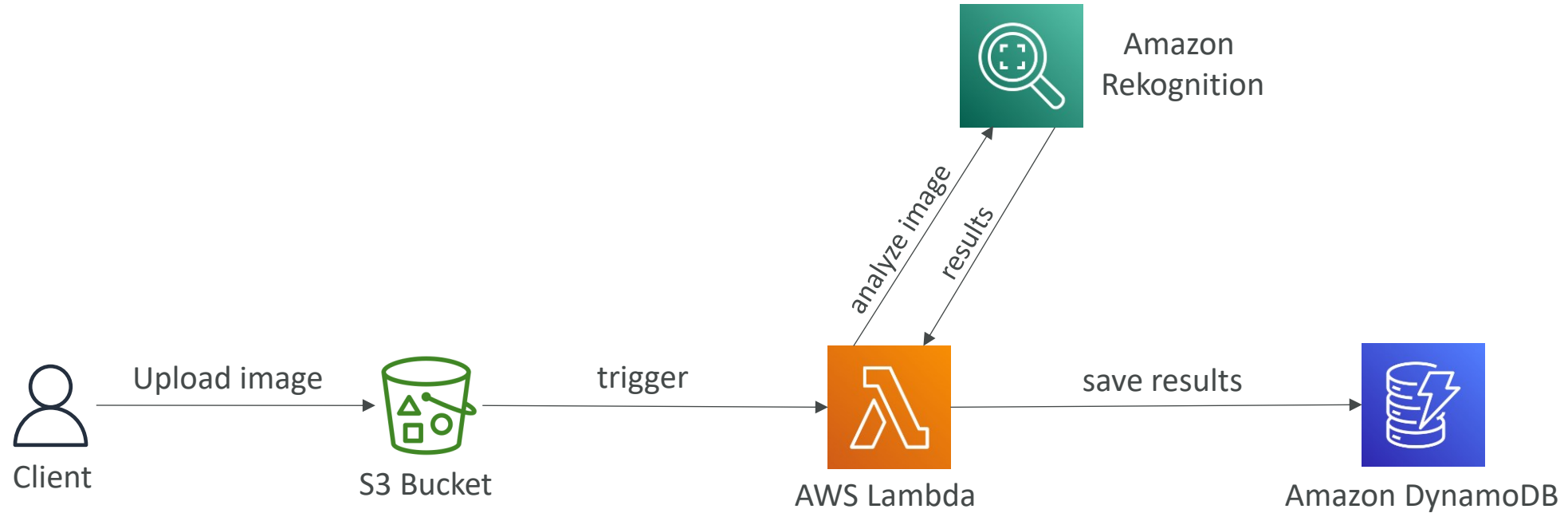
- SAM:
 - Serverless focused
 - Write your template declaratively in JSON or YAML
 - Great for quickly getting started with Lambda
 - Leverages CloudFormation
- CDK:
 - All AWS services
 - Write infra in a programming language JavaScript/TypeScript, Python, Java, and .NET
 - Leverages CloudFormation

CDK + SAM

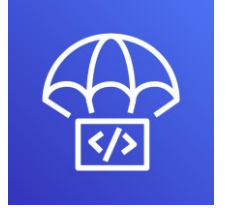
- You can use SAM CLI to locally test your CDK apps
- **You must first run *cdk synth***



CDK Hands-On

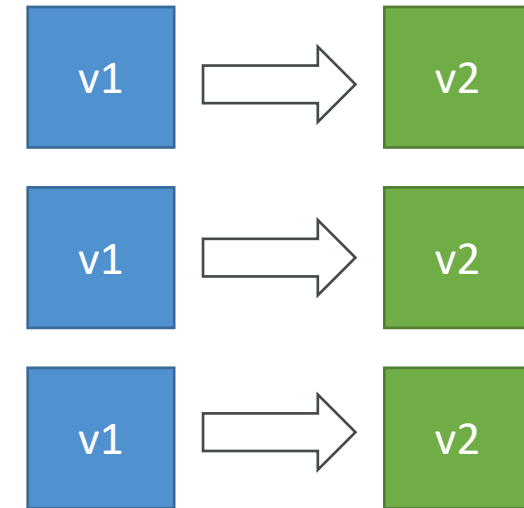


AWS CodeDeploy

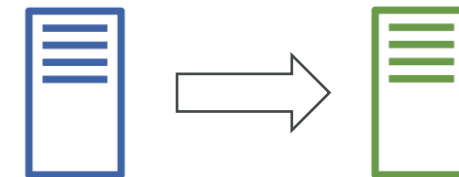


- We want to deploy our application **automatically**
- **Works with EC2 Instances**
- **Works with On-Premises Servers**
- **Hybrid service**
- Servers / Instances must be provisioned and configured ahead of time with the CodeDeploy Agent

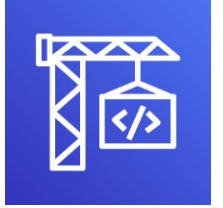
EC2 Instances being upgraded



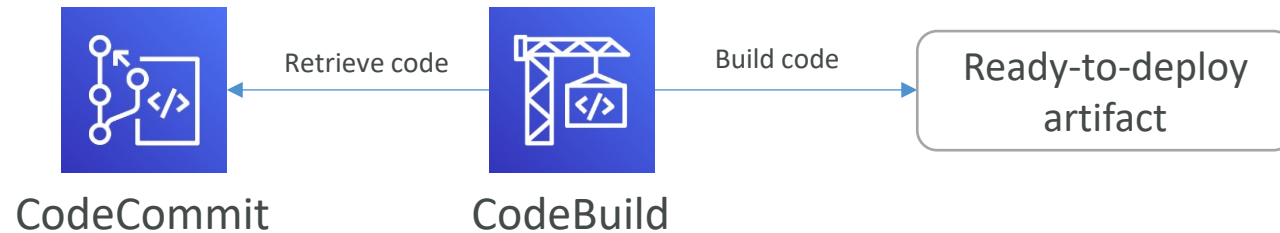
On-premises Servers being upgraded



AWS CodeBuild



- Code building service in the cloud (name is obvious)
- **Compiles source code, run tests, and produces packages that are ready to be deployed (by CodeDeploy for example)**

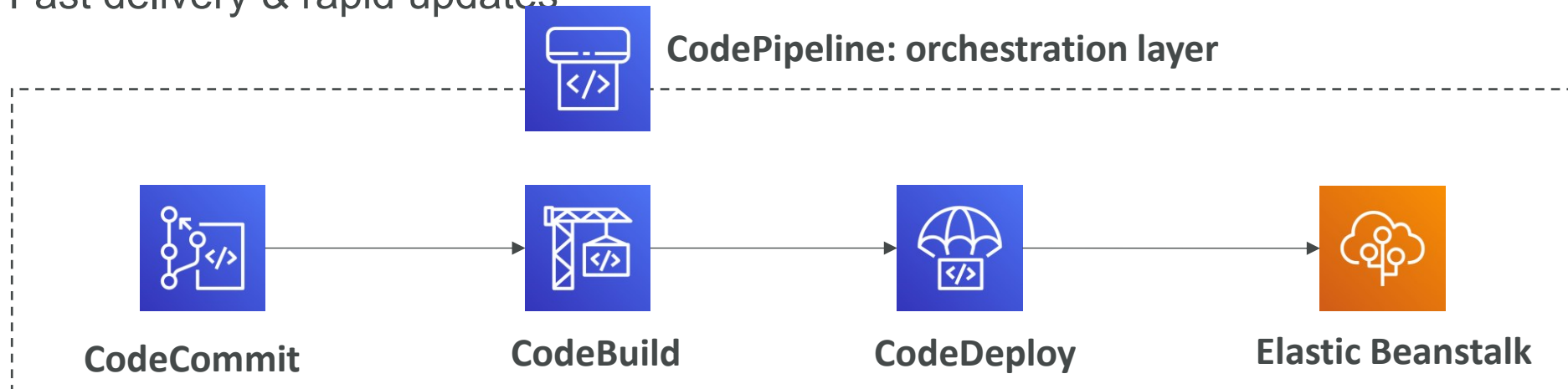


- **Benefits:**
 - Fully managed, serverless
 - Continuously scalable & highly available
 - Secure
 - Pay-as-you-go pricing – only pay for the build time

AWS CodePipeline



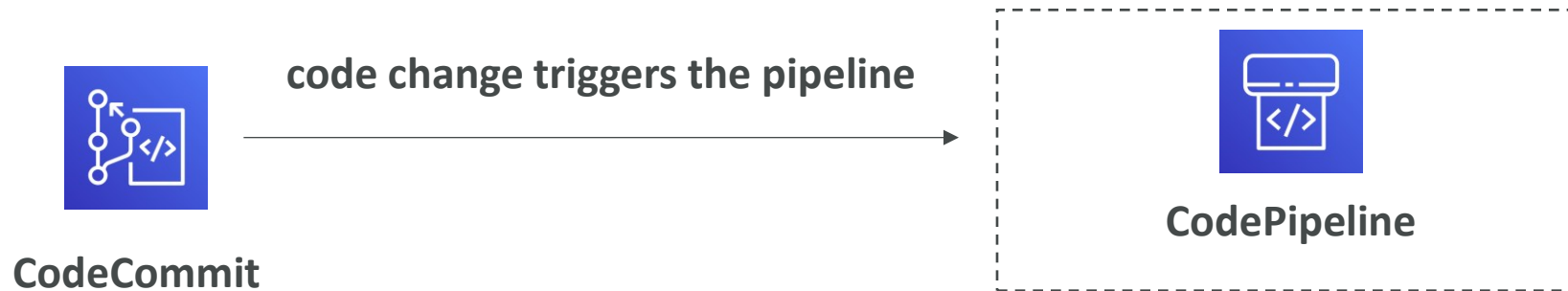
- **Orchestrate the different steps to have the code automatically pushed to production**
 - Code => Build => Test => Provision => Deploy
 - Basis for CI/CD (Continuous Integration & Continuous Delivery)
- **Benefits:**
 - Fully managed, compatible with CodeCommit, CodeBuild, CodeDeploy, Elastic Beanstalk, CloudFormation, GitHub, 3rd-party services (GitHub...) & custom plugins...
 - Fast delivery & rapid updates



CodeCommit and CodeConnections



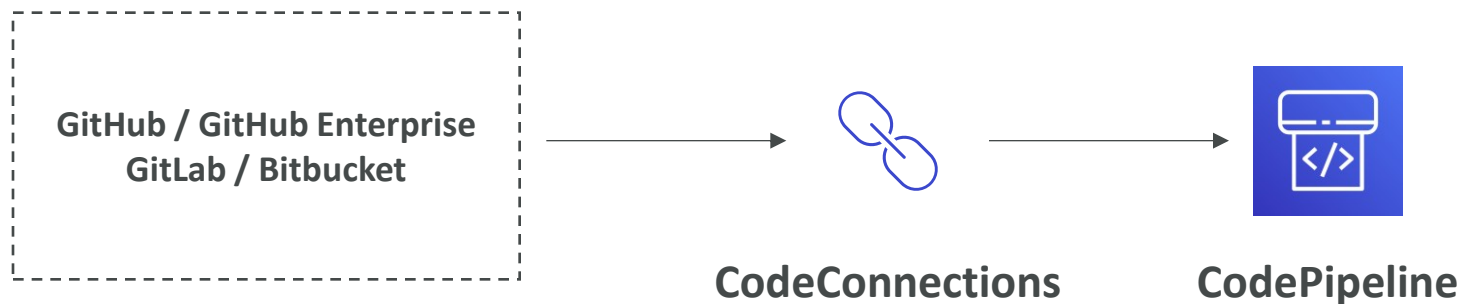
- **AWS CodeCommit – AWS-hosted, private Git repository (version control)**
 - Acts as the Source stage for CodePipeline in a CI/CD pipeline
 - A source-code change (git push) can automatically trigger the pipeline
 - Example: push updated ML training code => pipeline retrains and redeploys the model



CodeCommit and CodeConnections



- **AWS CodeConnections – managed connections between AWS services and third-party Git repositories**
 - Providers: GitHub, GitHub Enterprise, GitLab, Bitbucket
 - CodePipeline uses the connection as its Source => no third-party credentials stored in the pipeline
- **Exam tip: CodeCommit = source-code repository, CodeConnections = connects AWS to external repos**



CI/CD for AI Artifacts



- **AI applications introduce versioned artifacts beyond source code**
 - Prompt templates & prompt versions => Amazon Bedrock Prompt Management
 - ML models & fine-tuned / custom foundation models => SageMaker Model Registry, MLflow
 - Agents & agent configurations => Bedrock agent versions & aliases
 - Knowledge-base / retrieval configuration, versioned alongside the app



**Application / source
code**



**Prompt templates &
versions**



**ML models & custom
FMs**



Agents & agent configs



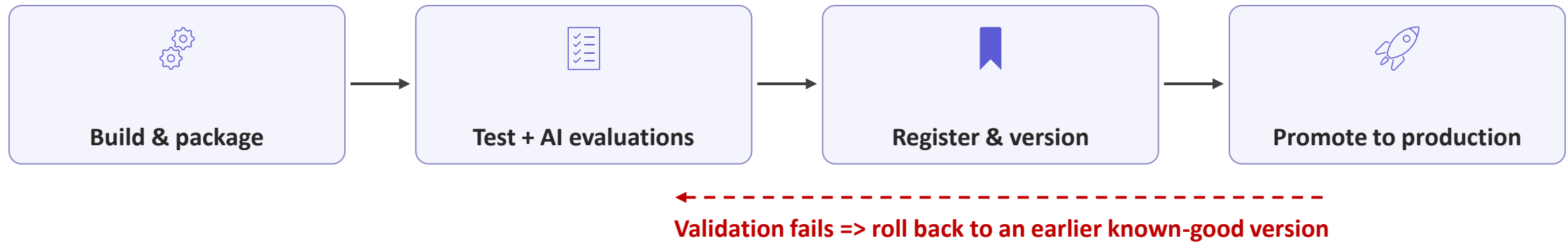
**Knowledge base /
retrieval config**

Each artifact type has its own version history and registry

CI/CD for AI Artifacts



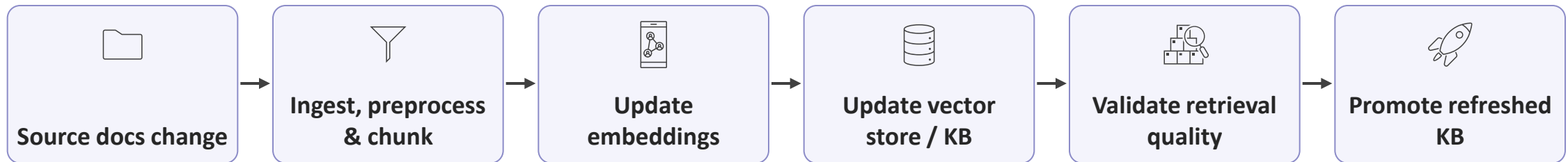
- **A CI/CD workflow handles AI artifacts just like code**
 - CodePipeline / CodeBuild orchestrate builds, automated tests + AI evaluations, and deployments
 - Only approved, registered versions are promoted to production
- **Exam tip: AI artifacts must be versioned, tested, traceable, and deployable – just like code**



Automating RAG / Knowledge Base Updates



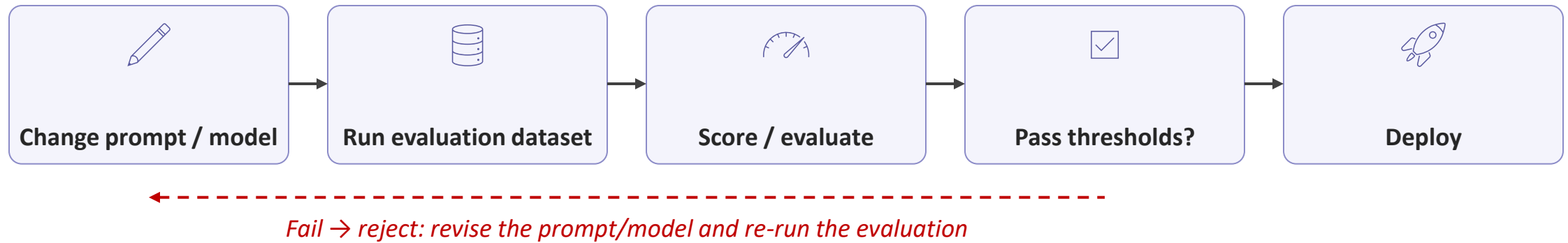
- **When source documents change, the retrieval index must stay in sync**
 - Bedrock Knowledge Bases sync is incremental – StartIngestionJob re-processes added, modified & deleted docs
 - Trigger it event-driven or on a schedule; check job status and validate retrieval before promoting
- **Exam tip: a refresh succeeds only when ingestion completes and retrieval quality is validated**



Prompt and GenAI Regression Testing in CI/CD



- **Foundation model responses vary between runs – tests judge quality against criteria, not exact string matches**
 - Maintain a fixed prompt/evaluation dataset: normal cases, edge cases & key business scenarios
 - Optionally add reference (ground-truth) responses when appropriate
 - Keep a held-out test set separate from the data used to optimize the prompt or model



Prompt and GenAI Regression Testing in CI/CD



- **Run every candidate prompt or model version against the evaluation dataset before promotion**
 - Score quality dimensions: correctness, completeness, relevance, safety & task-specific criteria
 - Set acceptance thresholds so the pipeline can block deployment when quality meaningfully regresses
- **Exam tip: the quality gate compares evaluation scores to thresholds – not exact output strings**

Amazon Bedrock Model Evaluations – choose the method per test



Built-in metrics



Custom metrics

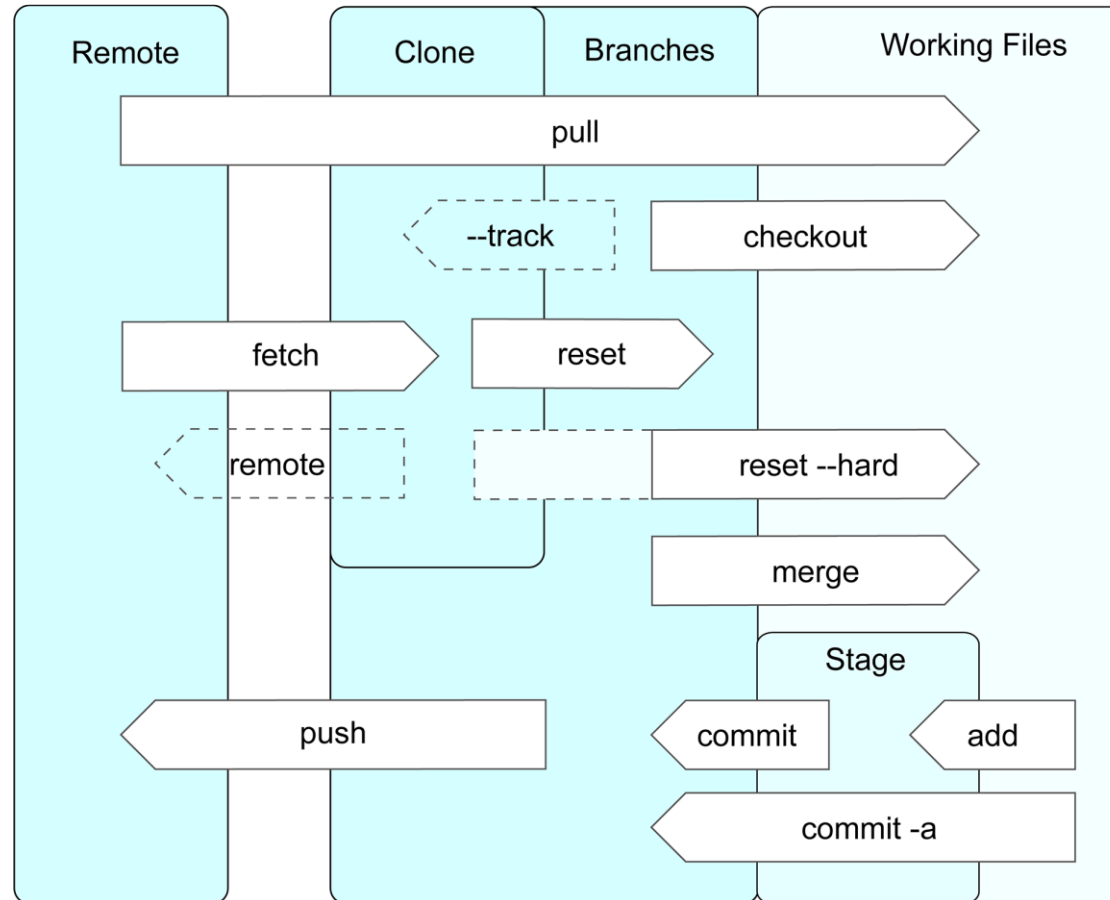


Human evaluation



LLM-as-a-judge

Git review



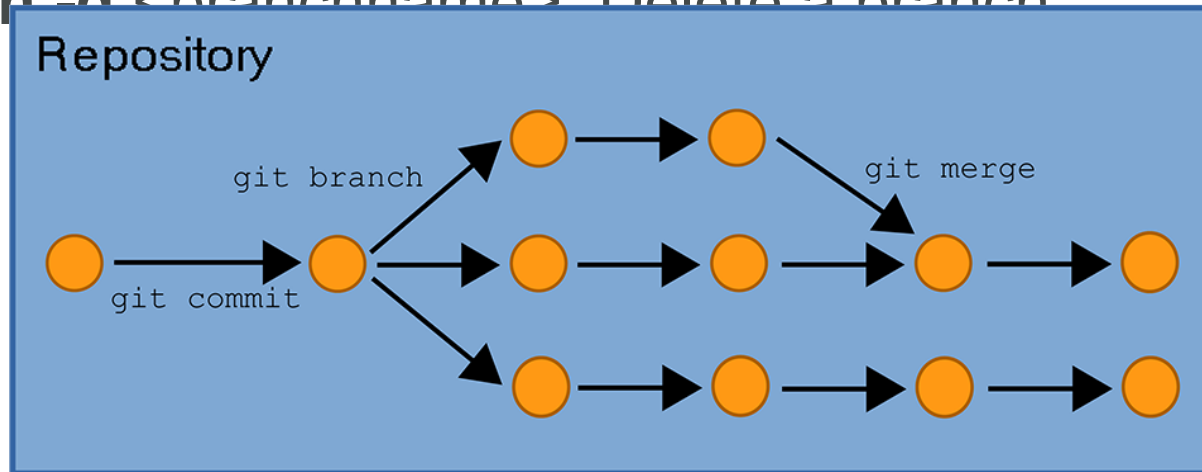
Daniel Kinzler, CC BY 3.0 <<https://creativecommons.org/licenses/by/3.0/>>, via Wikimedia Commons

Common git commands

- Setting Up and Configuration:
 - **git init**: Initialize a new Git repository.
 - **git config**: Set configuration values for user info, aliases, and more.
 - `git config --global user.name "Your Name"`: Set your name.
 - `git config --global user.email "your@email.com"`: Set your email.
- Basic Commands:
 - **git clone** <repository>: Clone (or download) a repository from an existing URL.
 - **git status**: Check the status of your changes in the working directory.
 - **git add** <filename>: Add changes in the file to the staging area.
 - `git add .`: Add all new and changed files to the staging area.
 - **git commit -m** "Commit message here": Commit the staged changes with a message.
 - **git log**: View commit logs.

Branching with git

- **git branch**: List all local branches.
 - `git branch <branchname>`: Create a new branch.
- **git checkout <branchname>**: Switch to a specific branch.
 - `git checkout -b <branchname>`: Create a new branch and switch to it.
- **git merge <branchname>**: Merge the specified branch into the current branch.
- **git branch -d <branchname>**: Delete a branch



Felix Dreissig, noris network AG

Remote repositories

- **git remote add** <name> <url>: Add a remote repository.
- **git remote**: List all remote repositories.
- **git push** <remote> <branch>: Push a branch to a remote repository.
- **git pull** <remote> <branch>: Pull changes from a remote repository branch into the current local branch.

Undoing changes

- **git reset**: Reset your staging area to match the most recent commit, without affecting the working directory.
- **git reset --hard**: Reset the staging area and the working directory to match the most recent commit.
- **git revert <commit>**: Create a new commit that undoes all of the changes from a previous commit.

Advanced git

- **git stash**: Temporarily save changes that are not yet ready for a commit.
 - git stash pop: Restore the most recently stashed changes.
- **git rebase <branch>**: Reapply changes from one branch onto another, often used to integrate changes from one branch into another.
- **git cherry-pick <commit>**: Apply changes from a specific commit to the current branch.

Git collaboration and inspection

- **git blame** <file>: Show who made changes to a file and when.
- **git diff**: Show changes between commits, commit and working tree, etc.
- **git fetch**: Fetch changes from a remote repository without merging them.

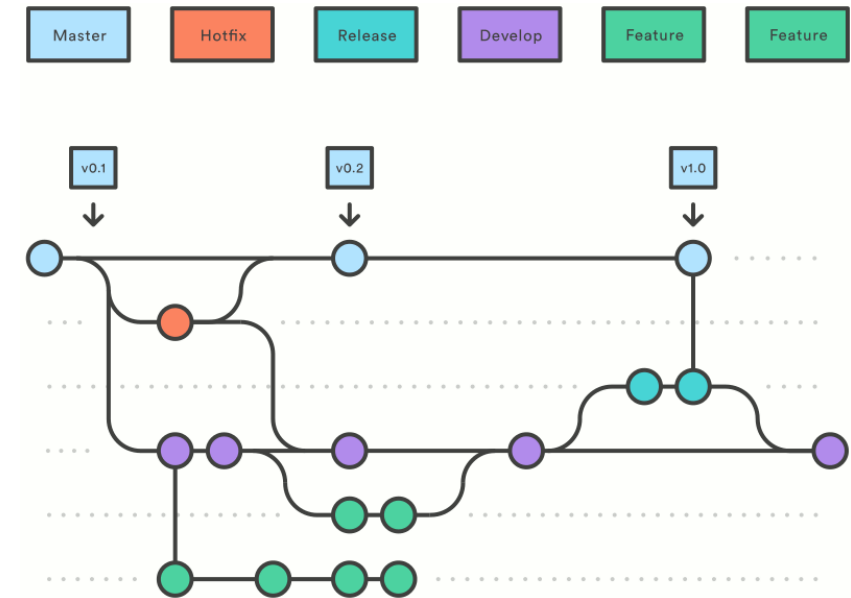
Git maintenance and data recovery

- **git fsck**: Check the database for errors.
- **git gc**: Clean up and optimize the local repository.
- **git reflog**: Record when refs were updated in the local repository, useful for recovering lost commits.



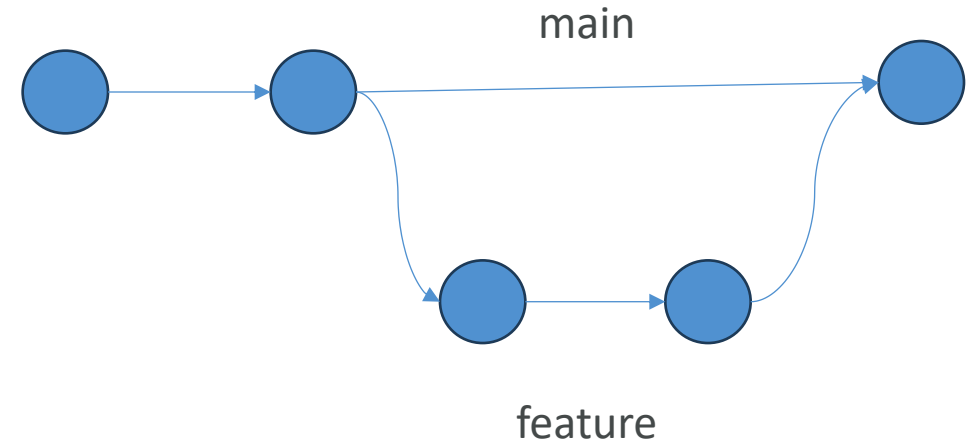
Git Flow

- Considered old-fashioned compared to trunk-based approaches, but still widely used for scrum
- Main / master branch: current release in production
- Develop: Changes since last release
- Feature: Branches from Develop for new features
- Release: Merge changes with Develop



GitHub Flow

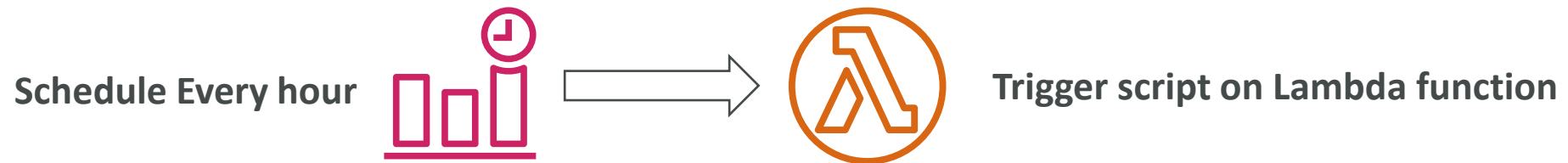
- Only two types of branches
 - Main (master) branch
 - Feature branches
 - Branch off main to work on new features
 - When tested, merge to main
 - Deploy to production immediately
- That's it
- Requires ability to release quickly
 - Automated tests and deployments
 - Maybe release multiple times *per day*



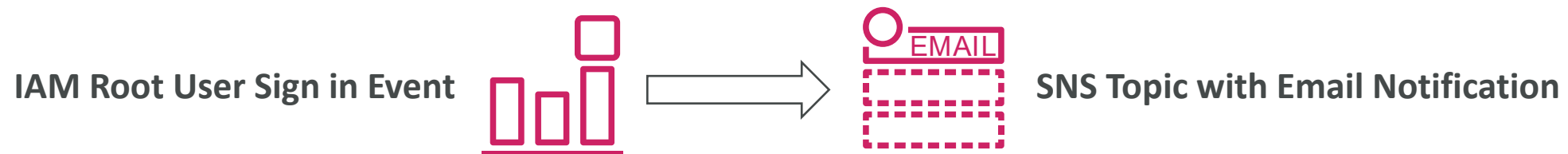
Amazon EventBridge (formerly CloudWatch Events)



- Schedule: Cron jobs (scheduled scripts)

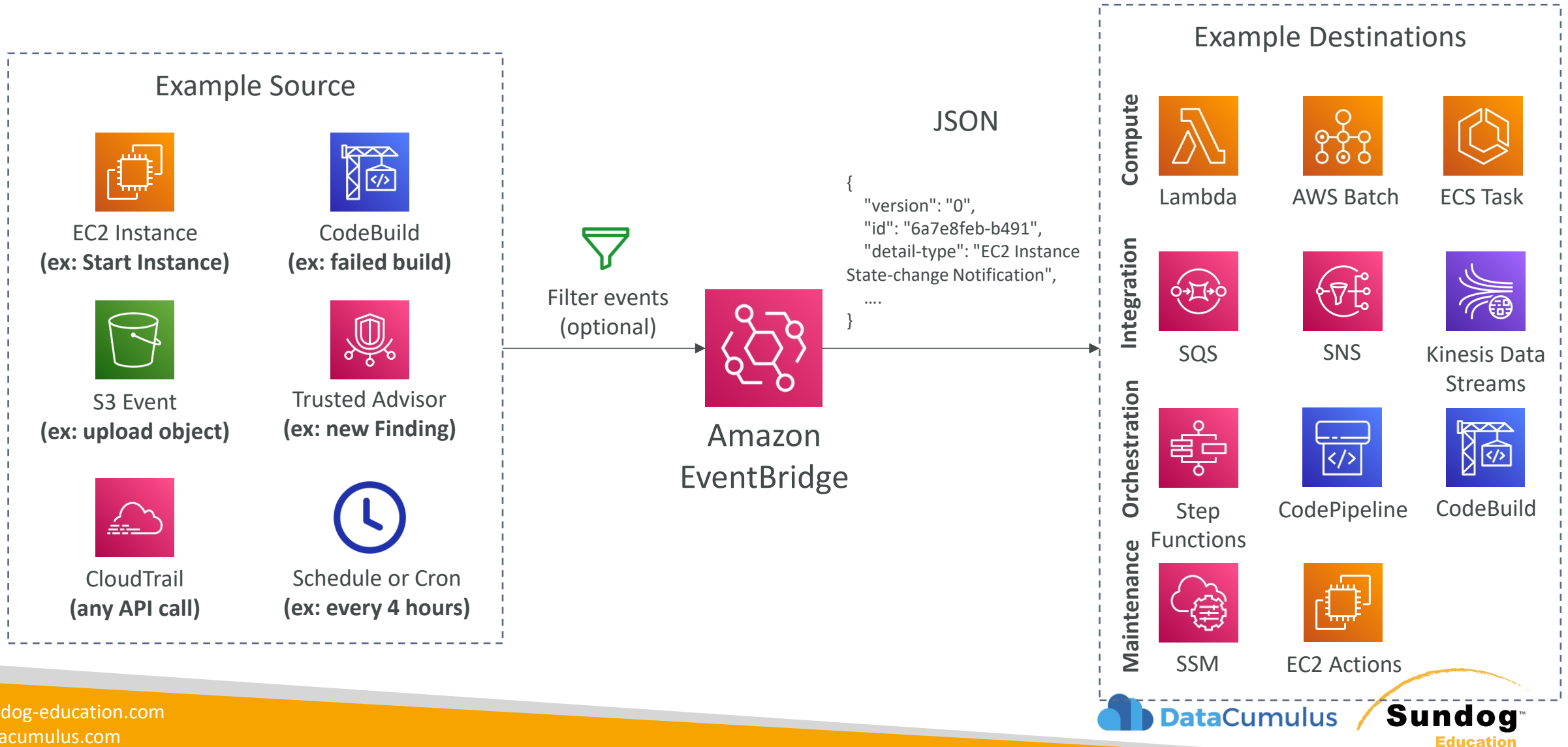


- Event Pattern: Event rules to react to a service doing something

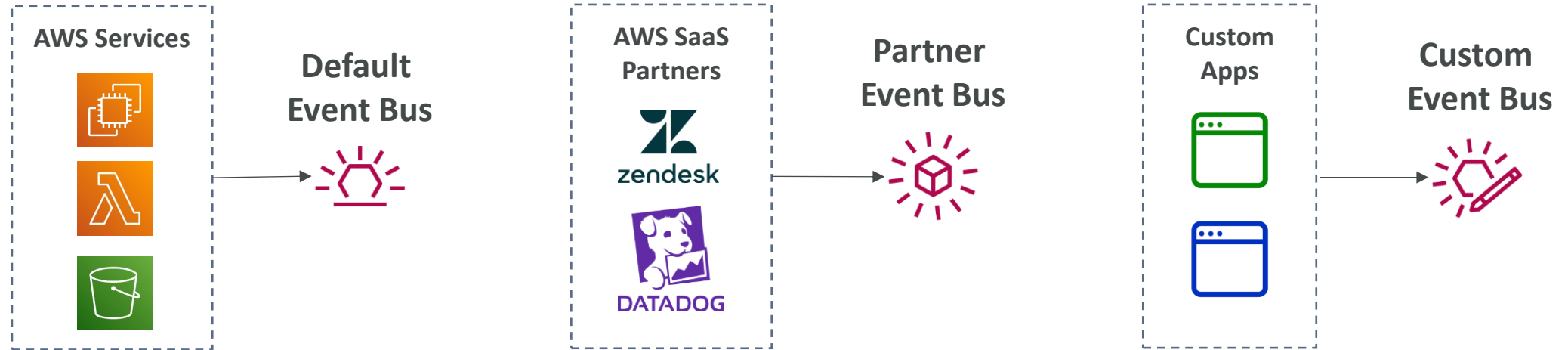


- Trigger Lambda functions, send SQS/SNS messages...

Amazon EventBridge Rules



Amazon EventBridge



- Event buses can be accessed by other AWS accounts using Resource-based Policies
- You can **archive events** (all/filter) sent to an event bus (indefinitely or set period)
- Ability to **replay archived events**

Amazon EventBridge – Schema Registry

- EventBridge can analyze the events in your bus and infer the **schema**
- The **Schema Registry** allows you to generate code for your application, that will know in advance how data is structured in the event bus
- Schema can be versioned

aws.codepipeline@CodePipelineActionExecuti...

Schema details

Schema name	Last modified	Schema ARN
aws.codepipeline@CodePipelineActionExecutionStateChange	Dec 1, 2019, 12:11 AM GMT	-
Description	Schema registry	Number of versions
Schema for event type CodePipelineActionExecutionStateChange, published by AWS service aws.codepipeline	aws.events	1
		Schema type
		OpenAPI 3.0

Version 1 Created on Dec 1, 2019, 12:11 AM GMT

Action [Download code bindings](#)

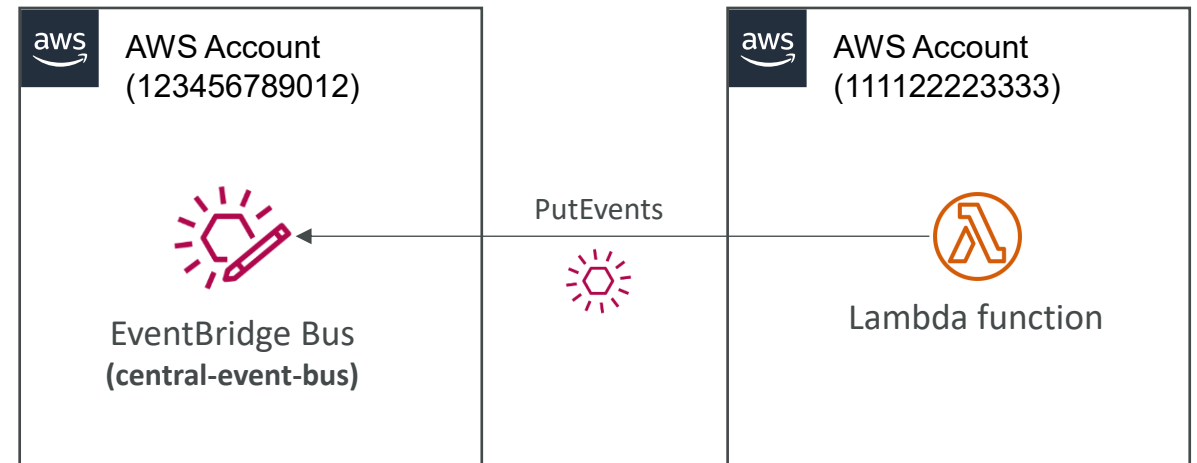
```
1 {
2   "openapi": "3.0.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "CodePipelineActionExecutionStateChange"
6   },
7   "paths": {},
8   "components": {
9     "schemas": {
10      "AWSEvent": {
```

Amazon EventBridge – Resource-based Policy

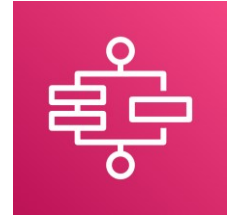
- Manage permissions for a specific Event Bus
- Example: allow/deny events from another AWS account or AWS region
- Use case: aggregate all events from your AWS Organization in a single AWS account or AWS region

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "events:PutEvents",
      "Principal": { "AWS": "111122223333" },
      "Resource": "arn:aws:events:us-east-1:123456789012:
event-bus/central-event-bus"
    }
  ]
}
```

Allow **events** from another AWS account



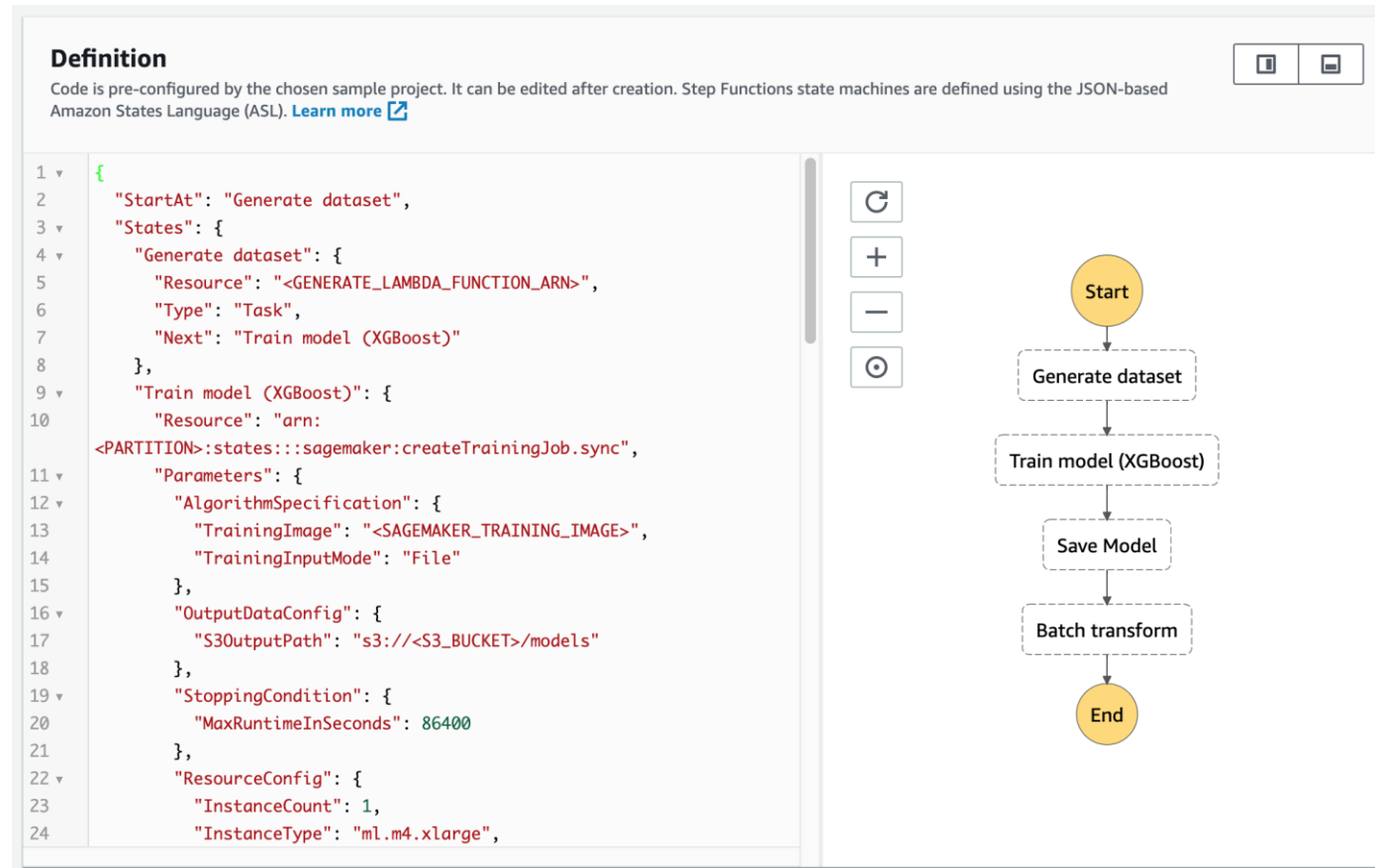
AWS



- **Use to design workflows**
- Easy visualizations
- Advanced Error Handling and Retry mechanism outside the code
- Audit of the history of workflows
- Ability to “Wait” for an arbitrary amount of time
- Max execution time of a State Machine is 1 year

– Examples

Train a Machine Learning Model



– Examples

Tune a Machine Learning Model

Definition

Code is pre-configured by the chosen sample project. It can be edited after creation. Step Functions state machines are defined using the JSON-based Amazon States Language (ASL). [Learn more](#)

```
1 {
2   "StartAt": "Generate Training Dataset",
3   "States": {
4     "Generate Training Dataset": {
5       "Resource": "<GENERATE_LAMBDA_FUNCTION_ARN>",
6       "Type": "Task",
7       "Next": "HyperparameterTuning (XGBoost)"
8     },
9     "HyperparameterTuning (XGBoost)": {
10      "Resource": "arn:
11      <PARTITION>:states::sagemaker:createHyperParameterTuningJob.sync",
12      "Parameters": {
13        "HyperParameterTuningJobName.$": "
14        <JOB_NAME_FROM_LAMBDA>",
15        "HyperParameterTuningJobConfig": {
16          "Strategy": "Bayesian",
17          "HyperParameterTuningJobObjective": {
18            "Type": "Minimize",
19            "MetricName": "validation:rmse"
20          },
21          "ResourceLimits": {
22            "MaxNumberOfTrainingJobs": 2,
23            "MaxParallelTrainingJobs": 2
24          }
25        },
26        "ParameterRanges": {
```

```
graph TD
    Start((Start)) --> Generate[Generate Training Dataset]
    Generate --> Tune[HyperparameterTuning (XGBoost)]
    Tune --> ExtractPath[Extract Model Path]
    ExtractPath --> SaveModel[HyperparameterTuning - Save Model]
    SaveModel --> ExtractName[Extract Model Name]
    ExtractName --> Batch[Batch transform]
    Batch --> End((End))
```


– Examples

Manage a Batch Job

Definition

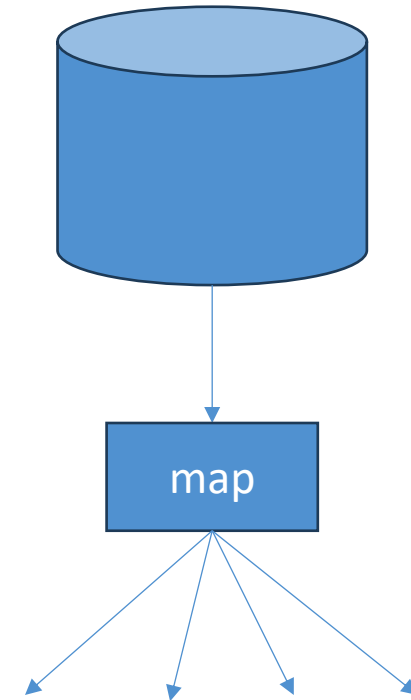
Code is pre-configured by the chosen sample project. It can be edited after creation. Step Functions state machines are defined using the JSON-based Amazon States Language (ASL). [Learn more](#)

```
1 {
2   "Comment": "An example of the Amazon States Language for
3   notification on an AWS Batch job completion",
4   "StartAt": "Submit Batch Job",
5   "TimeoutSeconds": 3600,
6   "States": {
7     "Submit Batch Job": {
8       "Type": "Task",
9       "Resource": "arn:<PARTITION>:states:::batch:submitJob.sync",
10      "Parameters": {
11        "JobName": "BatchJobNotification",
12        "JobQueue": "<BATCH_QUEUE_ARN>",
13        "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>"
14      },
15      "Next": "Notify Success",
16      "Catch": [
17        {
18          "ErrorEquals": [ "States.ALL" ],
19          "Next": "Notify Failure"
20        }
21      ],
22      "Notify Success": {
23        "Type": "Task",
24        "Resource": "arn:<PARTITION>:states:::sns:publish",
```

```
graph TD
    Start((Start)) --> SubmitBatchJob[Submit Batch Job]
    SubmitBatchJob --> NotifySuccess[Notify Success]
    SubmitBatchJob --> NotifyFailure[Notify Failure]
    NotifySuccess --> End((End))
    NotifyFailure --> End
```

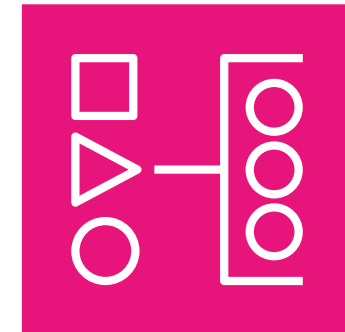
AWS

- Your workflow is called a *state machine*
- Each step in a workflow is a *state*
- Types of states
 - **Task**: Does something with Lambda, other AWS services, or third party API's
 - **Choice**: Adds conditional logic via Choice Rules (ie, comparisons)
 - **Wait**: Delays state machine for a specified time
 - **Parallel**: Add separate branches of execution
 - **Map**: Run a set of steps for each item in a dataset, in parallel
 - **This is most relevant to data engineering!**
Works with JSON, S3 objects, CSV files
 - Also **Pass, Succeed, Fail**



Amazon Managed Workflows for Apache Airflow (MWAA)

- Apache Airflow is batch-oriented workflow tool
- Develop, schedule, and monitor your workflows
- Workflows are defined as Python code that creates a Directed Acyclic Graph (DAG)
- Amazon MWAA provides a managed service for Apache Airflow so you don't have to deal with installing or maintaining it
- Use cases:
 - Complex workflows
 - ETL coordination
 - Preparing ML data



Amazon Managed Workflows
for Apache Airflow

Sample DAG in Airflow

```
from datetime import datetime

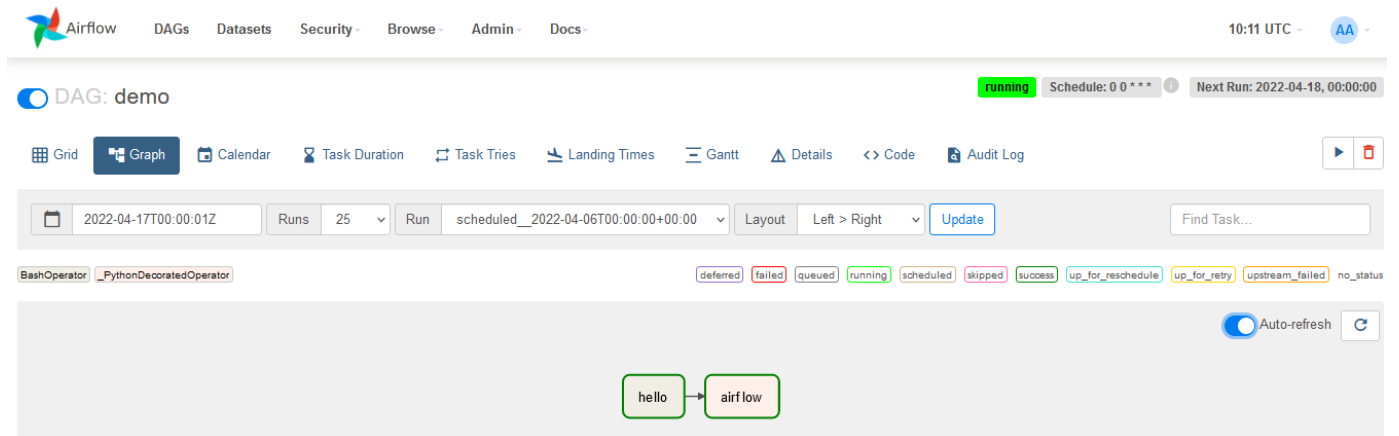
from airflow import DAG
from airflow.decorators import task
from airflow.operators.bash import BashOperator

# A DAG represents a workflow, a collection of tasks
with DAG(dag_id="demo", start_date=datetime(2022, 1, 1),
        schedule="0 0 * * *") as dag:

    # Tasks are represented as operators
    hello = BashOperator(task_id="hello", bash_command="echo
hello")

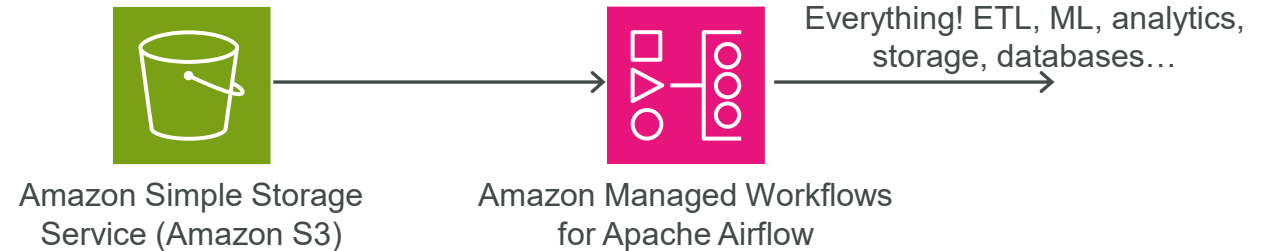
    @task()
    def airflow():
        print("airflow")

    # Set dependencies between tasks
    hello >> airflow()
```



Airflow + MWAA

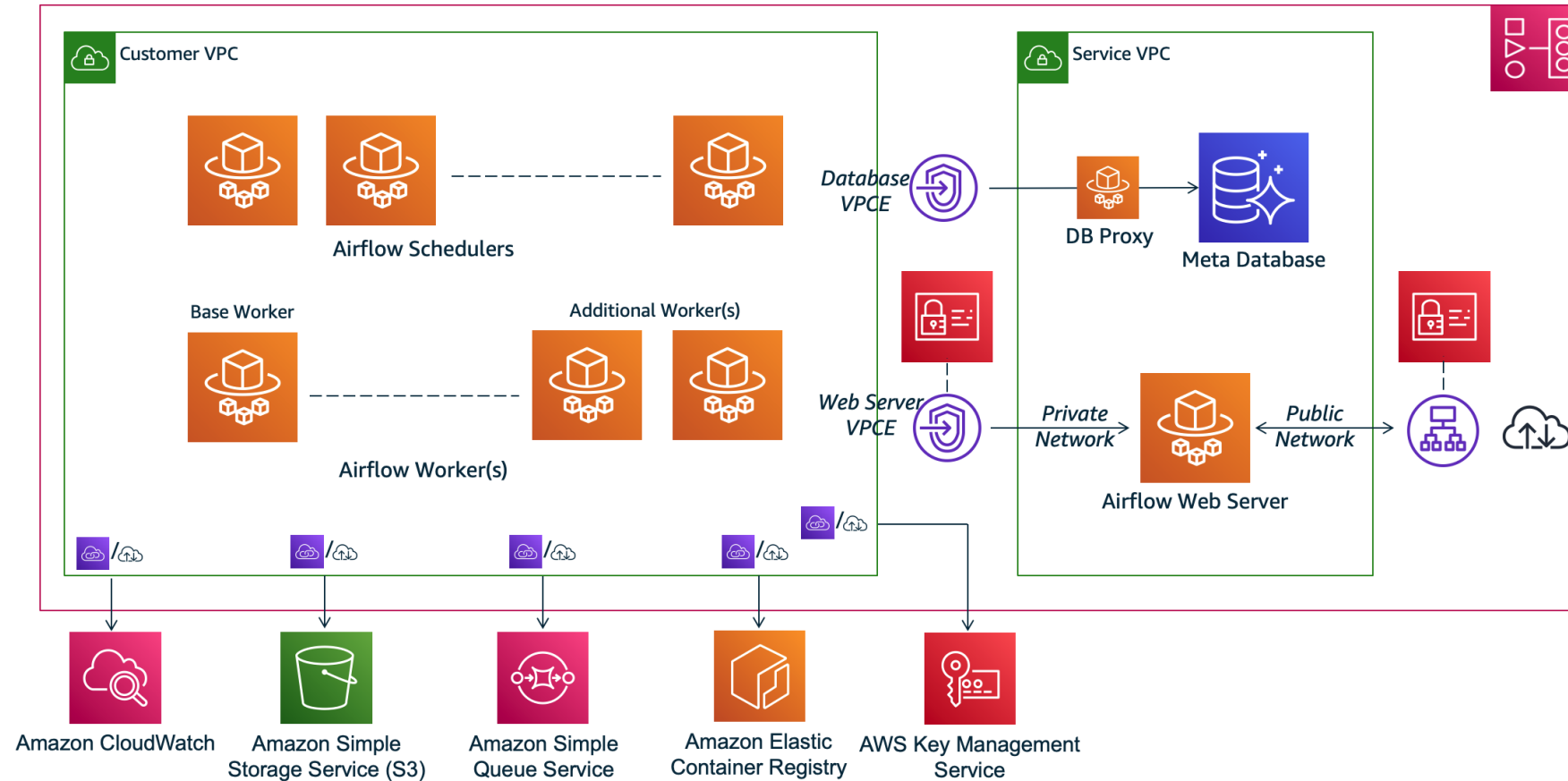
- Your DAGs (Python code) are uploaded into S3
 - May also zip it together with required plugins and requirements
- Amazon MWAA picks it up, and orchestrates and schedules the pipelines defined by each DAG.
- Runs within a VPC
 - In at least 2 AZ's
- Private or public endpoints
 - IAM-managed
 - (Access to Airflow Web Server)
- Automatic scaling
 - Airflow Workers autoscale up to the limits you define



Amazon MWAA Integration

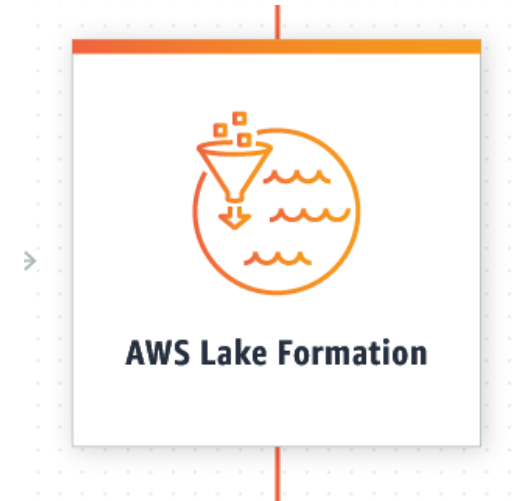
- Leverages open-source integrations
 - Athena, Batch, CloudWatch, DynamoDB, DataSync
 - EMR, Fargate, EKS, Kinesis, Glue, Lambda
 - Redshift, SQS, SNS, SageMaker, S3... and more
 - Security services (AWS Secrets Manager, etc)
- Schedulers and Workers are AWS Fargate containers

Amazon MWAA Architecture

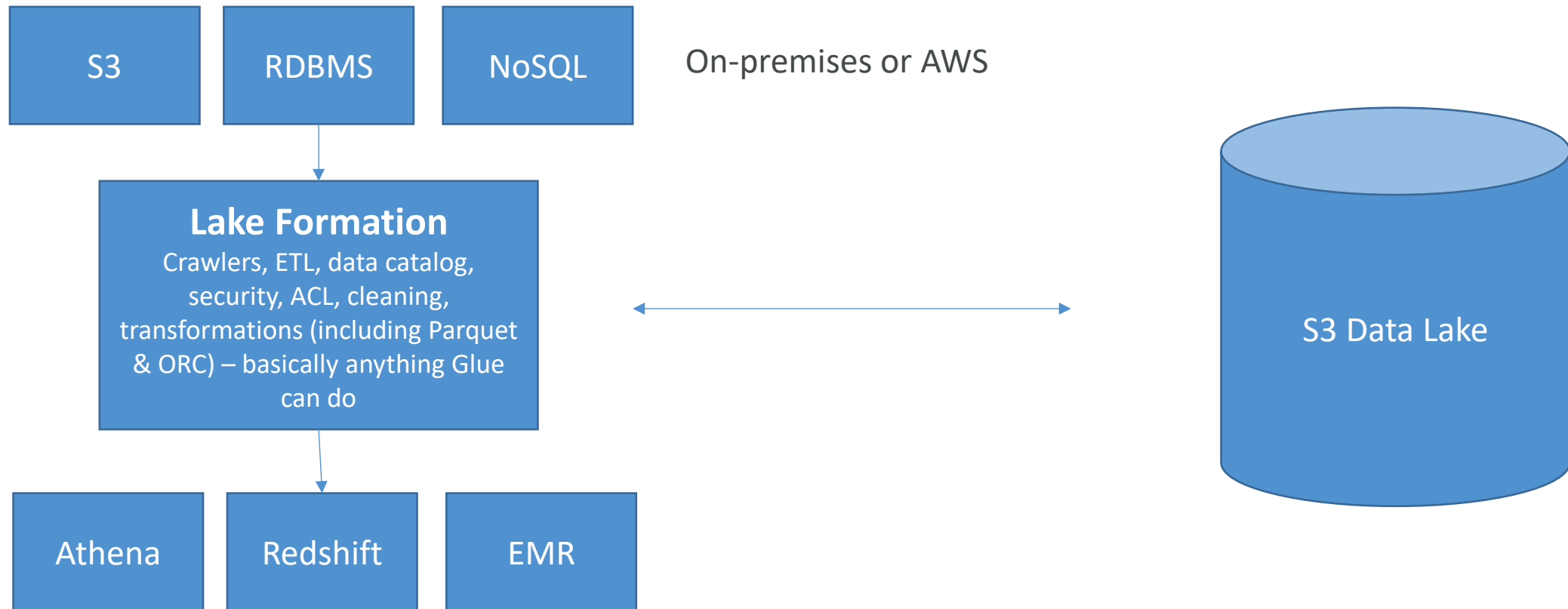


AWS Lake Formation

- “Makes it easy to set up a secure data lake in days”
- Loading data & monitoring data flows
- Setting up partitions
- Encryption & managing keys
- Defining transformation jobs & monitoring them
- Access control
- Auditing
- Built on top of Glue



AWS Lake Formation

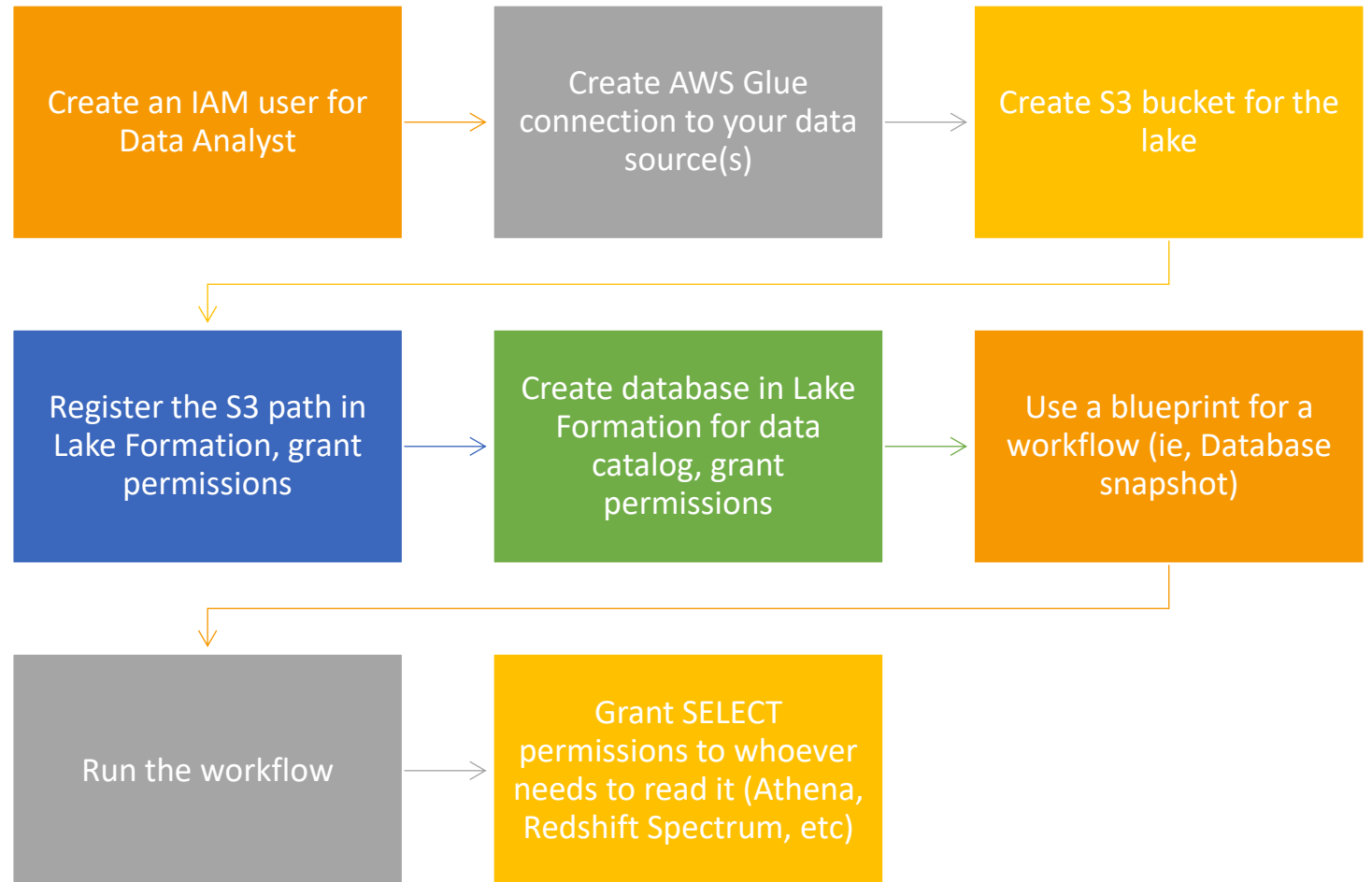


AWS Lake Formation: Pricing

- No cost for Lake Formation itself
- But underlying services incur charges
 - Glue
 - S3
 - EMR
 - Athena
 - Redshift



AWS Lake Formation: Building a Data Lake



AWS Lake Formation: The Finer Points

- Cross-account Lake Formation permission
 - Recipient must be set up as a data lake administrator
 - Can use AWS Resource Access Manager for accounts external to your organization
 - IAM permissions for cross-account access
- Lake Formation does not support manifests in Athena or Redshift queries
- IAM permissions on the KMS encryption key are needed for encrypted data catalogs in Lake Formation
- IAM permissions needed to create blueprints and workflows

AWS Lake Formation: Governed Tables and Security

- Now supports “Governed Tables” that support ACID transactions across multiple tables
 - New type of S3 table
 - Can’t change choice of governed afterwards
 - Works with streaming data too (Kinesis)
 - Can query with Athena
- Storage Optimization with Automatic Compaction
- Granular Access Control with Row and Cell-Level Security
 - Both for governed and S3 tables
- Above features incur additional charges based on usage

Data Permissions in Lake Formation

- Can tie to IAM users/roles, SAML, or external AWS accounts
- Can use policy tags on databases, tables, or columns
- Can select specific permissions for tables or columns

The screenshot shows the 'Grant Permissions' interface in the AWS Lake Formation console. The breadcrumb navigation at the top indicates 'AWS Lake Formation > Data permissions'. The main heading is 'Grant Permissions', followed by the instruction 'Choose the access permissions to grant to this or an external account.'

The interface is divided into three main sections:

- Principals:** This section allows selecting the principal to grant permissions to. It includes three radio buttons: 'IAM users and roles' (selected), 'SAML users and groups', and 'External accounts'. Below the radio buttons, there is a dropdown menu 'Choose IAM principals to add' and a list of selected principals, currently showing 'secure-if-admin User'.
- Policy tags or catalog resources:** This section allows selecting the resources to grant permissions to. It includes two radio buttons: 'Resources matched by policy tags (recommended)' and 'Named data catalog resources' (selected). Below the radio buttons, there are two dropdown menus: 'Choose databases' (showing 'secure-db 689900203792') and 'Choose tables' (showing 'All tables').
- Permissions:** This section allows selecting the permissions to grant. It includes two radio buttons: 'Table permissions' (selected) and 'Column-based permissions'. Below the radio buttons, there are two sections: 'Table permissions' and 'Grantable permissions'. Each section has a list of permissions: 'Select', 'Insert', 'Delete', 'Describe', 'Alter', and 'Drop'. The 'Super' permission is checked in both sections, with a note stating 'This permission is the union of the individual permissions above and supercedes them. Learn More'.

At the bottom right of the interface, there are 'Cancel' and 'Grant' buttons.

Data Filters in Lake Formation

- Column, row, or cell-level security
- Apply when granting SELECT permission on tables
- “All columns” + row filter = row-level security
- “All rows” + specific columns = column-level security
- Specific columns + specific rows = cell-level security
- Create filters via the console (seen here) or via `CreateDataCellsFilter` API.

The screenshot shows the 'Create data filter' console interface. It includes fields for 'Data filter name' (set to 'restrict-pharma'), 'Target database' (set to 'sales'), and 'Target table' (set to 'orders'). Under 'Column-level access', the 'Exclude columns' option is selected. The 'Select columns' field shows 'customer_name'. The 'Row filter expression' field contains the SQL snippet 'product_type='pharma''. At the bottom, there are 'Cancel' and 'Create filter' buttons.

Create data filter

Data filter name
Enter a name that describes this data access filter.
restrict-pharma
Name may contain letters (A-Z), numbers (0-9), hyphens (-), or under-scores (_), and be less than 256 characters.

Target database
Select the database that contains the target table.
Choose databases ▼ Load more
sales 054881201579

Target table
Select the table for which the data filter will be created.
Choose tables ▼ Load more
orders 054881201579

Column-level access
Choose whether this filter should have column-level restrictions.
☐ Access to all columns
Filter won't have any column restrictions.
☐ Include columns
Filter will only allow access to specific columns.
☒ Exclude columns
Filter will allow access to all but specific columns.

Select columns
Choose one or more columns
▼
customer_name string

Row filter expression
Enter the rest of the following query statement "SELECT * FROM orders WHERE..."
Please see the documentation for examples of filter expressions.
product_type='pharma'

Cancel Create filter

SageMaker AI Security

Principle of Least Privilege

- Grant only the permissions required to perform a task
- Start with broad permissions while developing
- But lock it down once you have a better idea of the exact services and operations a workload requires
- Can use IAM Access Analyzer to generate least-privilege policies based on access activity

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListBucket",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3::my-specific-bucket",
      "Condition": {
        "StringLike": {
          "s3:prefix": "data/reports/*"
        }
      }
    },
    {
      "Sid": "AllowReadCSVFiles",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3::my-specific-bucket/data/reports/*.csv"
    }
  ]
}
```

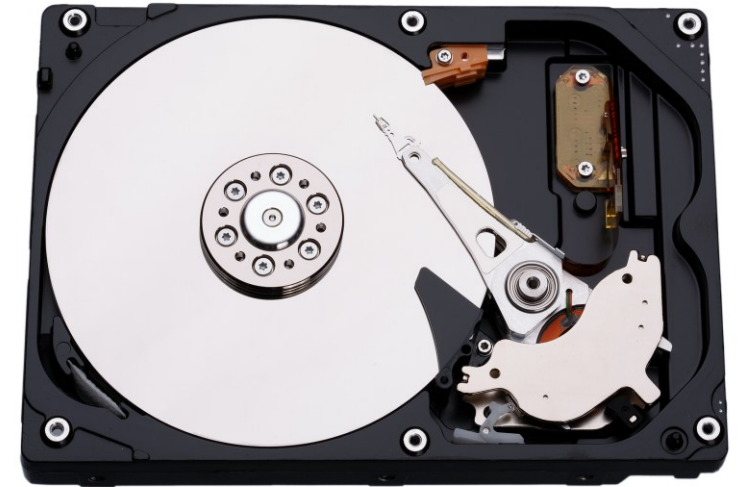
General AWS Security

- Use Identity and Access Management (IAM)
 - Set up user accounts with only the permissions they need
- Use MFA
- Use SSL/TLS when connecting to anything
- Use CloudTrail to log API and user activity
- Use encryption
- Be careful with PII



Protecting your Data at Rest in SageMaker AI

- AWS Key Management Service (KMS)
 - Accepted by notebooks and all SageMaker jobs
 - Training, tuning, batch transform, endpoints
 - Notebooks and everything under /opt/ml/ and /tmp can be encrypted with a KMS key
- S3
 - Can use encrypted S3 buckets for training data and hosting models
 - S3 can also use KMS



Protecting Data in Transit in SageMaker AI

- All traffic supports TLS / SSL
- IAM roles are assigned to SageMaker to give it permissions to access resources
- Inter-node training communication may be optionally encrypted
 - Can increase training time and cost with deep learning
 - AKA inter-container traffic encryption
 - Enabled via console or API when setting up a training or tuning job



SageMaker AI + VPC

- Training jobs run in a Virtual Private Cloud (VPC)
- You can use a private VPC for even more security
 - You'll need to set up S3 VPC endpoints
 - Custom endpoint policies and S3 bucket policies can keep this secure
- Notebooks are Internet-enabled by default
 - This can be a security hole
 - **If disabled, your VPC needs an interface endpoint (PrivateLink) or NAT Gateway, and allow outbound connections, for training and hosting to work**
- Training and Inference Containers are also Internet-enabled by default
 - Network isolation is an option, but this also prevents S3 access

SageMaker AI + IAM

- User permissions for:
 - CreateTrainingJob
 - CreateModel
 - CreateEndpointConfig
 - CreateTransformJob
 - CreateHyperParameterTuningJob
 - CreateNotebookInstance
 - UpdateNotebookInstance
- Predefined policies:
 - AmazonSageMakerReadOnly
 - AmazonSageMakerFullAccess
 - AdministratorAccess
 - DataScientist

SageMaker AI Logging and Monitoring

- CloudWatch can log, monitor and alarm on:
 - Invocations and latency of endpoints
 - Health of instance nodes (CPU, memory, etc)
- CloudTrail records actions from users, roles, and services within SageMaker
 - Log files delivered to S3 for auditing

IAM: Users & Groups



- IAM = Identity and Access Management, **Global** service
- **Root account** created by default, shouldn't be used or shared
- **Users** are people within your organization, and can be grouped
- **Groups** only contain users, not other groups
- Users don't have to belong to a group, and user can belong to multiple groups



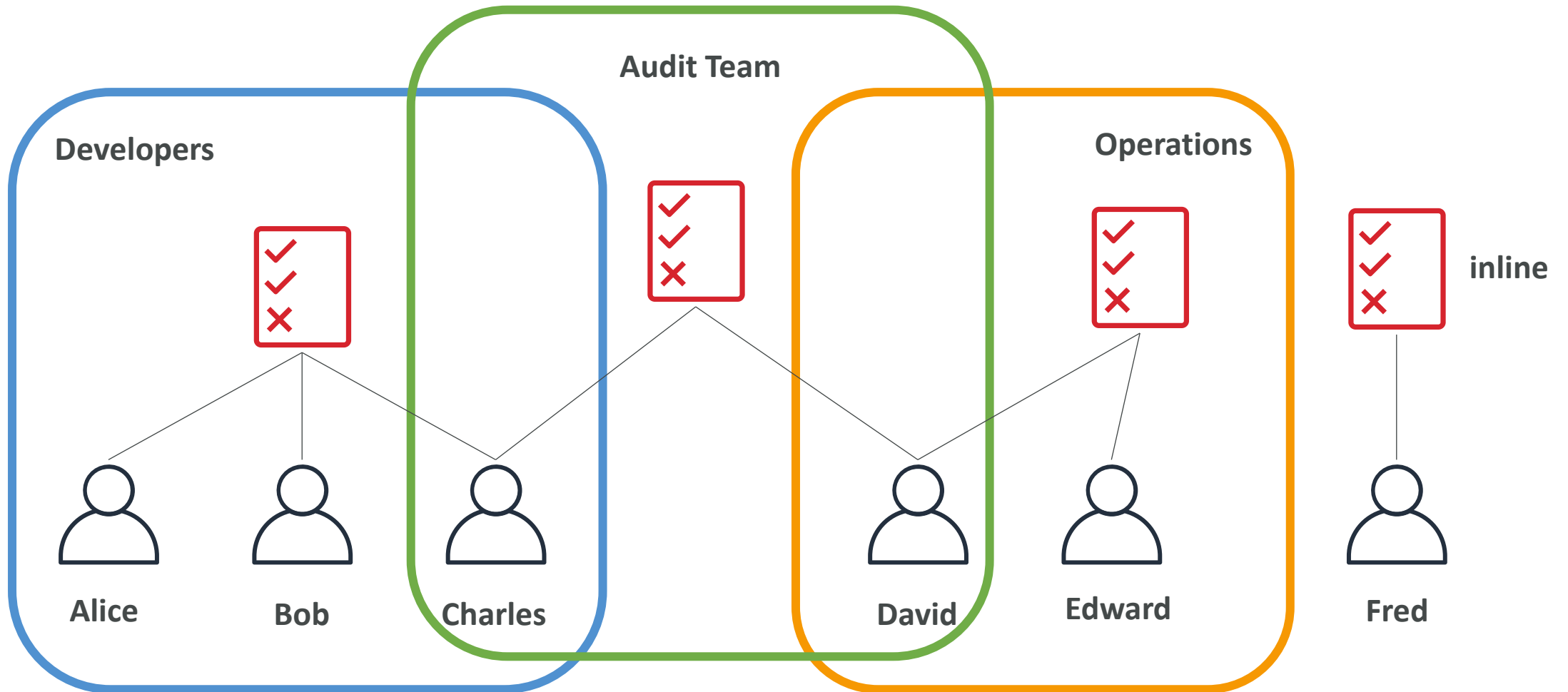
IAM: Permissions

- **Users or Groups** can be assigned JSON documents called policies
- These policies define the **permissions** of the users
- In AWS you apply the **least privilege principle**: don't give more permissions than a user needs

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "elasticloadbalancing:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```



IAM Policies inheritance



IAM Policies Structure

- Consists of
 - **Version:** policy language version, always include "2012-10-17"
 - **Id:** an identifier for the policy (optional)
 - **Statement:** one or more individual statements (required)
- Statements consists of
 - **Sid:** an identifier for the statement (optional)
 - **Effect:** whether the statement allows or denies access (Allow, Deny)
 - **Principal:** account/user/role to which this policy applied to
 - **Action:** list of actions this policy allows or denies
 - **Resource:** list of resources to which the actions applied to
 - **Condition:** conditions for when this policy is in effect (optional)

```
{
  "Version": "2012-10-17",
  "Id": "S3-Account-Permissions",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::123456789012:root"]
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::mybucket/*"]
    }
  ]
}
```

IAM – Password Policy

- Strong passwords = higher security for your account
- In AWS, you can setup a password policy:
 - Set a minimum password length
 - Require specific character types:
 - including uppercase letters
 - lowercase letters
 - numbers
 - non-alphanumeric characters
 - Allow all IAM users to change their own passwords
 - Require users to change their password after some time (password expiration)
 - Prevent password re-use

Multi Factor Authentication - MFA



- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- **You want to protect your Root Accounts and IAM users**
- MFA = password *you know* + security device *you own*



Alice

Password

+



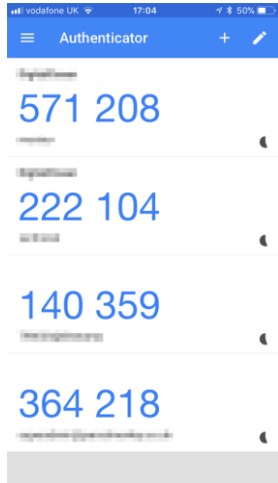
=>

Successful login

- **Main benefit of MFA:**
if a password is stolen or hacked, the account is not compromised

MFA devices options in AWS

Virtual MFA device



Google Authenticator
(phone only)

Support for multiple tokens on a single device.



Authy
(phone only)

Universal 2nd Factor (U2F) Security Key



YubiKey by Yubico (3rd party)

Support for multiple root and IAM users
using a single security key

MFA devices options in AWS

Hardware Key Fob MFA Device



Provided by Gemalto (3rd party)

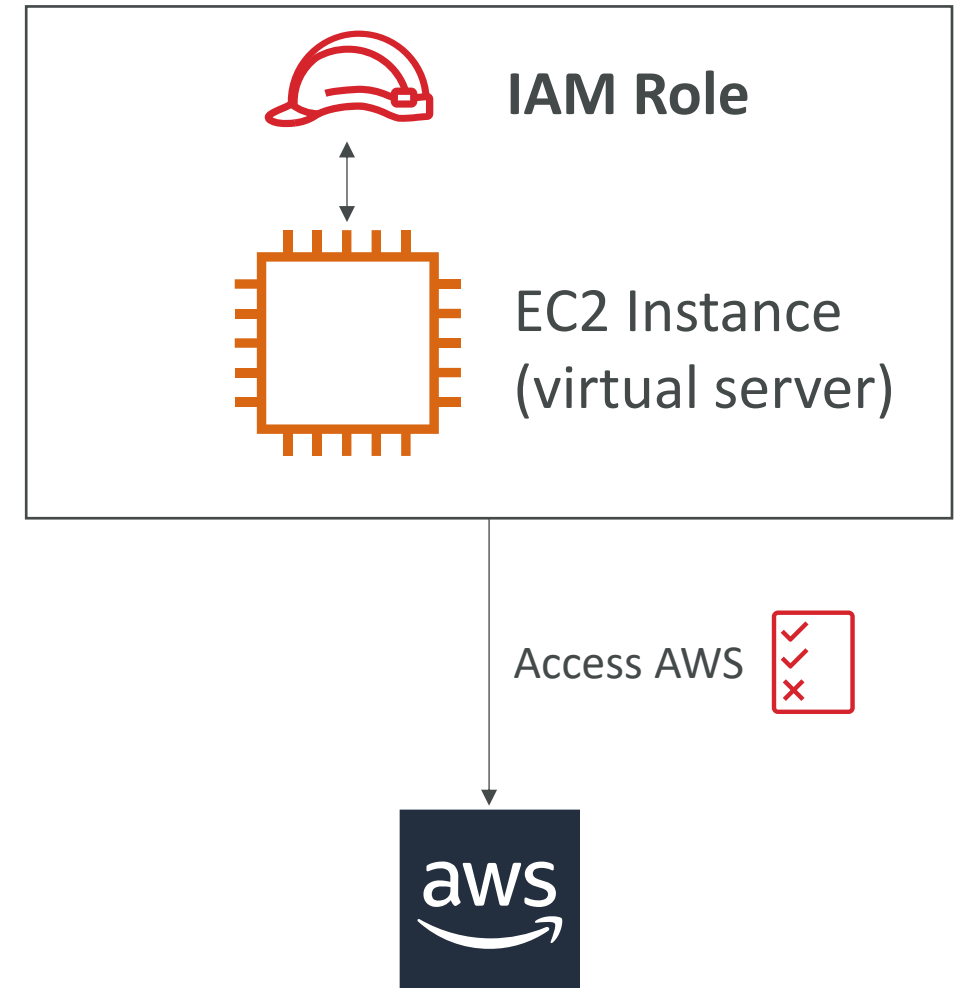
Hardware Key Fob MFA Device for AWS GovCloud (US)



Provided by SurePassID (3rd party)

IAM Roles for Services

- Some AWS service will need to perform actions on your behalf
- To do so, we will assign **permissions** to AWS services with **IAM Roles**
- Common roles:
 - EC2 Instance Roles
 - Lambda Function Roles
 - Roles for CloudFormation



Why encryption?

Encryption in flight (TLS / SSL)

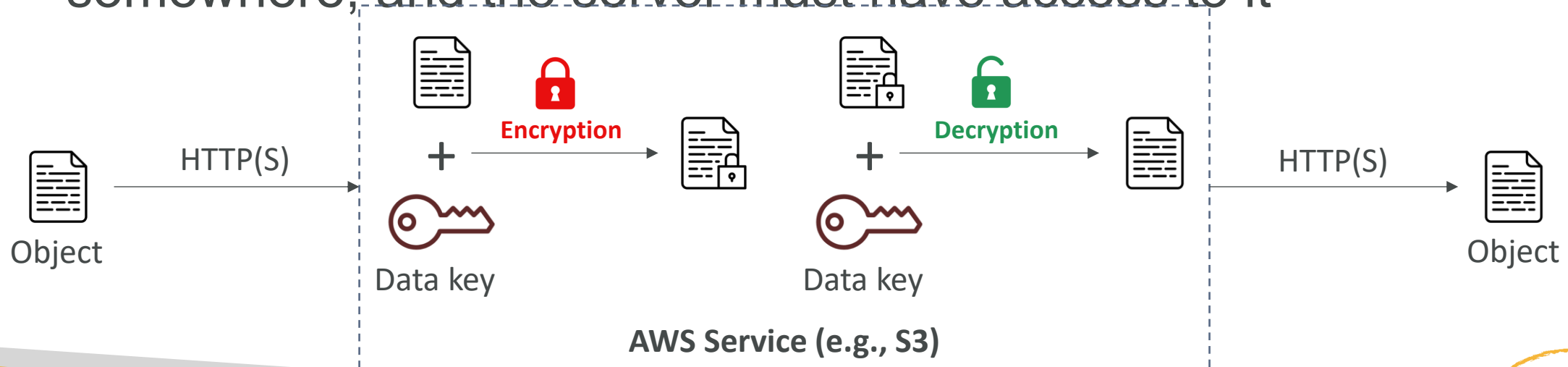
- Data is encrypted before sending and decrypted after receiving
- TLS certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen



Why encryption?

Server-side encryption at rest

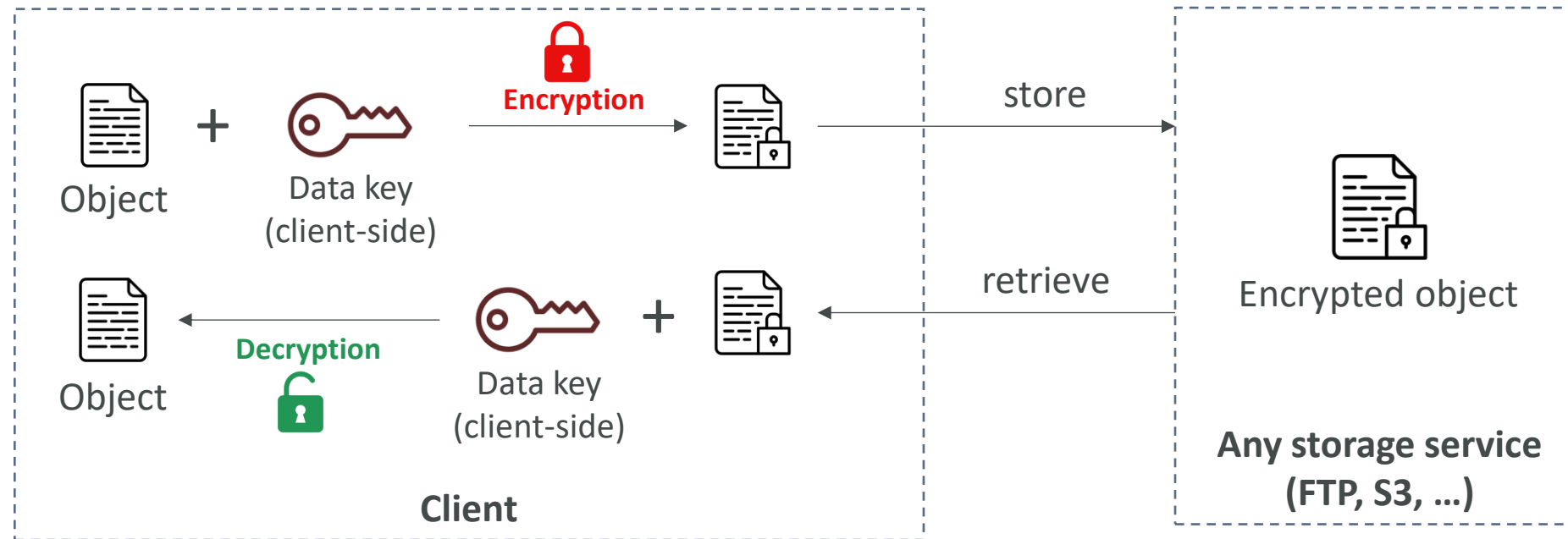
- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere, and the server must have access to it



Why encryption?

Client-side encryption

- Data is encrypted by the client and never decrypted by the server
- Data will be decrypted by a receiving client
- The server should not be able to decrypt the data
- Could leverage Envelope Encryption



AWS KMS (Key Management Service)



- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- AWS manages encryption keys for us
- Fully integrated with IAM for authorization
- Easy way to control access to your data
- Able to audit KMS Key usage using CloudTrail
- Seamlessly integrated into most AWS services (EBS, S3, RDS, SSM...)
- **Never ever store your secrets in plaintext, especially in your code!**

• KMS Key Encryption also available through API calls (SDK, CLI)

• Encrypted secrets can be stored in the code / environment variables

KMS Keys Types

- **KMS Keys** is the new name of KMS Customer *Master Key*
- **Symmetric (AES-256 keys)**
 - Single encryption key that is used to Encrypt and Decrypt
 - AWS services that are integrated with KMS use Symmetric CMKs
 - You never get access to the KMS Key unencrypted (must call KMS API to use)
- **Asymmetric (RSA & ECC key pairs)**
 - Public (Encrypt) and Private Key (Decrypt) pair
 - Used for Encrypt/Decrypt, or Sign/Verify operations
 - The public key is downloadable, but you can't access the Private Key unencrypted
 - Use case: encryption outside of AWS by users who can't call the KMS

AWS KMS (Key Management Service)

• Types of KMS Keys:

- AWS Owned Keys (free): SSE-S3, SSE-SQS, SSE-DDB (default key)
- AWS Managed Key: **free** (aws/service-name, example: aws/rds or aws/ebs)
- Customer managed keys created in KMS: **\$1 / month**
- Customer managed keys imported: **\$1 / month**
- + pay for API call to KMS (\$0.03 / 10000 calls)

Encryption key management

☒ Owned by Amazon DynamoDB

☐ AWS managed key [Lea](#)

Key alias: aws/dynamodb.

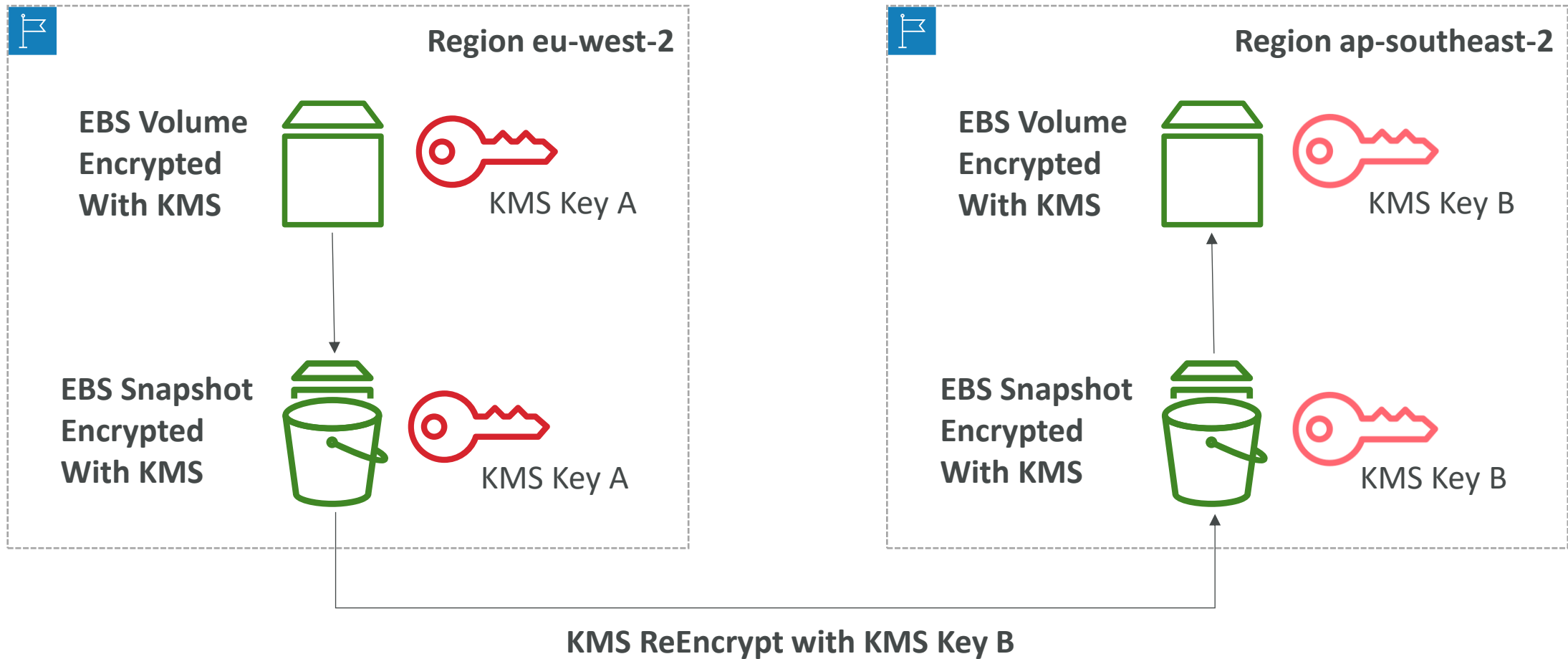
☐ Stored in your account,
and owned and managed by you

• Automatic Key rotation:

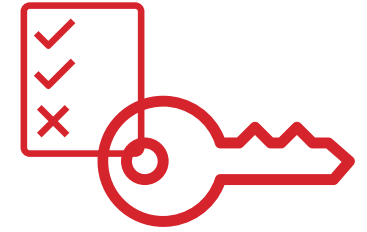
- AWS-managed KMS Key: automatic every 1 year
- Customer-managed KMS Key: (must be enabled) automatic & on-demand

• Imported KMS Key: only manual rotation possible using alias

Copying Snapshots across regions



KMS Key Policies



- Control access to KMS keys, “similar” to S3 bucket policies
- Difference: you cannot control access without them
- **Default KMS Key Policy:**
 - Created if you don’t provide a specific KMS Key Policy
 - Complete access to the key to the root user = entire AWS account
- **Custom KMS Key Policy:**
 - Define users, roles that can access the KMS key
 - Define who can administer the key
 - Useful for cross-account access of your KMS key

Copying Snapshots across accounts

1. Create a Snapshot, encrypted with your own KMS Key (Customer Managed Key)
2. **Attach a KMS Key Policy to authorize cross-account access**
3. Share the encrypted snapshot
4. (in target) Create a copy of the Snapshot, encrypt it with a CMK in your account
5. Create a volume from the snapshot

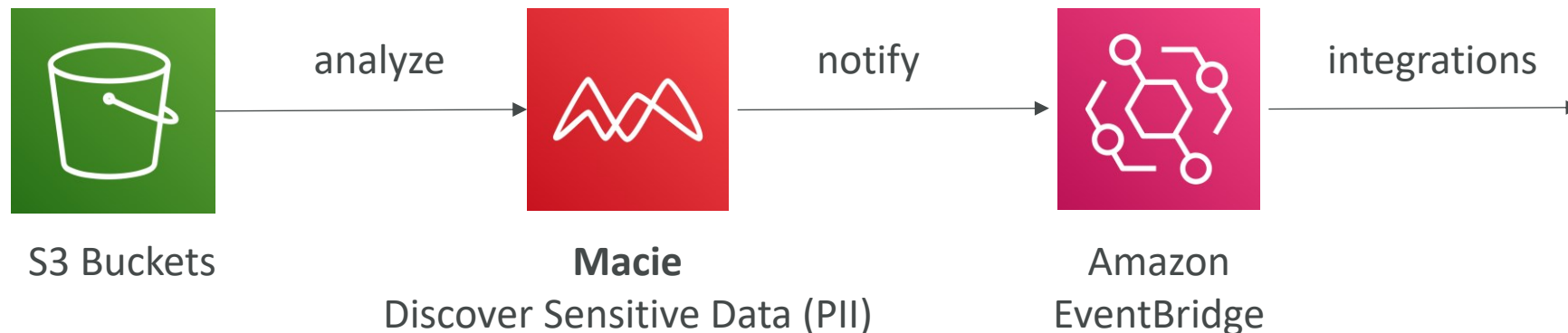
```
{  
  "Sid": "Allow use of the key with destination account",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::TARGET-ACCOUNT-ID:role/ROLENAME"  
  },  
  "Action": [  
    "kms:Decrypt",  
    "kms:CreateGrant"  
  ],  
  "Resource": "*",  
  "Condition": {  
    "StringEquals": {  
      "kms:ViaService": "ec2.REGION.amazonaws.com",  
      "kms:CallerAccount": "TARGET-ACCOUNT-ID"  
    }  
  }  
}
```

KMS Key Policy

AWS Macie

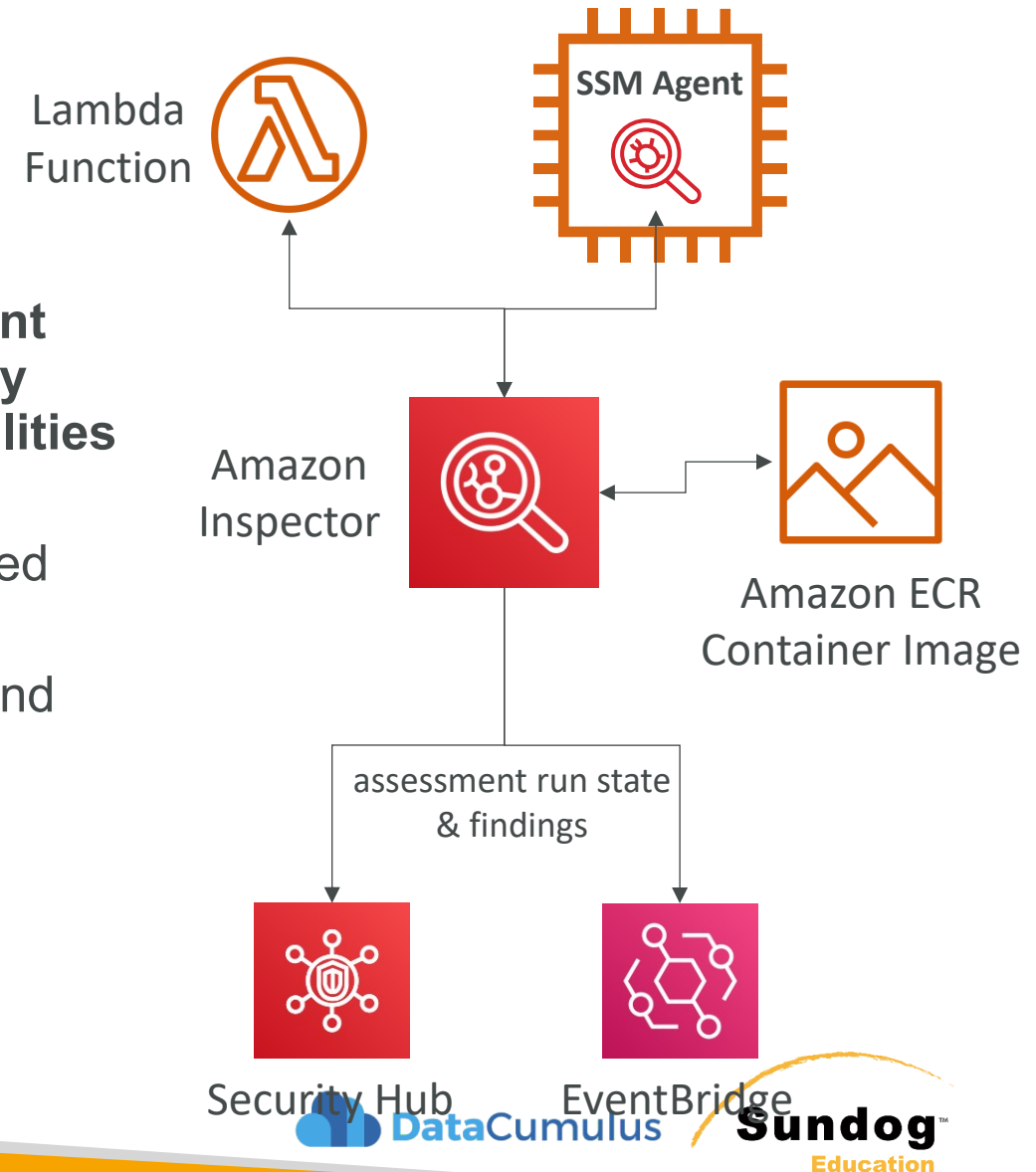


- Amazon Macie is a fully managed data security and data privacy service that uses **machine learning and pattern matching to discover and protect your sensitive data in AWS.**
- Macie helps identify and alert you to **sensitive data, such as personally identifiable information (PII)**



Amazon Inspector

- **Automated Security Assessments**
- **For EC2 instances**
 - Leveraging the **AWS System Manager (SSM) agent**
 - Analyze against **unintended network accessibility**
 - Analyze the **running OS** against **known vulnerabilities**
- **For Container Images push to Amazon ECR**
 - Assessment of Container Images as they are pushed
- **For Lambda Functions**
 - Identifies software vulnerabilities in function code and package dependencies
 - Assessment of functions as they are deployed
- Reporting & integration with AWS Security Hub
- Send findings to Amazon Event Bridge



What does Amazon Inspector evaluate



- **Remember: only for EC2 instances, Container Images & Lambda functions**
- Continuous scanning of the infrastructure, only when needed
- Package vulnerabilities (EC2, ECR & Lambda) – database of CVE
- Network reachability (EC2)
- A risk score is associated with all vulnerabilities for prioritization

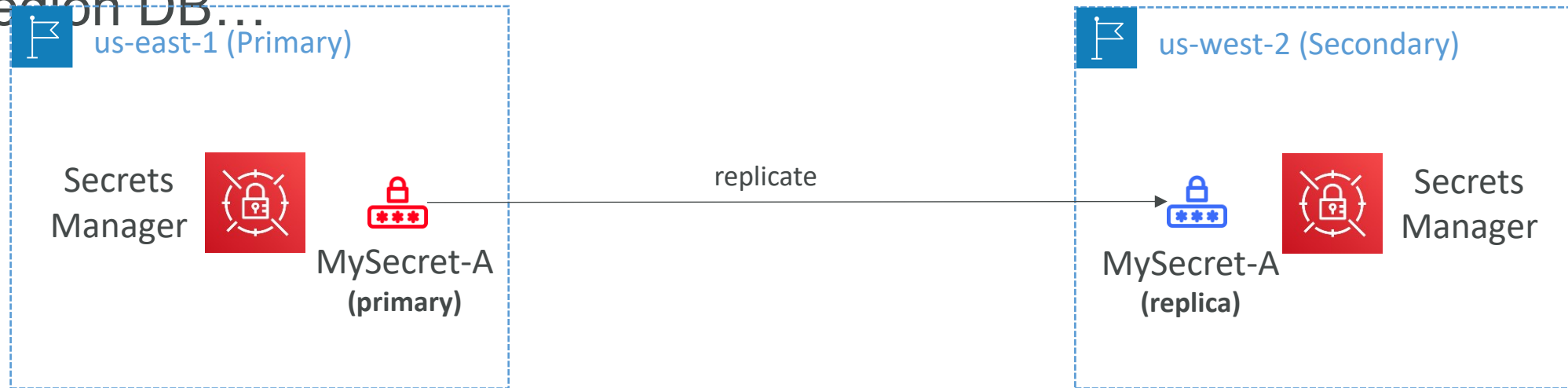
AWS Secrets Manager



- Newer service, meant for storing secrets
 - Capability to force **rotation of secrets** every X days
 - Automate generation of secrets on rotation (uses Lambda)
 - Integration with **Amazon RDS** (MySQL, PostgreSQL, Aurora)
 - Secrets are encrypted using KMS
-
- Mostly meant for RDS integration

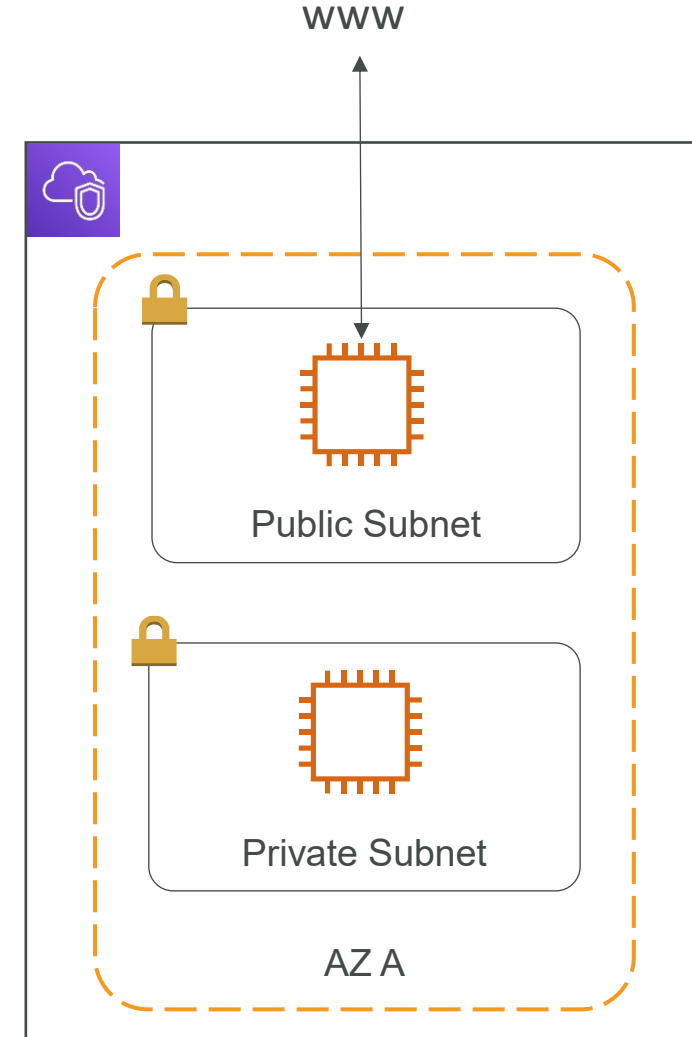
AWS Secrets Manager – Multi-Region Secrets

- Replicate Secrets across multiple AWS Regions
- Secrets Manager keeps read replicas in sync with the primary Secret
- Ability to promote a read replica Secret to a standalone Secret
- Use cases: multi-region apps, disaster recovery strategies, multi-region DB...

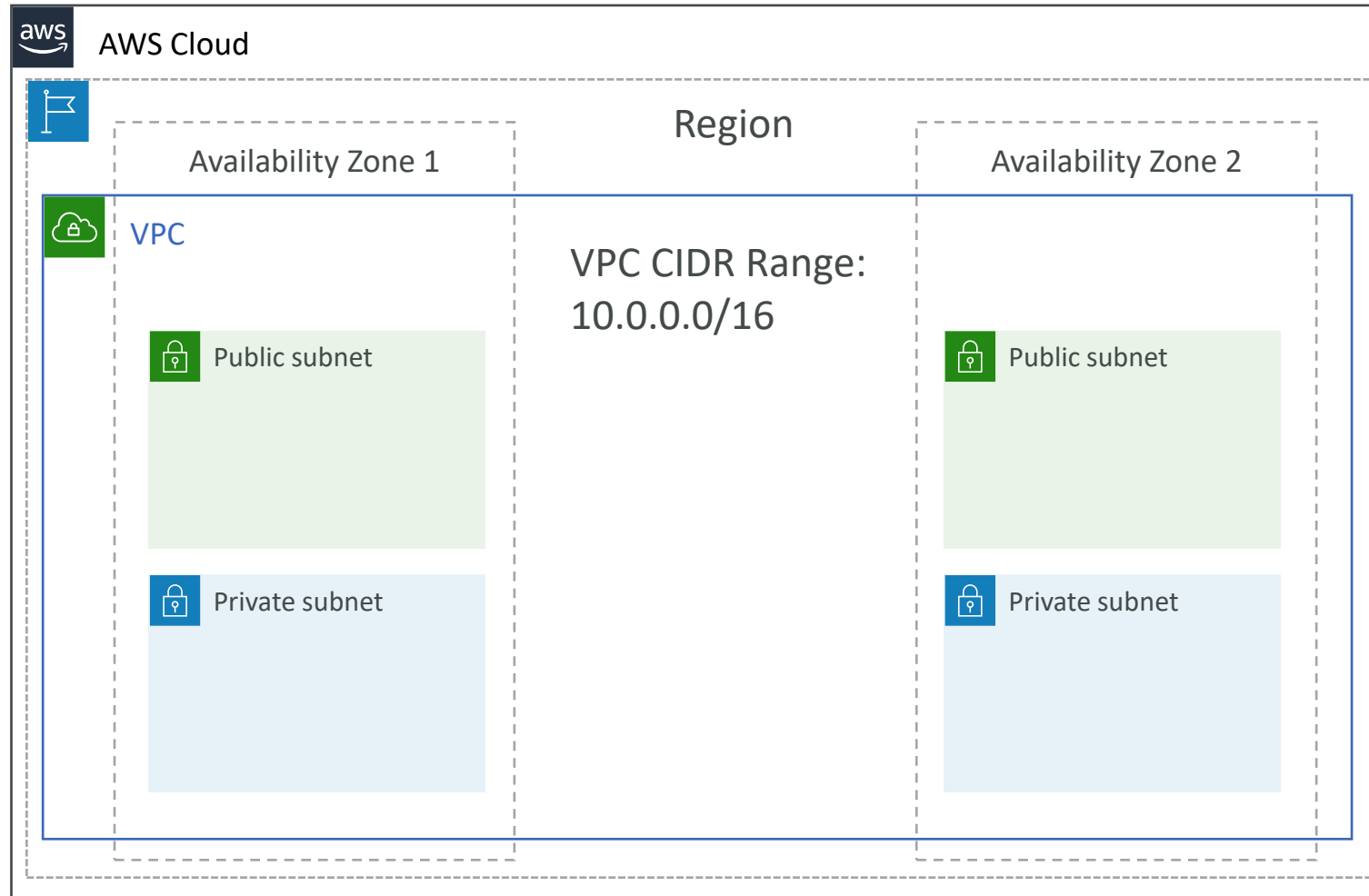


VPC & Subnets Primer

- **VPC**: private network to deploy your resources (regional resource)
- **Subnets** allow you to partition your network inside your VPC (Availability Zone resource)
- A **public subnet** is a subnet that is accessible from the internet
- A **private subnet** is a subnet that is not accessible from the internet
- To define access to the internet and between subnets, we use **Route Tables**.

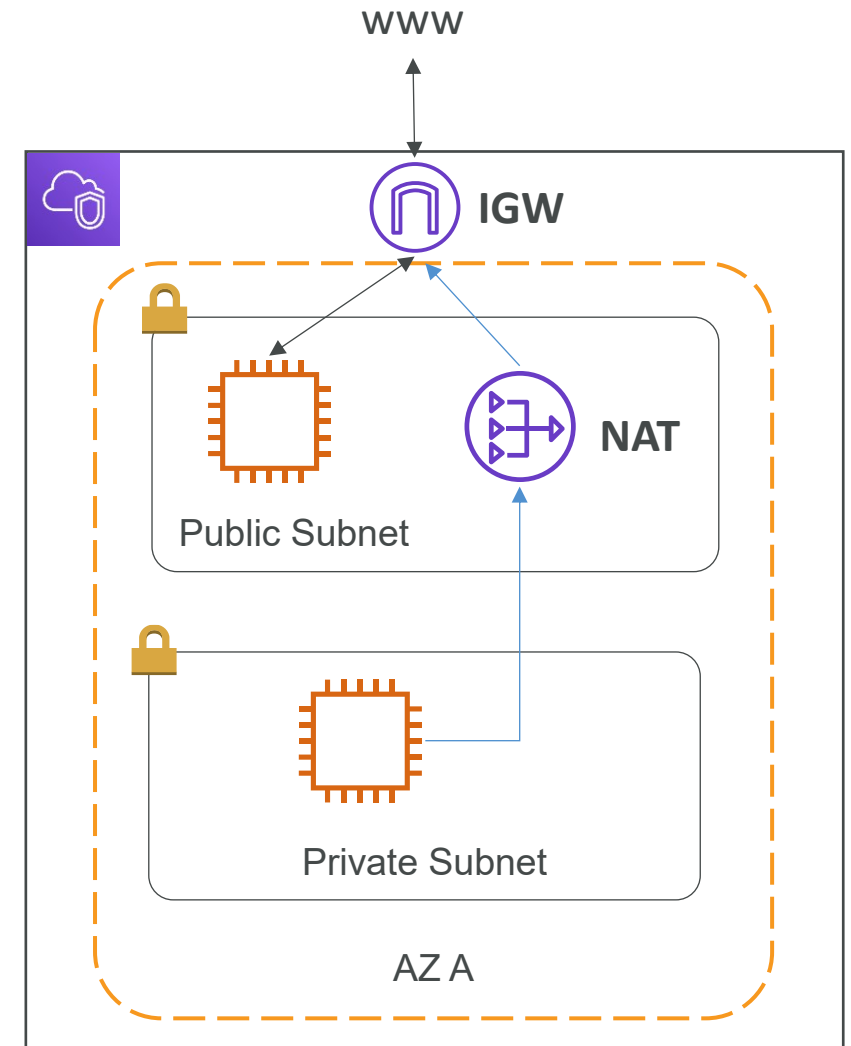


VPC Diagram



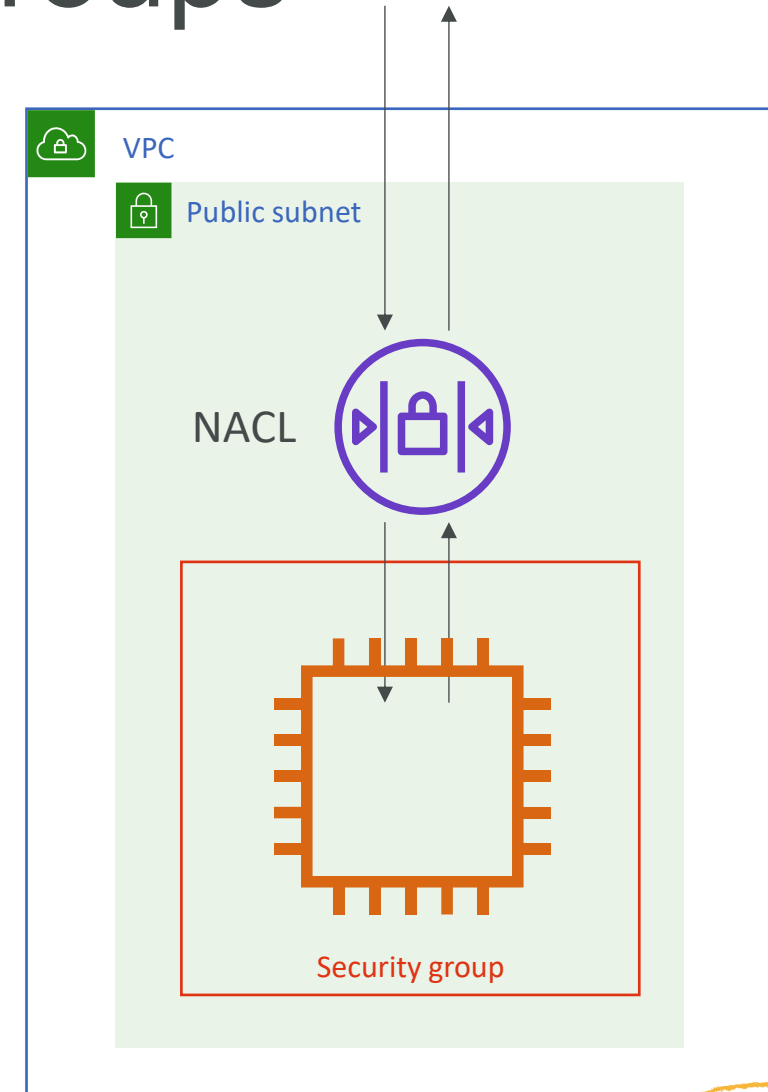
Internet Gateway & NAT Gateways

- **Internet Gateways** helps our VPC instances connect with the internet
- Public Subnets have a route to the internet gateway.
- **NAT Gateways** (AWS-managed) & **NAT Instances** (self-managed) allow your instances in your **Private Subnets** to access the internet while remaining private



Network ACL & Security Groups

- **NACL** (Network ACL)
 - A firewall which controls traffic from and to subnet
 - Can have ALLOW and DENY rules
 - Are attached at the **Subnet** level
 - Rules only include IP addresses
- **Security Groups**
 - A firewall that controls traffic to and from an **ENI / an EC2 Instance**
 - Can have only ALLOW rules
 - Rules include IP addresses and other security groups

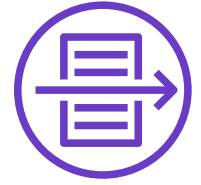


Network ACLs vs Security Groups

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, you don't have to rely on users to specify the security group)

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html#VPC_Security_Comparison

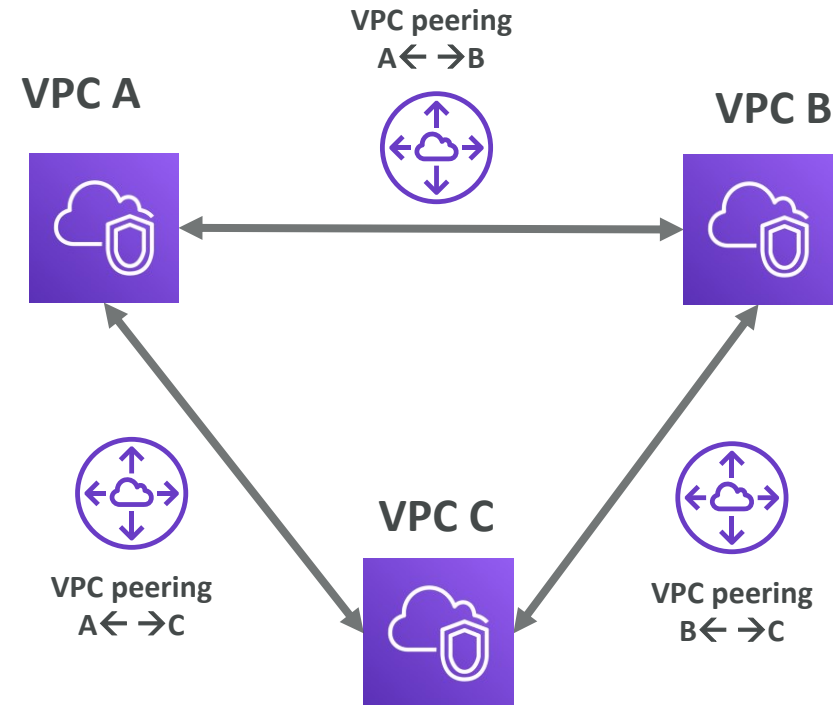
VPC Flow Logs



- Capture information about IP traffic going into your interfaces:
 - **VPC** Flow Logs
 - **Subnet** Flow Logs
 - **Elastic Network Interface** Flow Logs
- Helps to monitor & troubleshoot connectivity issues. Example:
 - Subnets to internet
 - Subnets to subnets
 - Internet to subnets
- Captures network information from AWS managed interfaces too: Elastic Load Balancers, ElastiCache, RDS, Aurora, etc...
- VPC Flow logs data can go to S3, CloudWatch Logs, and Kinesis Data Firehose

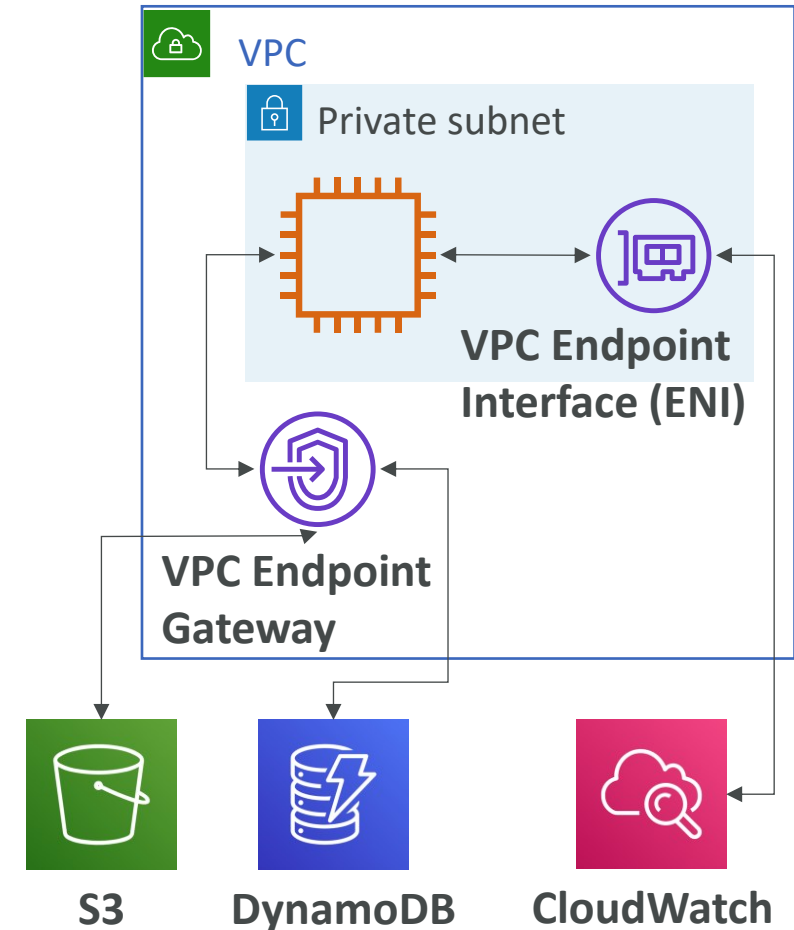
VPC Peering

- Connect two VPC, privately using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDR (IP address range)
- VPC Peering connection is **not transitive** (must be established for each VPC that need to communicate with one another)



VPC Endpoints

- Endpoints allow you to connect to AWS Services **using a private network** instead of the public www network
- This gives you enhanced security and lower latency to access AWS services
- VPC Endpoint Gateway: S3 & DynamoDB
- VPC Endpoint Interface: most services (including S3 & DynamoDB)
- Only used within your VPC



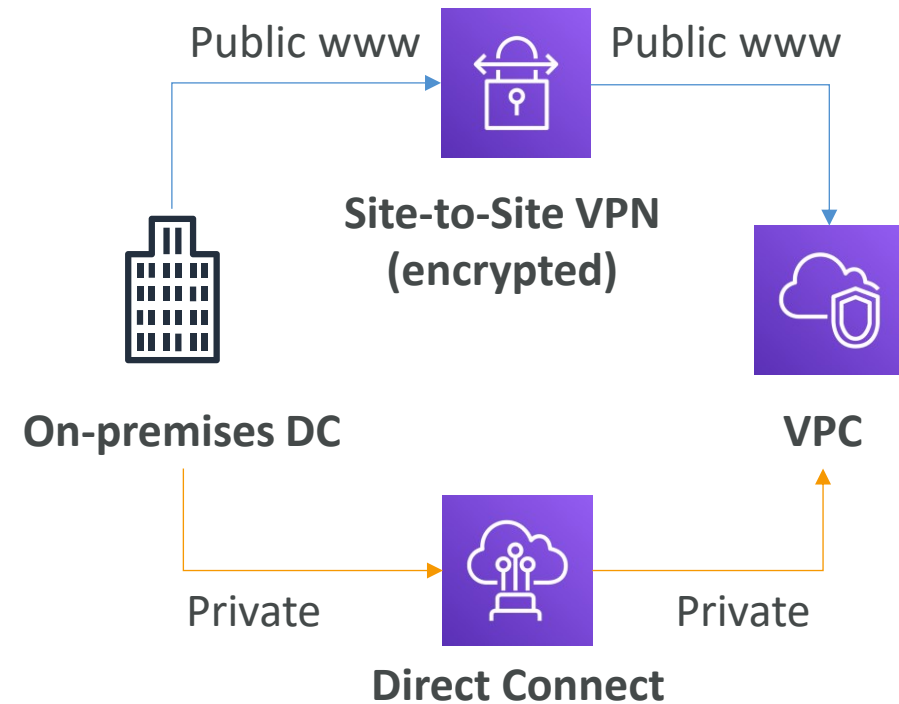
Site to Site VPN & Direct Connect

- **Site to Site VPN**

- Connect an on-premises VPN to AWS
- The connection is automatically encrypted
- Goes over the public internet

- **Direct Connect (DX)**

- Establish a physical connection between on-premises and AWS
- The connection is private, secure and fast
- Goes over a private network
- Takes at least a month to establish



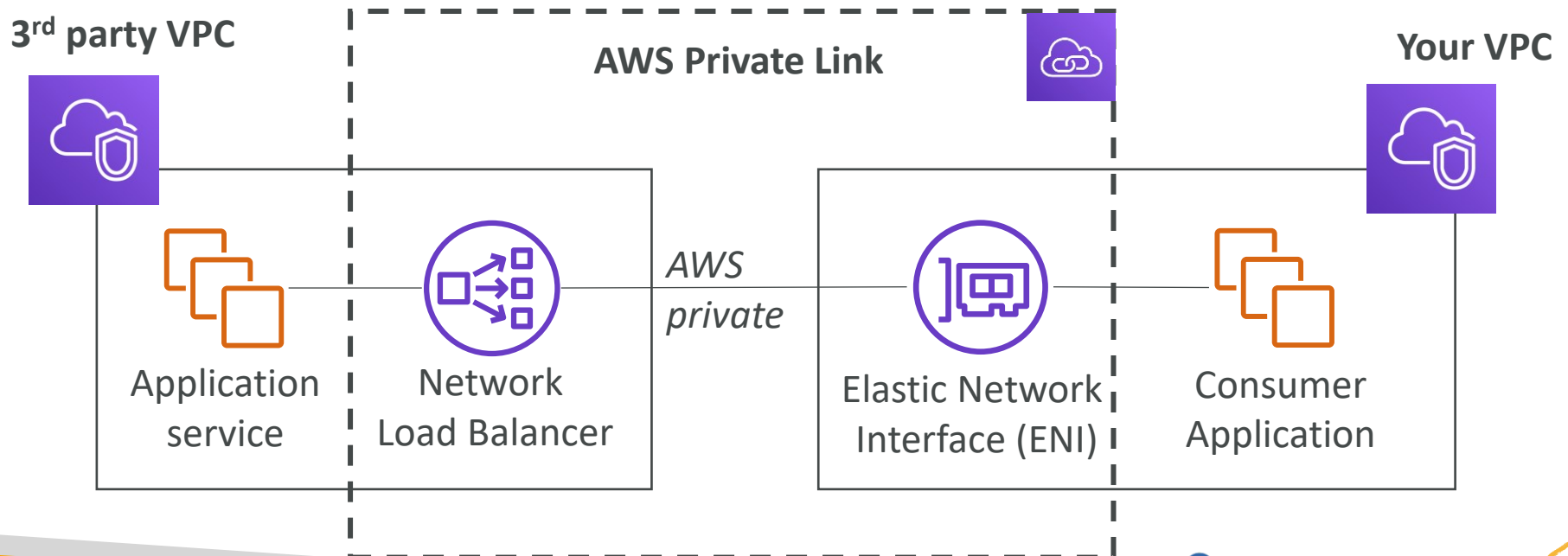
VPC Closing Comments

- **VPC:** Virtual Private Cloud
- **Subnets:** Tied to an AZ, network partition of the VPC
- **Internet Gateway:** at the VPC level, provide Internet Access
- **NAT Gateway / Instances:** give internet access to private subnets
- **NACL:** Stateless, subnet rules for inbound and outbound
- **Security Groups:** Stateful, operate at the EC2 instance level or ENI
- **VPC Peering:** Connect two VPC with non overlapping IP ranges, non transitive
- **VPC Endpoints:** Provide private access to AWS Services within VPC
- **VPC Flow Logs:** network traffic logs
- **Site to Site VPN:** VPN over public internet between on-premises DC and AWS
- **Direct Connect:** direct private connection to a AWS

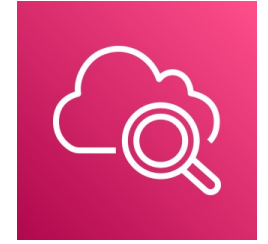
AWS PrivateLink (VPC Endpoint Services)



- Most secure & scalable way to expose a service to 1000s of VPCs
- Does not require VPC peering, internet gateway, NAT, route tables...
- Requires a network load balancer (Service VPC) and ENI (Customer VPC)



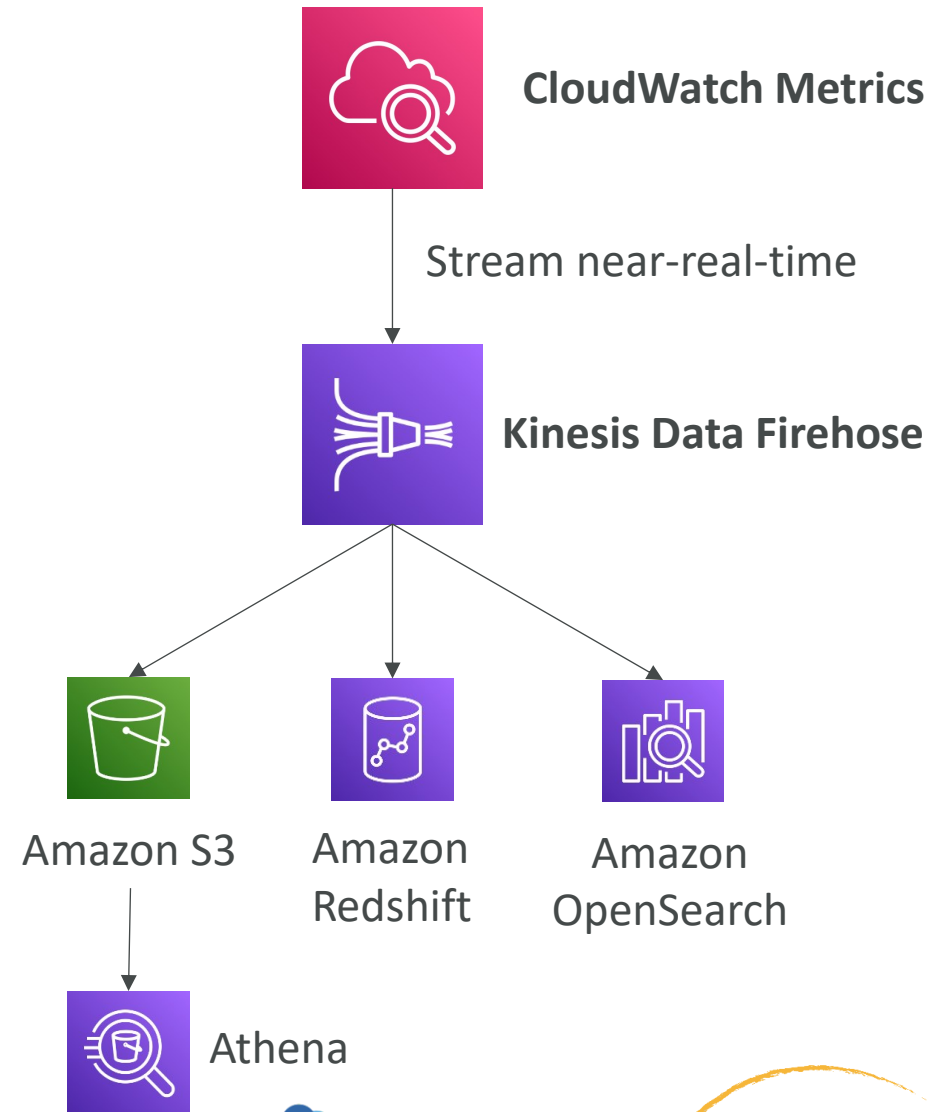
Amazon CloudWatch Metrics



- CloudWatch provides metrics for *every* service in AWS
- **Metric** is a variable to monitor (CPUUtilization, NetworkIn...)
- Metrics belong to **namespaces**
- **Dimension** is an attribute of a metric (instance id, environment, etc...).
- Up to 30 dimensions per metric
- Metrics have **timestamps**
- Can create CloudWatch dashboards of metrics
- Can create **CloudWatch Custom Metrics** (for the RAM for example)

CloudWatch Metric Streams

- Continually stream CloudWatch metrics to a destination of your choice, with **near-real-time delivery** and low latency.
 - Amazon Kinesis Data Firehose (and then its destinations)
 - 3rd party service provider: Datadog, Dynatrace, New Relic, Splunk, Sumo Logic...
- Option to **filter metrics** to only stream a subset of them



CloudWatch Logs



- **Log groups:** arbitrary name, usually representing an application
- **Log stream:** instances within application / log files / containers
- Can define log expiration policies (never expire, 1 day to 10 years...)
- **CloudWatch Logs can send logs to:**
 - Amazon S3 (exports)
 - Kinesis Data Streams
 - Kinesis Data Firehose
 - AWS Lambda
 - OpenSearch
- Logs are encrypted by default
- Can setup KMS-based encryption with your own keys

CloudWatch Logs - Sources

- SDK, CloudWatch Logs Agent, CloudWatch Unified Agent
- Elastic Beanstalk: collection of logs from application
- ECS: collection from containers
- AWS Lambda: collection from function logs
- VPC Flow Logs: VPC specific logs
- API Gateway
- CloudTrail based on filter
- Route53: Log DNS queries

CloudWatch Logs Insights

The screenshot shows the AWS CloudWatch Logs Insights interface. Annotations with orange boxes and arrows point to various features:

- Change the time range here.** Points to the time range selector showing "2021-11-09 (06:40:02) > 2021-11-09 (06:55:17)".
- Discovered Fields in your log groups.** Points to the "Fields" link in the right-hand navigation menu.
- Write your query here.** Points to the query editor containing:

```
1 fields @timestamp, @message
2 | .sort @timestamp desc
3 | .limit 20
```
- Export the results, or add to a dashboard.** Points to the "Export results" and "Add to dashboard" buttons.
- Tabs for query results, and visualization options.** Points to the "Logs" and "Visualization" tabs.

The interface also includes a "Select log group(s)" dropdown, a "Run query" button, and a "History" button. Below the query editor, it states "Queries are allowed to run for up to 15 minutes." The results section shows a histogram and a table of log records.

#	@timestamp	@message
▶ 1	2021-11-09T06:54:17.62...	{"Severity": "INFO", "message": "This is where the message detail would go", "IP Address": "10.30.86.98", "Timestamp": "2021-11-09T11:..."}
▶ 2	2021-11-09T06:54:13.38...	{"Severity": "INFO", "message": "This is where the message detail would go", "IP Address": "192.168.0.43", "Timestamp": "2021-11-09T11:..."}

<https://mng.workshop.aws/operations-2022/detect/cwlogs.html>

CloudWatch Logs Insights

- Search and analyze log data stored in CloudWatch Logs
- Example: find a specific IP inside a log, count occurrences of “ERROR” in your logs...
- Provides a purpose-built query language
 - Automatically discovers fields from AWS services and JSON log events
 - Fetch desired event fields, filter based on conditions, calculate aggregate statistics, sort events, limit number of events...
 - Can save queries and add them to CloudWatch Dashboards
- Can query multiple Log Groups in different AWS accounts
- It's a query engine, not a real-time engine

Sample queries [Learn more](#)

▶ Lambda

▶ VPC Flow Logs

▶ CloudTrail

▼ Common queries

▼ 25 most recently added log events

```
fields @timestamp, @message
| sort @timestamp desc
| limit 25
```

[Apply](#)

▼ Number of exceptions logged every 5 minutes

```
filter @message like /Exception/
| stats count(*) as exceptionCount by
  bin(5m)
| sort exceptionCount desc
```

[Apply](#)

▼ List of log events that are not exceptions

```
fields @message
| filter @message not like /Exception/
```

[Apply](#)

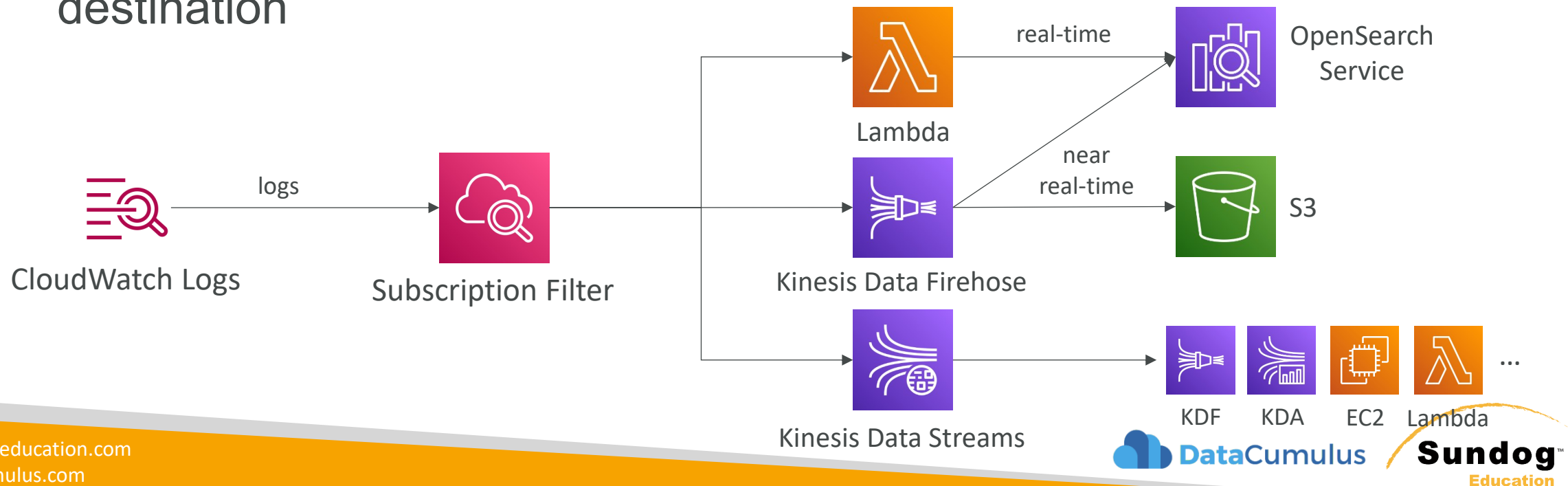
CloudWatch Logs – S3 Export



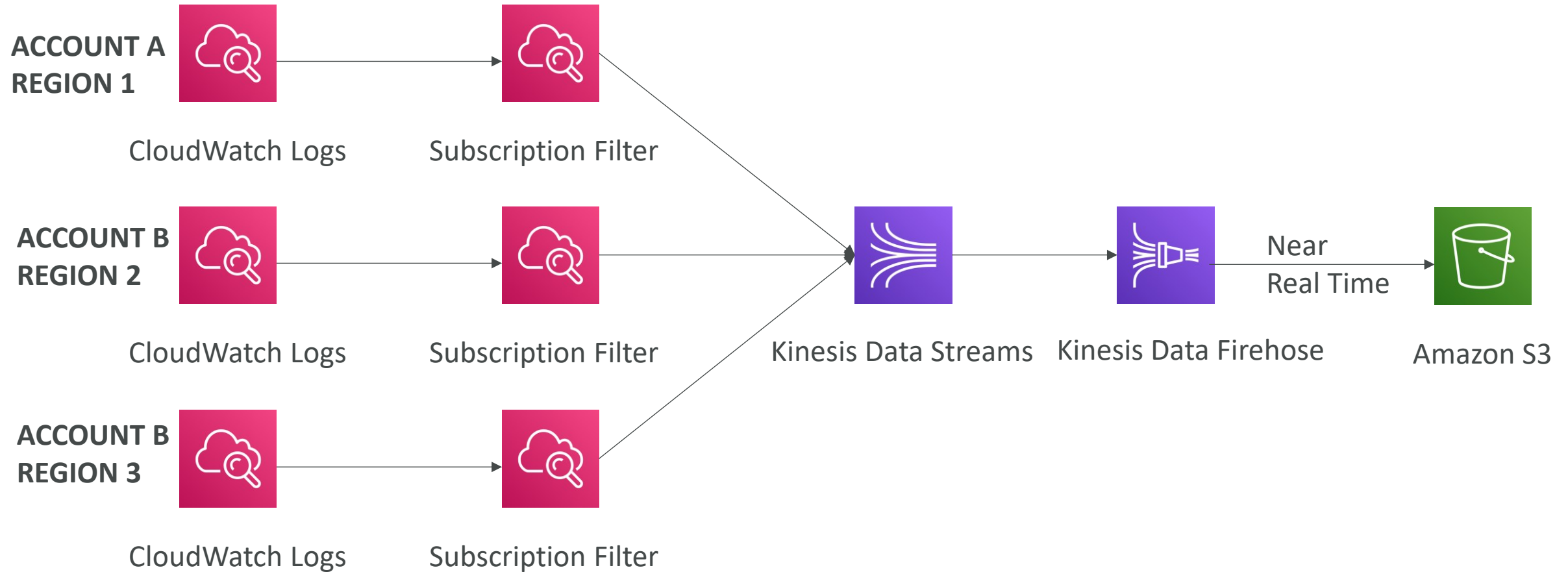
- Log data can take **up to 12 hours** to become available for export
- The API call is **CreateExportTask**
- Not near-real time or real-time... use Logs Subscriptions instead

CloudWatch Logs Subscriptions

- Get a real-time log events from CloudWatch Logs for processing and analysis
- Send to Kinesis Data Streams, Kinesis Data Firehose, or Lambda
- **Subscription Filter** – filter which logs are events delivered to your destination

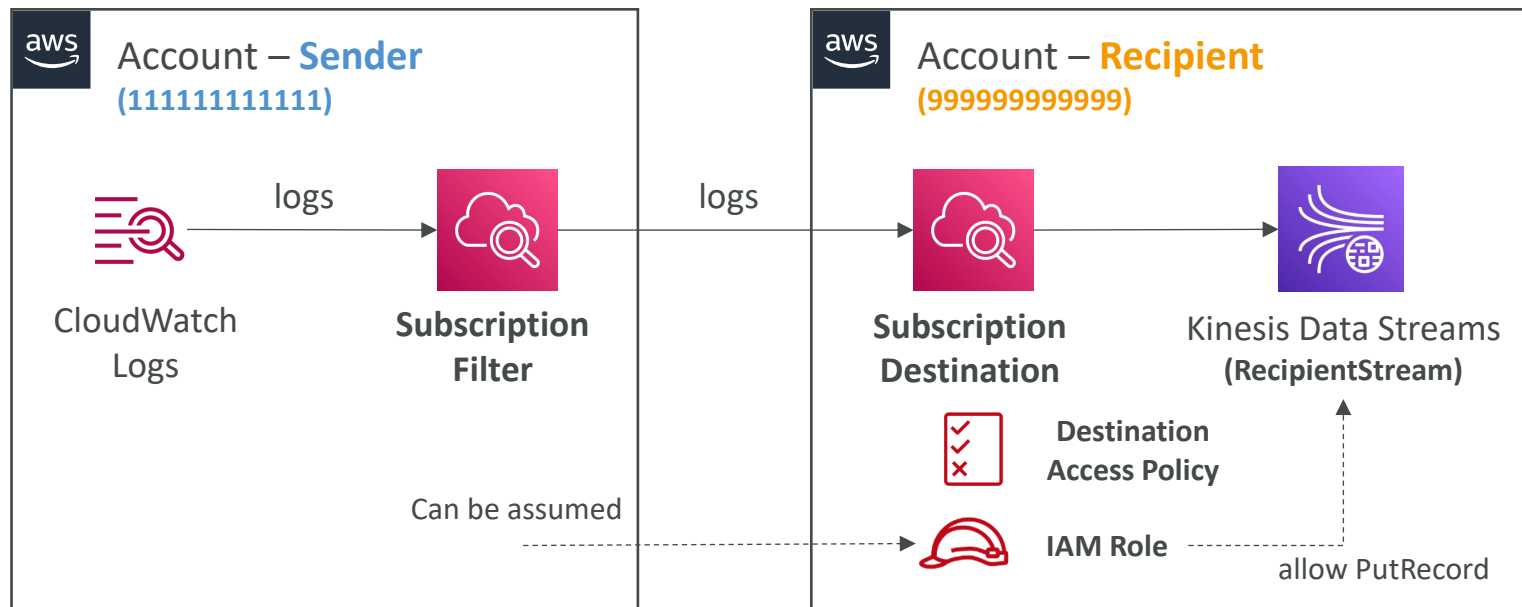


CloudWatch Logs Aggregation Multi-Account & Multi Region



CloudWatch Logs Subscriptions

- **Cross-Account Subscription** – send log events to resources in a different AWS account (KDS, KDF)



```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "kinesis:PutRecord",  
      "Resource": "arn:aws:kinesis:us-east-1:  
999999999999:stream/RecipientStream"  
    }  
  ]  
}
```

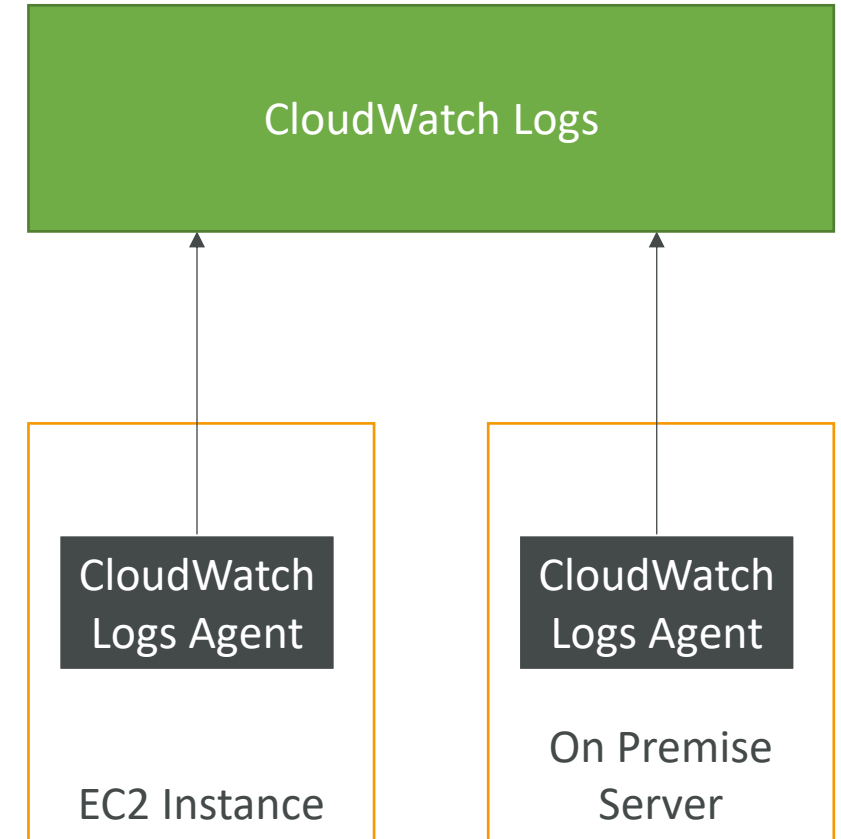
IAM Role
(Cross-Account)

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "111111111111"  
      },  
      "Action": "logs:PutSubscriptionFilter",  
      "Resource": "arn:aws:logs:us-east-1:999999999999:  
destination:testDestination"  
    }  
  ]  
}
```

Destination
Access Policy

CloudWatch Logs for EC2

- By default, no logs from your EC2 machine will go to CloudWatch
- You need to run a CloudWatch agent on EC2 to push the log files you want
- Make sure IAM permissions are correct
- The CloudWatch log agent can be setup on-premises too



CloudWatch Logs Agent & Unified Agent

- For virtual servers (EC2 instances, on-premises servers...)
- **CloudWatch Logs Agent**
 - Old version of the agent
 - Can only send to CloudWatch Logs
- **CloudWatch Unified Agent**
 - Collect additional system-level metrics such as RAM, processes, etc...
 - Collect logs to send to CloudWatch Logs
 - Centralized configuration using SSM Parameter Store

CloudWatch Unified Agent – Metrics

- Collected directly on your Linux server / EC2 instance
- **CPU** (active, guest, idle, system, user, steal)
- **Disk metrics** (free, used, total), Disk IO (writes, reads, bytes, iops)
- **RAM** (free, inactive, used, total, cached)
- **Netstat** (number of TCP and UDP connections, net packets, bytes)
- **Processes** (total, dead, bloqued, idle, running, sleep)
- **Swap Space** (free, used, used %)
- Reminder: out-of-the box metrics for EC2 – disk, CPU, network (high level)

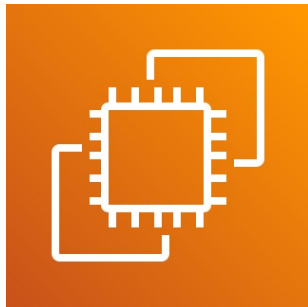
CloudWatch Alarms



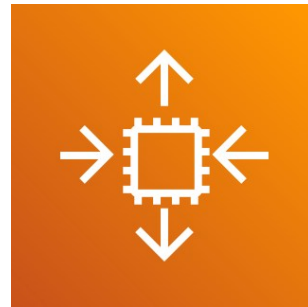
- Alarms are used to trigger notifications for any metric
- Various options (sampling, %, max, min, etc...)
- Alarm States:
 - OK
 - INSUFFICIENT_DATA
 - ALARM
- Period:
 - Length of time in seconds to evaluate the metric
 - High resolution custom metrics: 10 sec, 30 sec or multiples of 60 sec

CloudWatch Alarm Targets

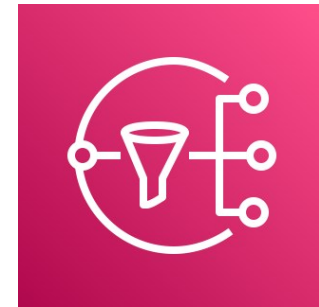
- Stop, Terminate, Reboot, or Recover an EC2 Instance
- Trigger Auto Scaling Action
- Send notification to SNS (from which you can do pretty much anything)



Amazon EC2



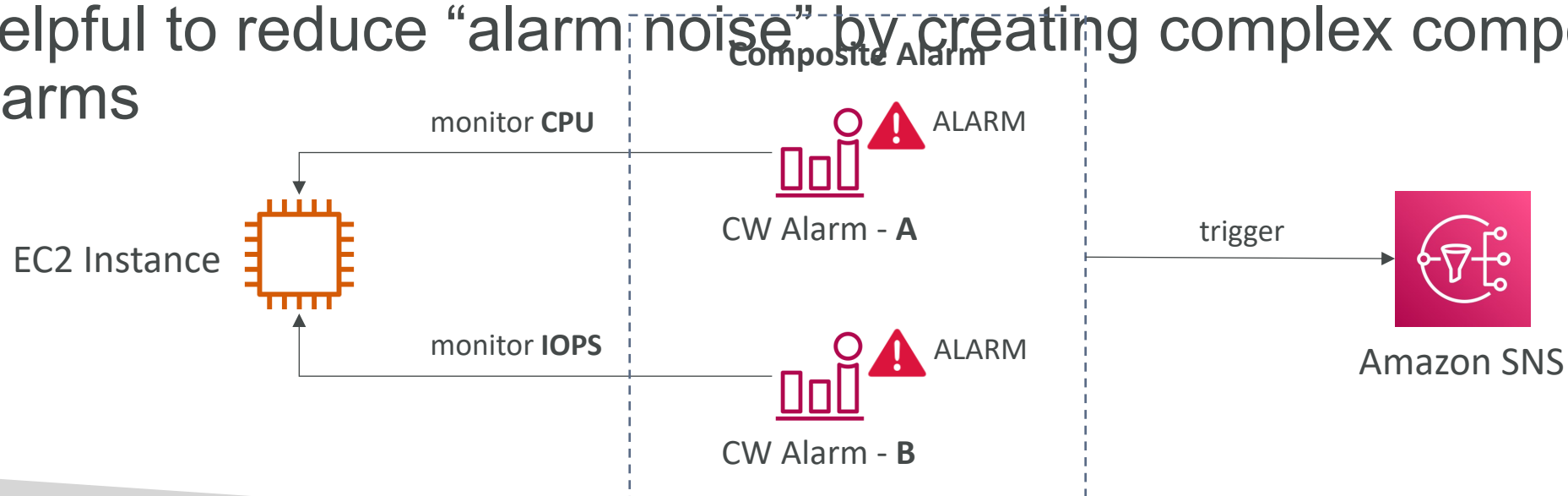
EC2 Auto Scaling



Amazon SNS

CloudWatch Alarms – Composite Alarms

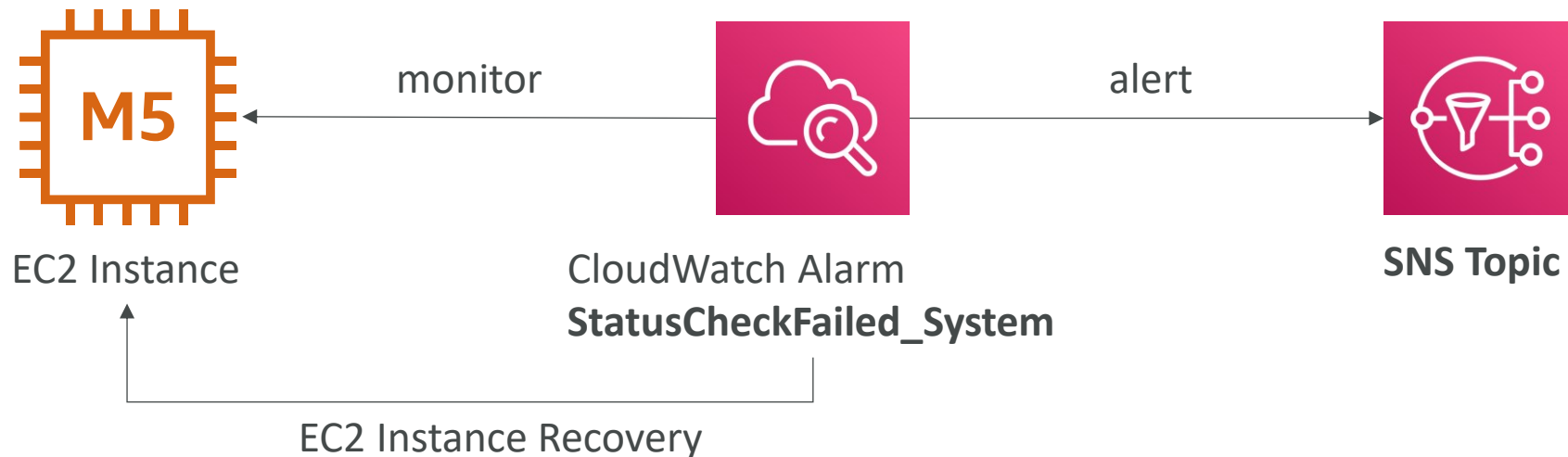
- CloudWatch Alarms are on a single metric
- **Composite Alarms are monitoring the states of multiple other alarms**
- AND and OR conditions
- Helpful to reduce “alarm noise” by creating complex composite alarms



EC2 Instance Recovery

- Status Check:

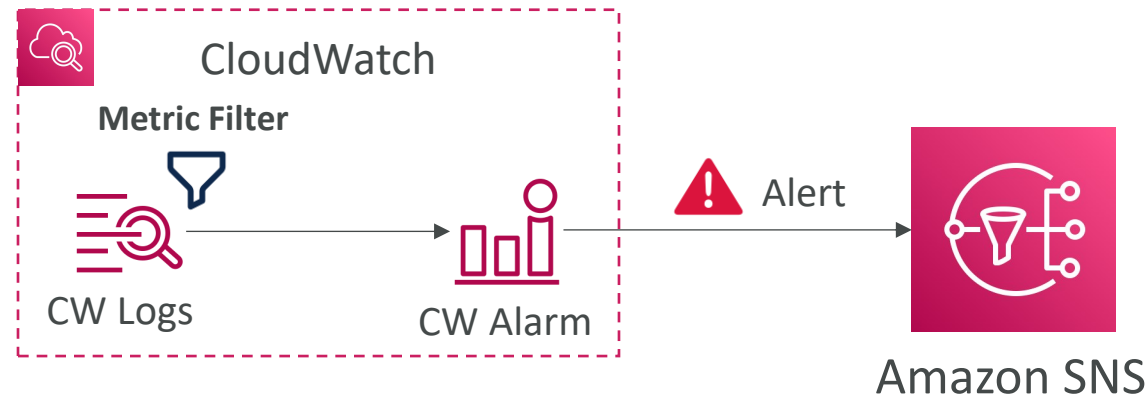
- Instance status = check the EC2 VM
- System status = check the underlying hardware
- Attached EBS status = check attached EBS volumes



- **Recovery:** Same Private, Public, Elastic IP, metadata, placement group

CloudWatch Alarm: good to know

- Alarms can be created based on CloudWatch Logs Metrics Filters



- To test alarms and notifications, set the alarm state to Alarm using CLI

aws cloudwatch set-alarm-state --alarm-name "myalarm" --state-value ALARM --state-reason "testing purposes"

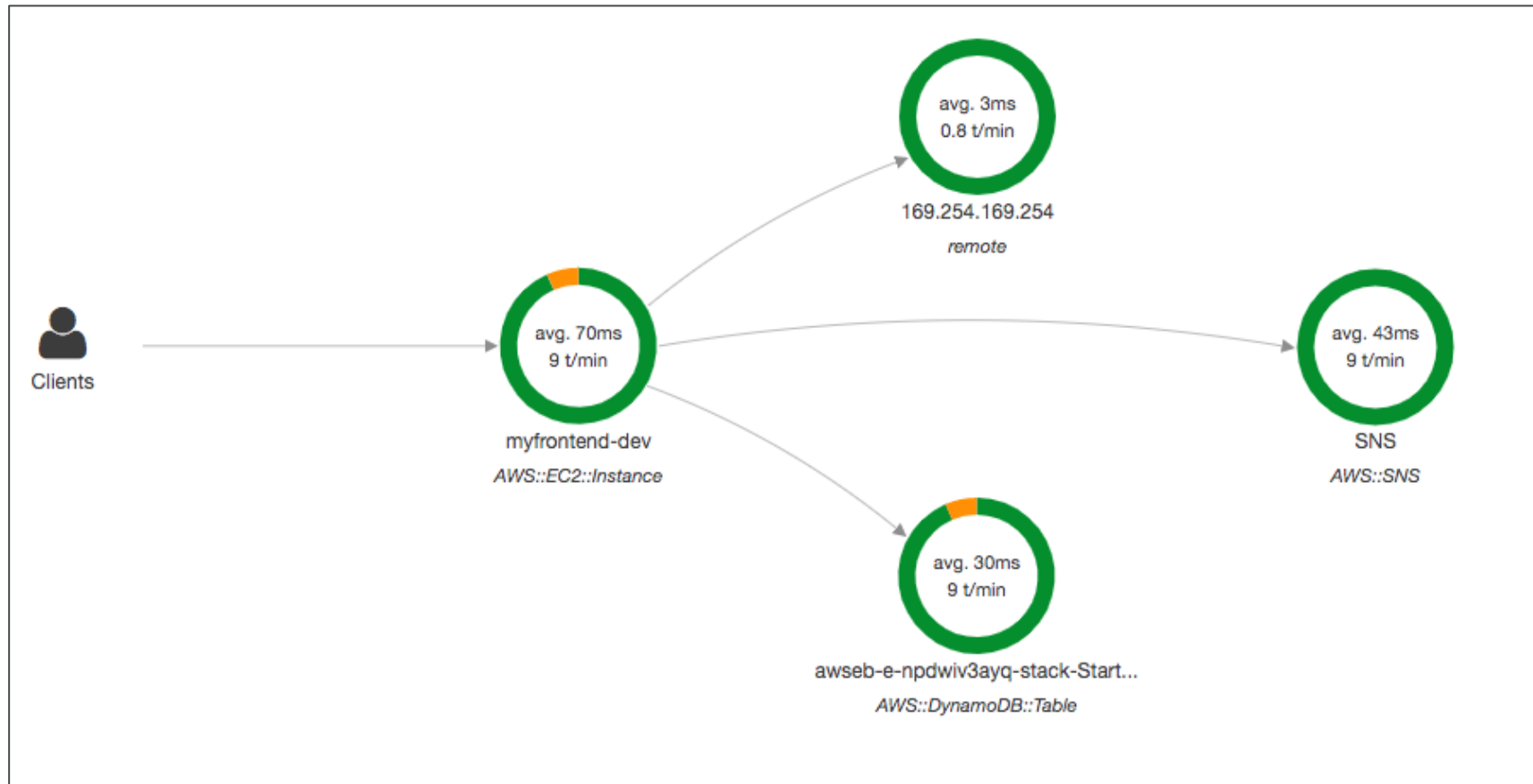
AWS X-Ray



- Debugging in Production, the good old way:
 - Test locally
 - Add log statements everywhere
 - Re-deploy in production
- Log formats differ across applications using CloudWatch and analytics is hard.
- Debugging: monolith “easy”, distributed services “hard”
- No common views of your entire architecture!
- Enter... AWS X-Ray!

AWS X-Ray

Visual analysis of our applications



AWS X-Ray advantages

- Troubleshooting performance (bottlenecks)
- Understand dependencies in a microservice architecture
- Pinpoint service issues
- Review request behavior
- Find errors and exceptions
- Are we meeting time SLA?
- Where I am throttled?
- Identify users that are impacted

X-Ray compatibility

- AWS Lambda
- Elastic Beanstalk
- ECS
- ELB
- API Gateway
- EC2 Instances or any application server (even on premise)

AWS X-Ray Leverages Tracing

- Tracing is an end to end way to following a “request”
- Each component dealing with the request adds its own “trace”
- Tracing is made of segments (+ sub segments)
- Annotations can be added to traces to provide extra-information
- Ability to trace:
 - Every request
 - Sample request (as a % for example or a rate per minute)
- X-Ray Security:
 - IAM for authorization
 - KMS for encryption at rest

AWS X-Ray

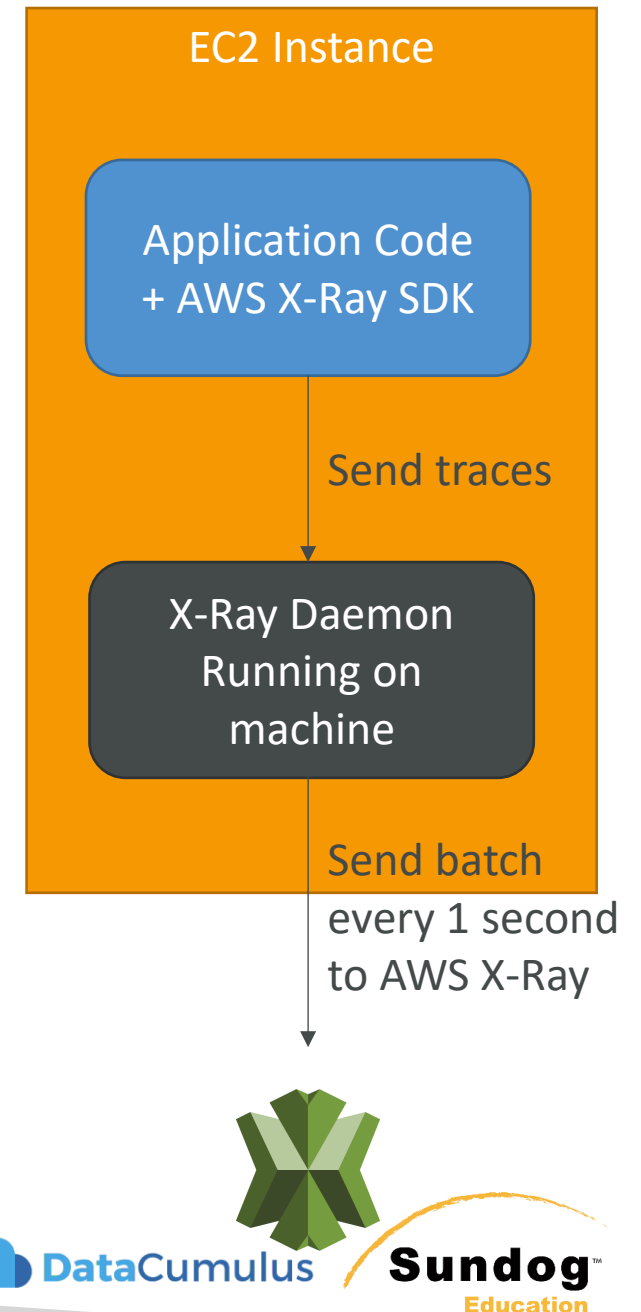
How to enable it?

1) Your code (Java, Python, Go, Node.js, .NET) must import the **AWS X-Ray SDK**

- Very little code modification needed
- The application SDK will then capture:
 - Calls to AWS services
 - HTTP / HTTPS requests
 - Database Calls (MySQL, PostgreSQL, DynamoDB)
 - Queue calls (SQS)

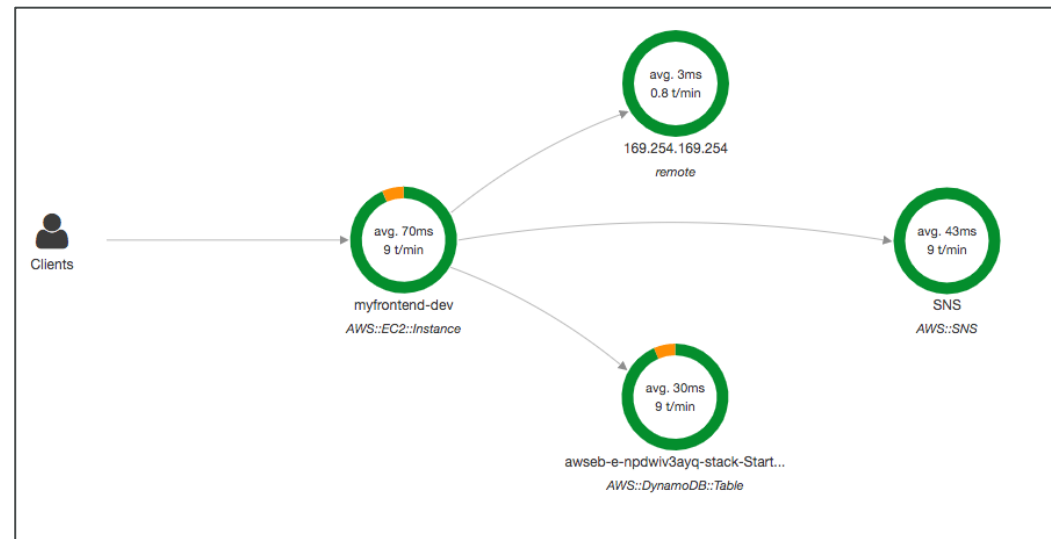
2) Install the X-Ray daemon or enable X-Ray AWS Integration

- X-Ray daemon works as a low level UDP packet interceptor (Linux / Windows / Mac...)
- AWS Lambda / other AWS services already run the X-Ray daemon for you
- Each application must have the IAM rights to write data to X-Ray



The X-Ray magic

- X-Ray service collects data from all the different services
- Service map is computed from all the segments and traces
- X-Ray is graphical, so even non technical people can help troubleshoot



AWS X-Ray Troubleshooting

- If X-Ray is not working on EC2
 - Ensure the EC2 IAM Role has the proper permissions
 - Ensure the EC2 instance is running the X-Ray Daemon
- To enable on AWS Lambda:
 - Ensure it has an IAM execution role with proper policy (AWSX-RayWriteOnlyAccess)
 - Ensure that X-Ray is imported in the code
 - Enable **Lambda X-Ray Active Tracing**

Amazon QuickSight

Business analytics and visualizations in the cloud

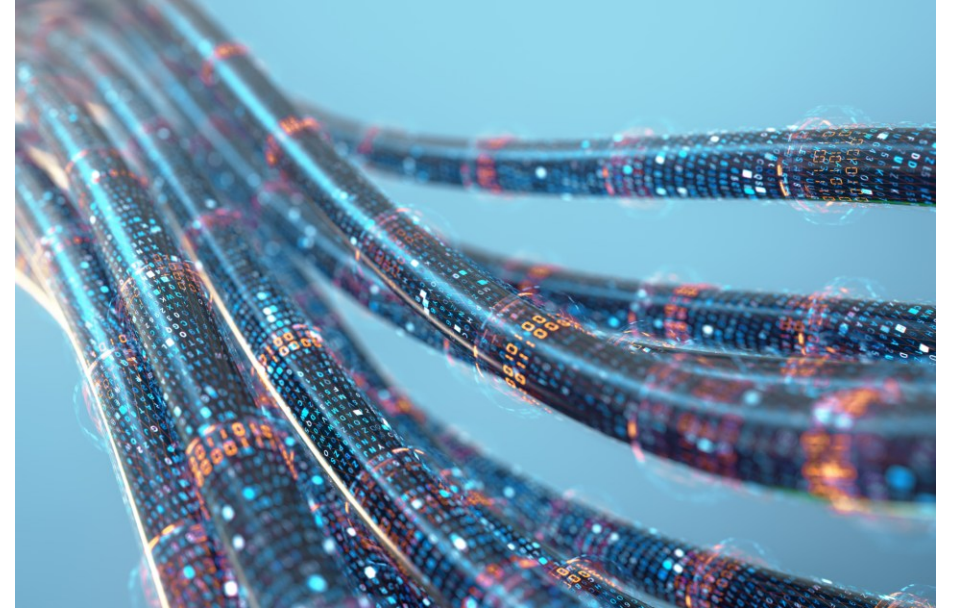
What is QuickSight?

- Fast, easy, cloud-powered business analytics service
- Allows all employees in an organization to:
 - Build visualizations
 - Perform ad-hoc analysis
 - Quickly get business insights from data
 - Anytime, on any device (browsers, mobile)
- Serverless



QuickSight Data Sources

- Redshift
- Aurora / RDS
- Athena
- EC2-hosted databases
- Files (S3 or on-premises)
 - Excel
 - CSV, TSV
 - Common or extended log format
- AWS IoT Analytics
- Data preparation allows limited ETL



SPICE

- Data sets are imported into SPICE
 - Super-fast, Parallel, In-memory Calculation Engine
 - Uses columnar storage, in-memory, machine code generation
 - Accelerates interactive queries on large datasets
- Each user gets 10GB of SPICE
- Highly available / durable
- Scales to hundreds of thousands of users

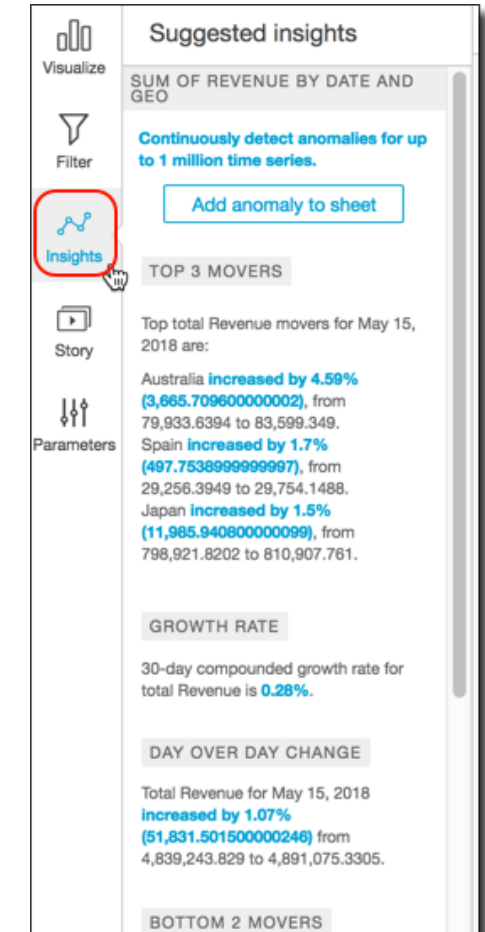
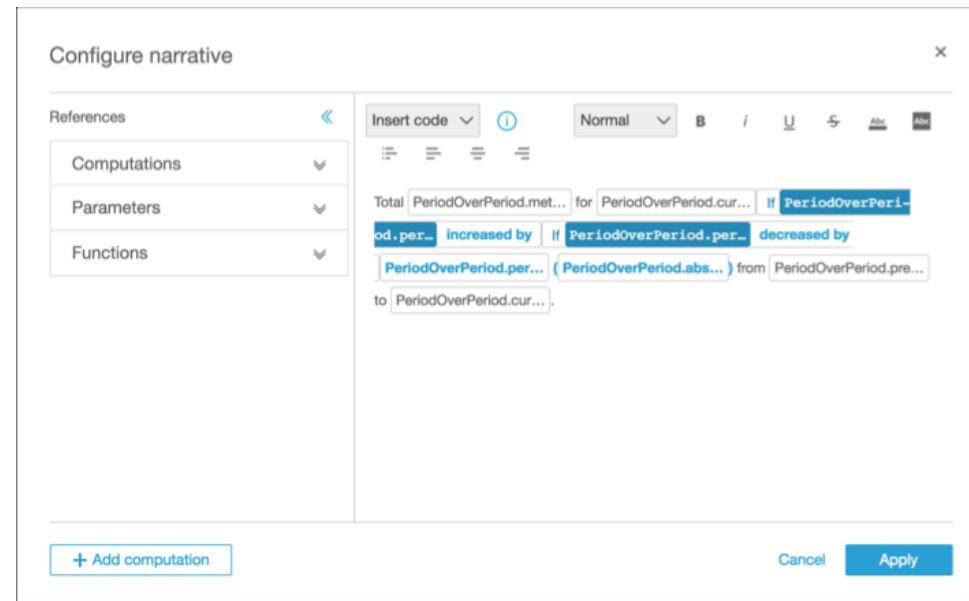


QuickSight Use Cases

- Interactive ad-hoc exploration / visualization of data
- Dashboards and KPI's
- Analyze / visualize data from:
 - Logs in S3
 - On-premise databases
 - AWS (RDS, Redshift, Athena, S3)
 - SaaS applications, such as Salesforce
 - Any JDBC/ODBC data source

Machine Learning Insights

- Anomaly detection
- Forecasting
- Auto-narratives



Quicksight Q

- Machine learning-powered
- Answers business questions with Natural Language Processing
 - “What are the top-selling items in Florida?”
- Offered as an add-on for given regions
- Personal training on how to use it is required
- Must set up *topics* associated with *datasets*
 - Datasets and their fields must be NLP-friendly
 - How to handle dates must be defined



Quicksight Paginated Reports

- Reports designed to be printed
- May span many pages
- Can be based on existing Quicksight dashboards
- New in Nov 2022

Financial Statements

Income Statement (P&L)

Report Generated at
10/03/2022

Actual Vs Budget

				2021				
				Actual	Budget	Variance	Variance %	
Operating Income	Gross Profit	Revenue	Lamps	\$165,000.00	\$142,230.83	22,769.17	16.00%	
			Fixtures	\$40,000.00	\$35,454.69	4,545.31	12.82%	
			Subtotal	\$205,000.00	\$177,685.52	27,314.48	28.82%	
	Cost of Goods sold		Direct Labour	-\$15,000.00	-\$13,987.20	-1,012.8	7.24%	
			Material	-\$54,000.00	-\$45,540.89	-8,459.11	18.57%	
			Subtotal	-\$69,000.00	-\$59,528.09	-9,471.91	25.81%	
		Subtotal		\$136,000.00	\$118,157.43	17,842.57	54.63%	
	Operating Expenses	Operating Expenses	Advertising	-\$15,000.00	-\$12,149.89	-2,850.11	23.45%	
			Auto and Travel	-\$7,500.00	-\$6,844.58	-655.42	9.57%	
			Insurance	-\$4,000.00	-\$3,478.07	-521.93	15.00%	
			Legal and accounting	-\$10,000.00	-\$8,896.64	-1,103.36	12.40%	
			Utilities	-\$4,000.00	-\$3,440.29	-559.71	16.26%	
			Rent	-\$24,000.00	-\$19,048.27	-4,951.73	25.99%	
			Salaries and benefits	-\$15,000.00	-\$28,176.08	-6,823.92	24.21%	
			Subtotal	-\$99,500.00	-\$82,033.82	-17,466.18	126.88%	
		Subtotal		-\$99,500.00	-\$82,033.82	-17,466.18	126.88%	
	Subtotal			\$36,500.00	\$36,123.61	\$376.39	181.51%	
Other ...	Depreciation and...	Depreciation and...	Depreciation	-\$2,500.00	-\$2,209.04	-290.96	13.17%	

Page 1 of 5

ABC Company Confidential



Financial Statements

Income Statement (P&L)

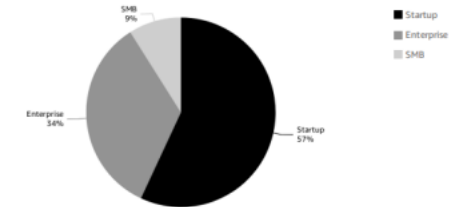
Report Generated at

10/03/2022

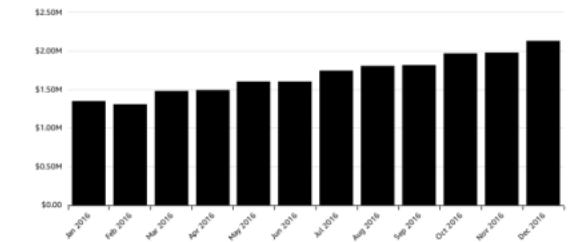
				2021			
				Actual	Budget	Variance	Variance %
Other Income/Expenses	Depreciation and Amortization	Depreciation and Amortization	Interest	-\$6,500.00	-\$5,210.69	-1,289.31	24.74%
				-\$9,000.00	-\$7,419.73	-1,580.27	37.91%
				-\$9,000.00	-\$7,419.73	-1,580.27	37.91%
	Subtotal	Subtotal		-\$9,000.00	-\$7,419.73	-1,580.27	37.91%

Section 1 - The Year at a Glance

Annual expenditures by category



Sum of Billed Amount by Date



Year-End Summary 2016



QuickSight Anti-Patterns

- ~~Highly formatted canned reports~~
 - ~~QuickSight is for ad-hoc queries, analysis, and visualization~~
 - No longer true with paginated reports!
- ETL
 - Use Glue instead, although QuickSight can do some transformations



QuickSight Security

- Multi-factor authentication on your account
- VPC connectivity
 - Add QuickSight's IP address range to your database security groups
- Row-level security
 - Column-level security too (CLS) – Enterprise edition only
- Private VPC access
 - Elastic Network Interface, AWS Direct Connect



QuickSight User Management

- Users defined via IAM, or email signup
- SAML-based single sign-on
- Active Directory integration (Enterprise Edition)
- MFA



QuickSight Pricing

- Annual subscription
 - Standard: \$9 / user / month
 - Enterprise: \$18 / user / month
 - Enterprise with Q: \$28 / user / month
- Extra SPICE capacity (beyond 10GB)
 - \$0.25 (standard) \$0.38 (enterprise) / GB / month
- Month to month
 - Standard: \$12 / user / month
 - Enterprise: \$24 / user / month
 - Enterprise with Q: \$34 / user / month
- Additional charges for paginated reports, alerts & anomaly detection, Q capacity, readers, and reader session capacity.
- Enterprise edition
 - ~~Encryption at rest~~ (now included in standard as well)
 - Microsoft Active Directory integration

QuickSight Dashboards

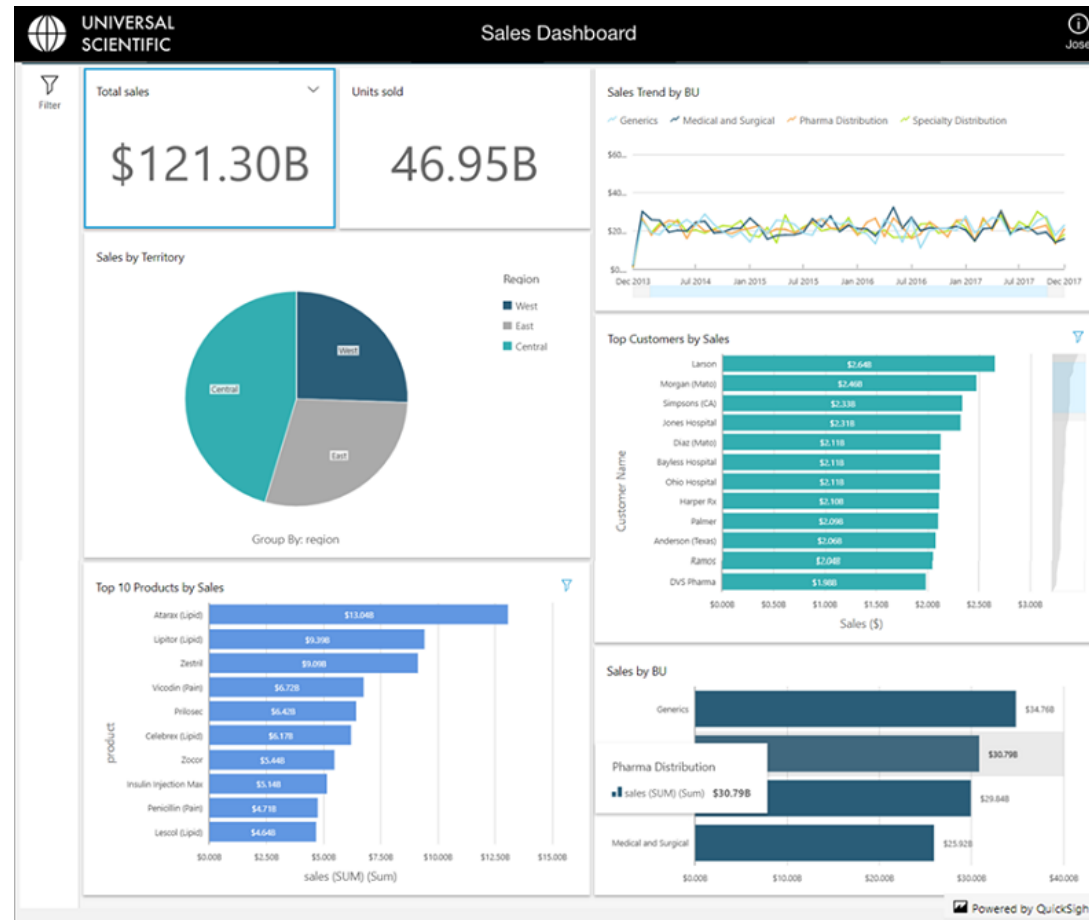


Image: AWS Big Data Blog

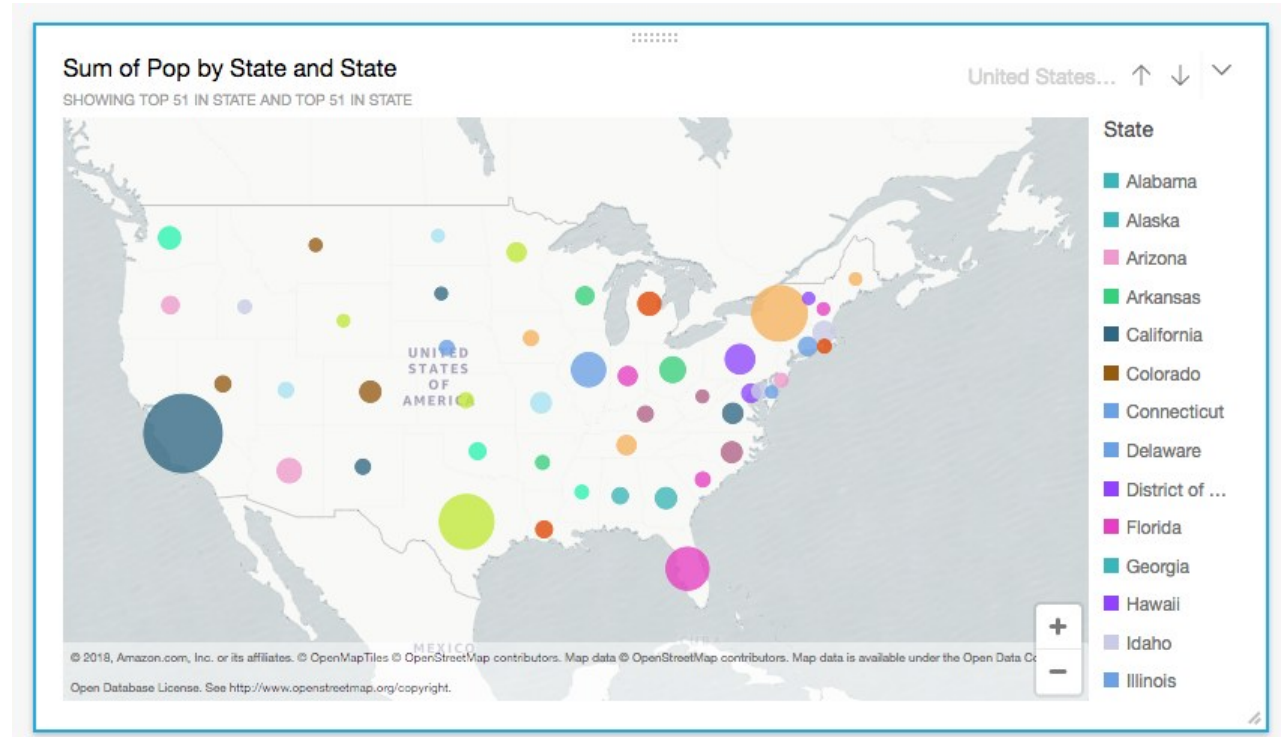
QuickSight Visual Types

- AutoGraph
- Bar Charts
 - For comparison and distribution (histograms)
- Line graphs
 - For changes over time
- Scatter plots, heat maps
 - For correlation
- Pie graphs, tree maps
 - For aggregation
- Pivot tables
 - For tabular data



Additional Visual Types

- KPIs
- Geospatial Charts (maps)
- Donut Charts
- Gauge Charts
- Word Clouds



Bar Charts: Comparison, Distribution

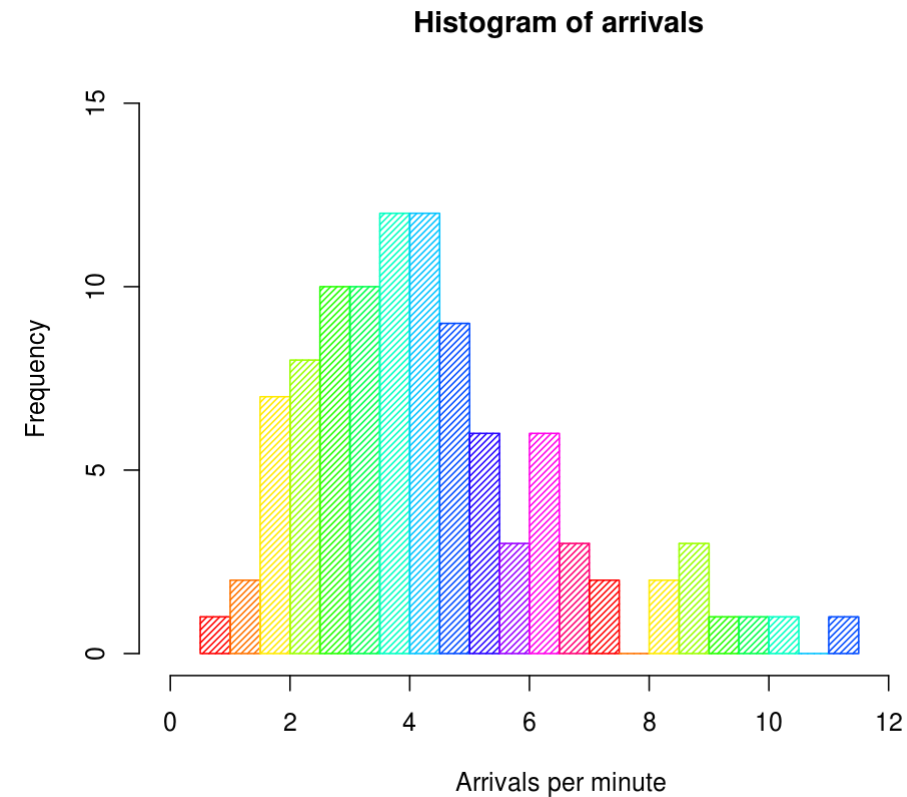
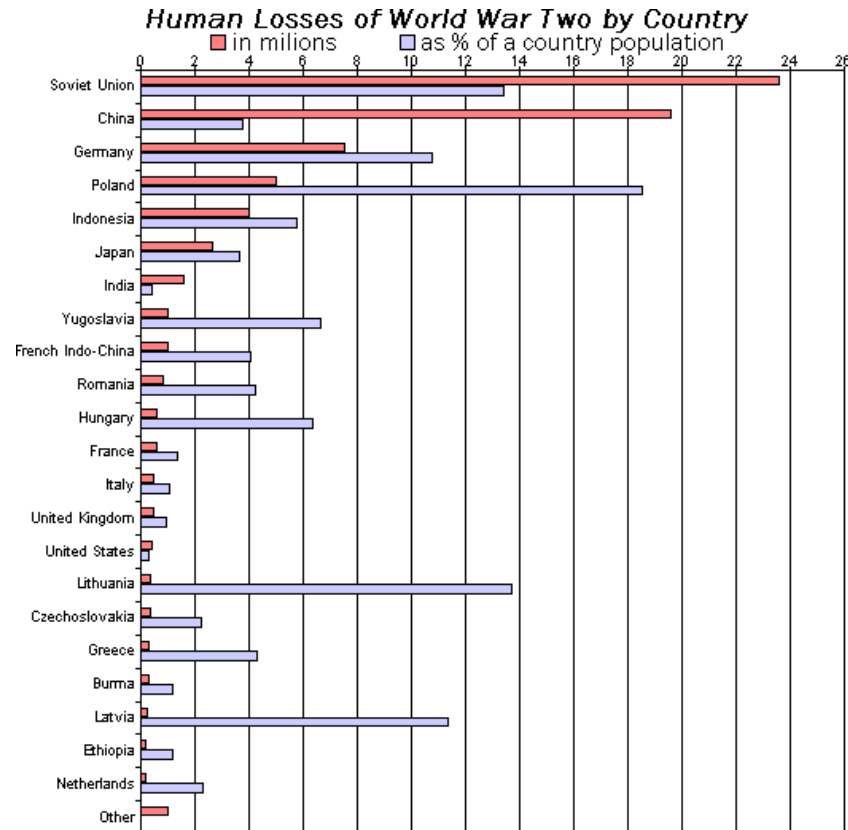
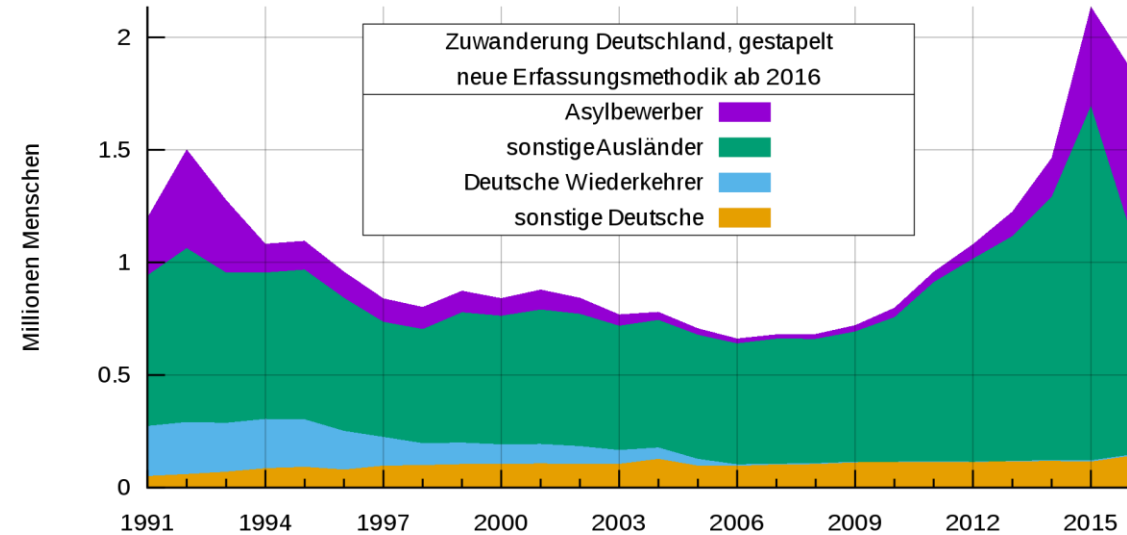
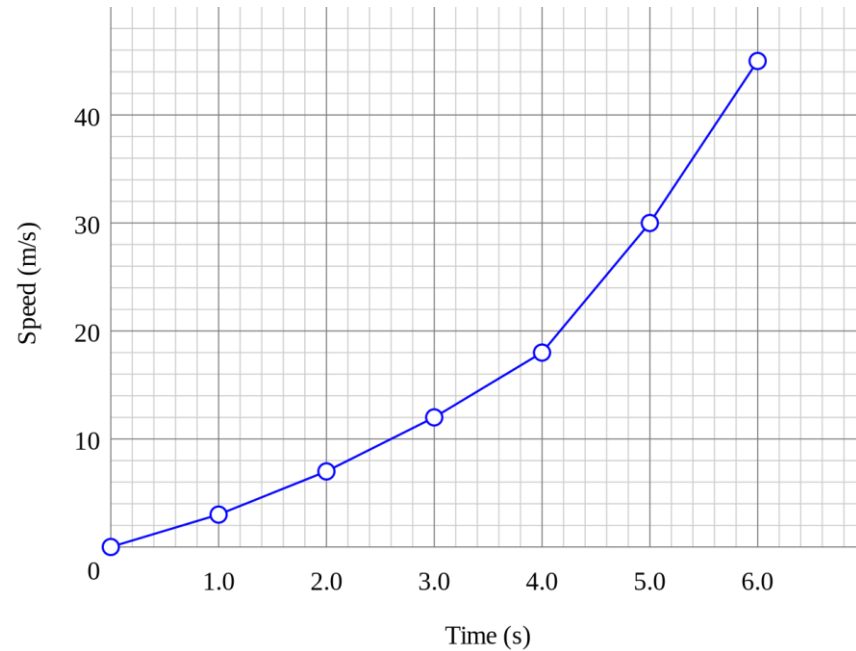
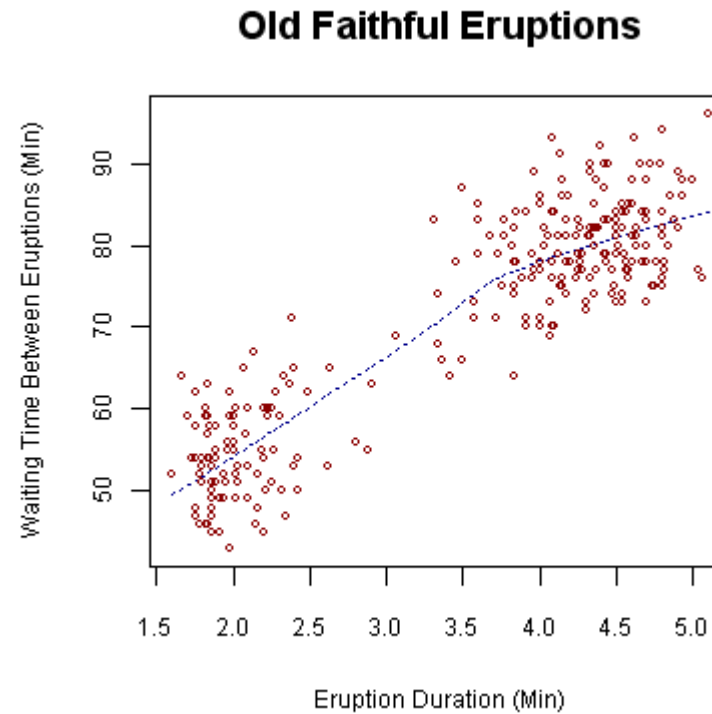


Image: DanielPenfield, Wikipedia CC BY-SA 3.0

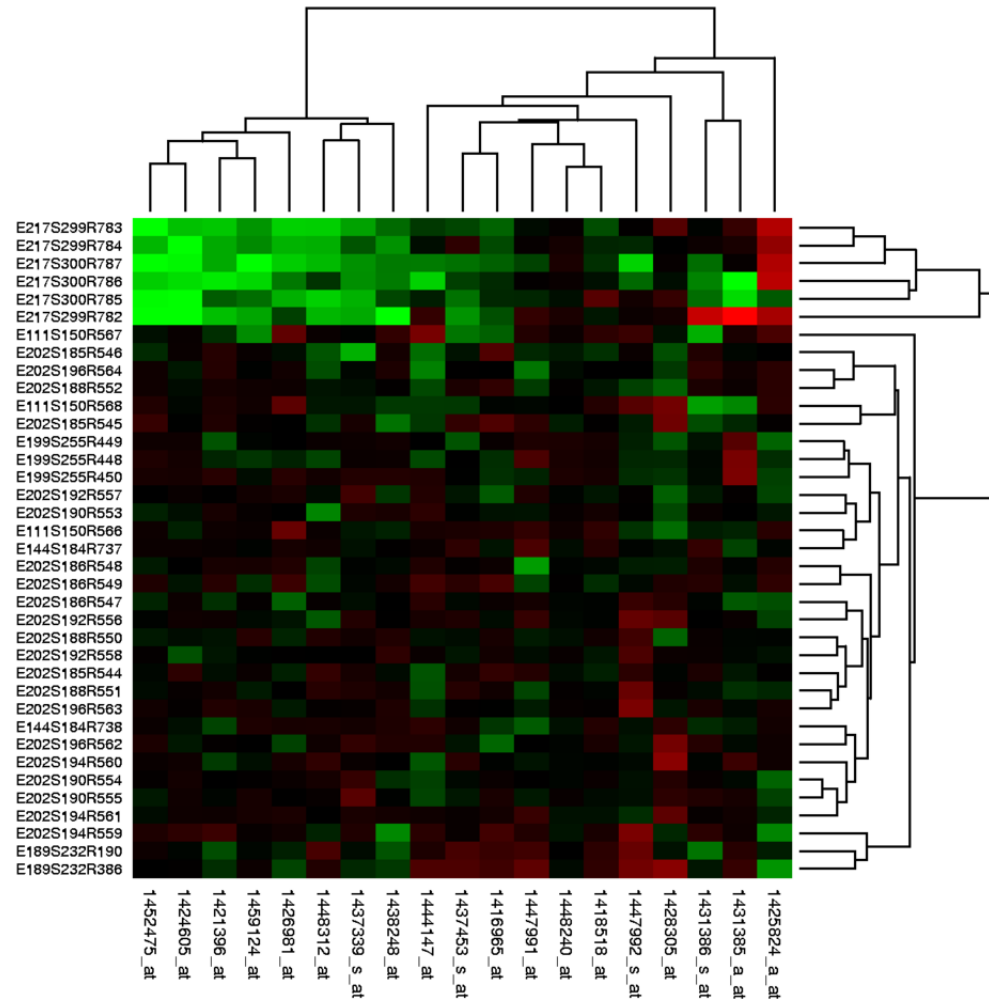
Line Charts: Changes over Time



Scatter Plots: Correlation



Heat Maps: Correlation



Pie Charts: Aggregation

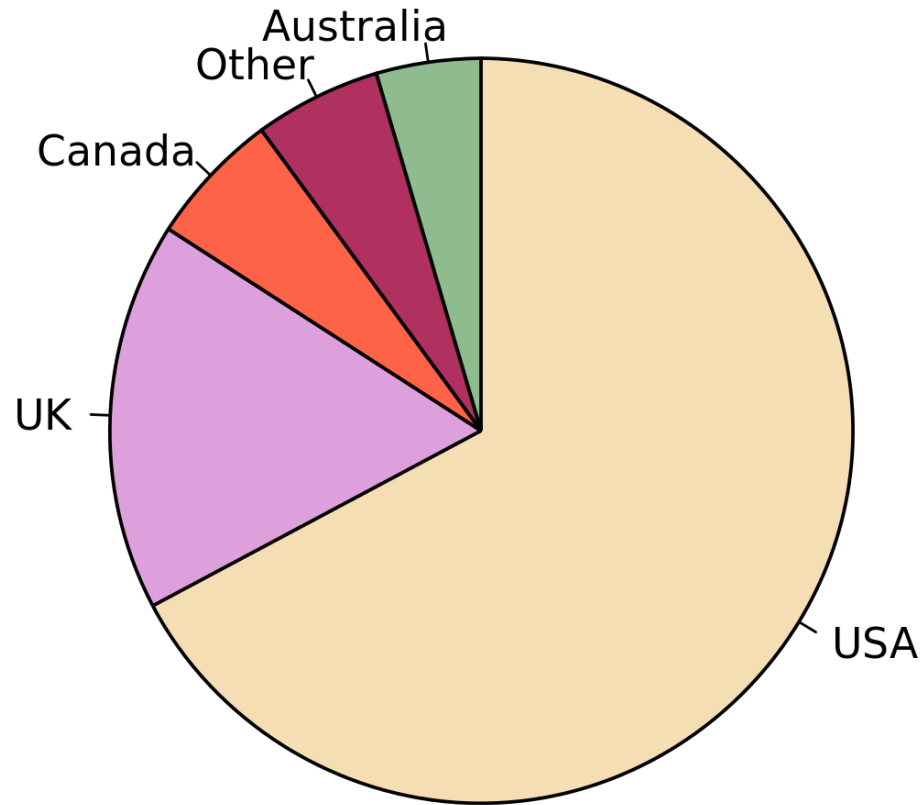
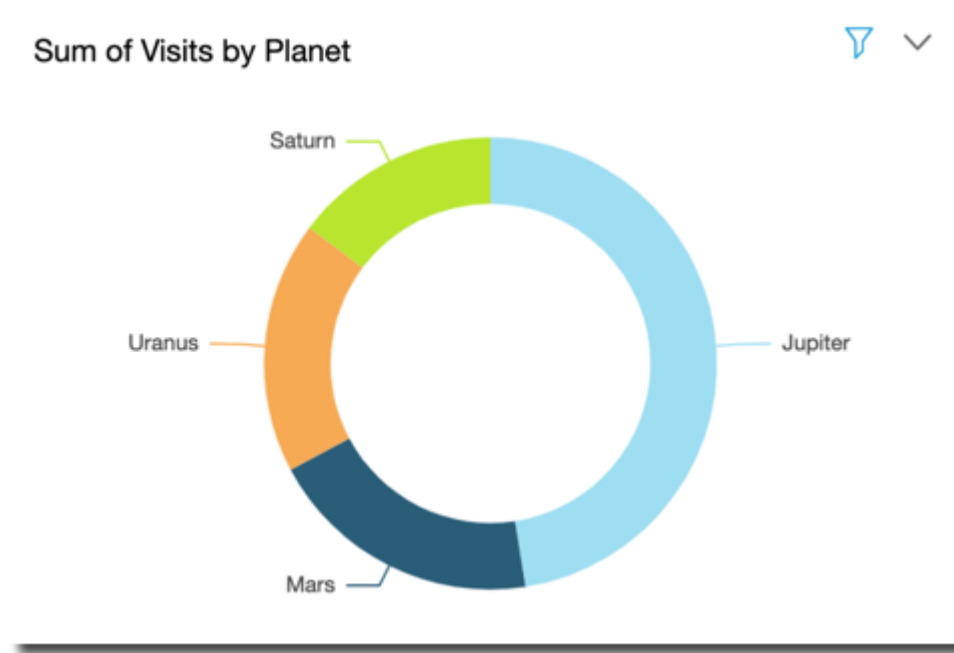
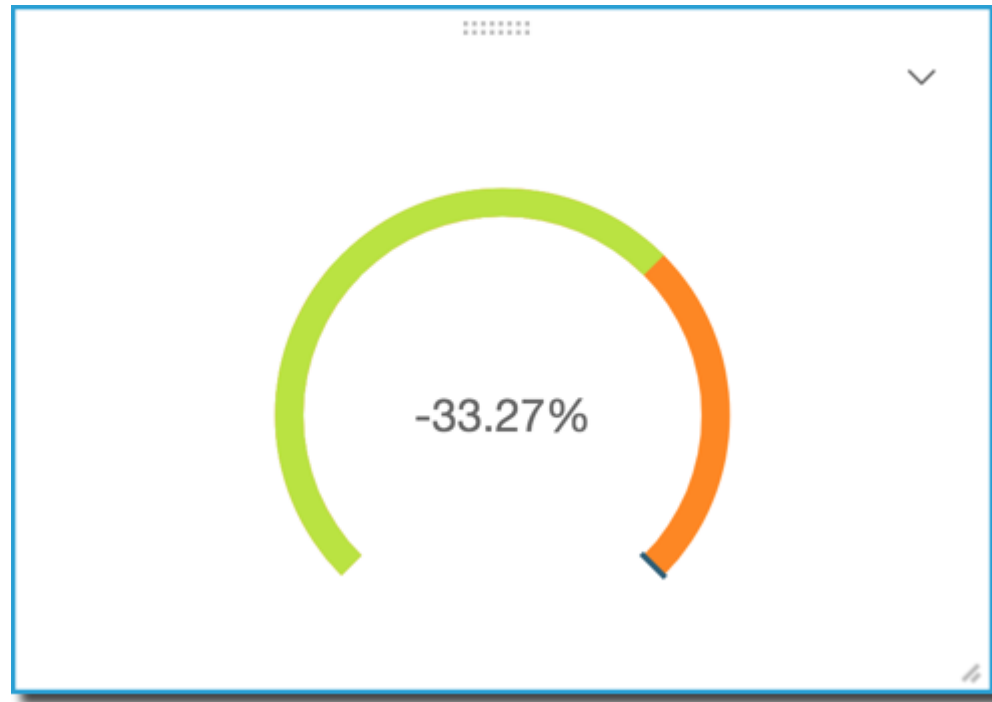


Image: M.W. Toews, Wikipedia, CC BY-SA 4.0

Donut Charts: Percentage of Total Amount



Gauge Charts: Compare values in a measure



Tree Maps: Heirarchical Aggregation

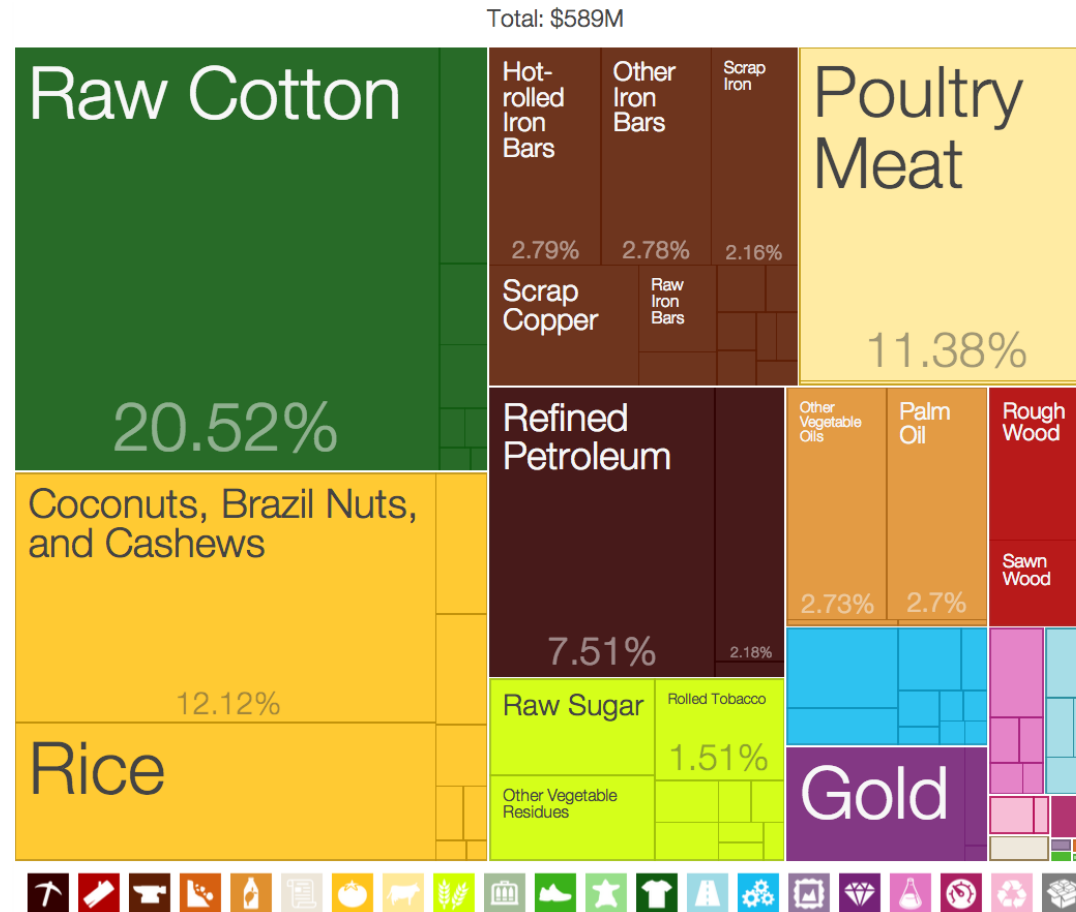


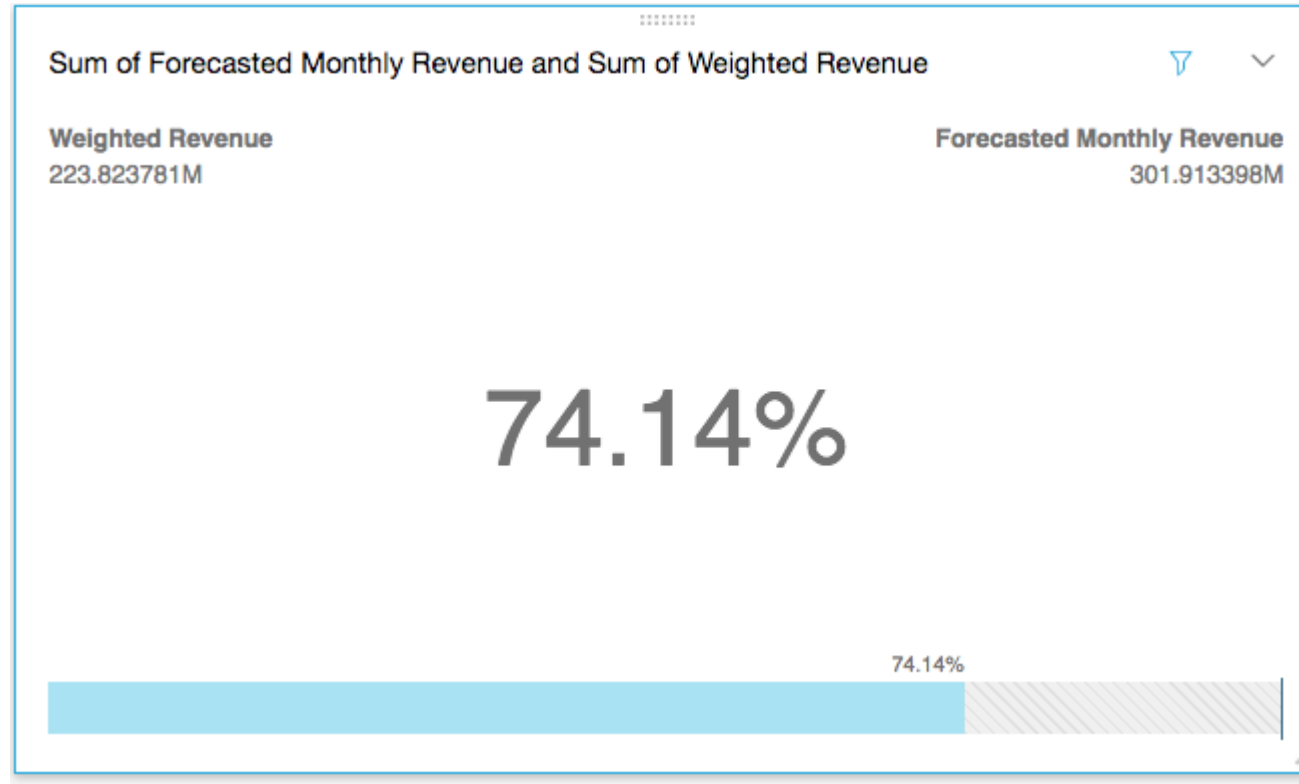
Image: Harvard-MIT Observatory of Economic Complexity, Wikipedia, CC-BY-SA 3.0

Pivot Tables: Tabular Data

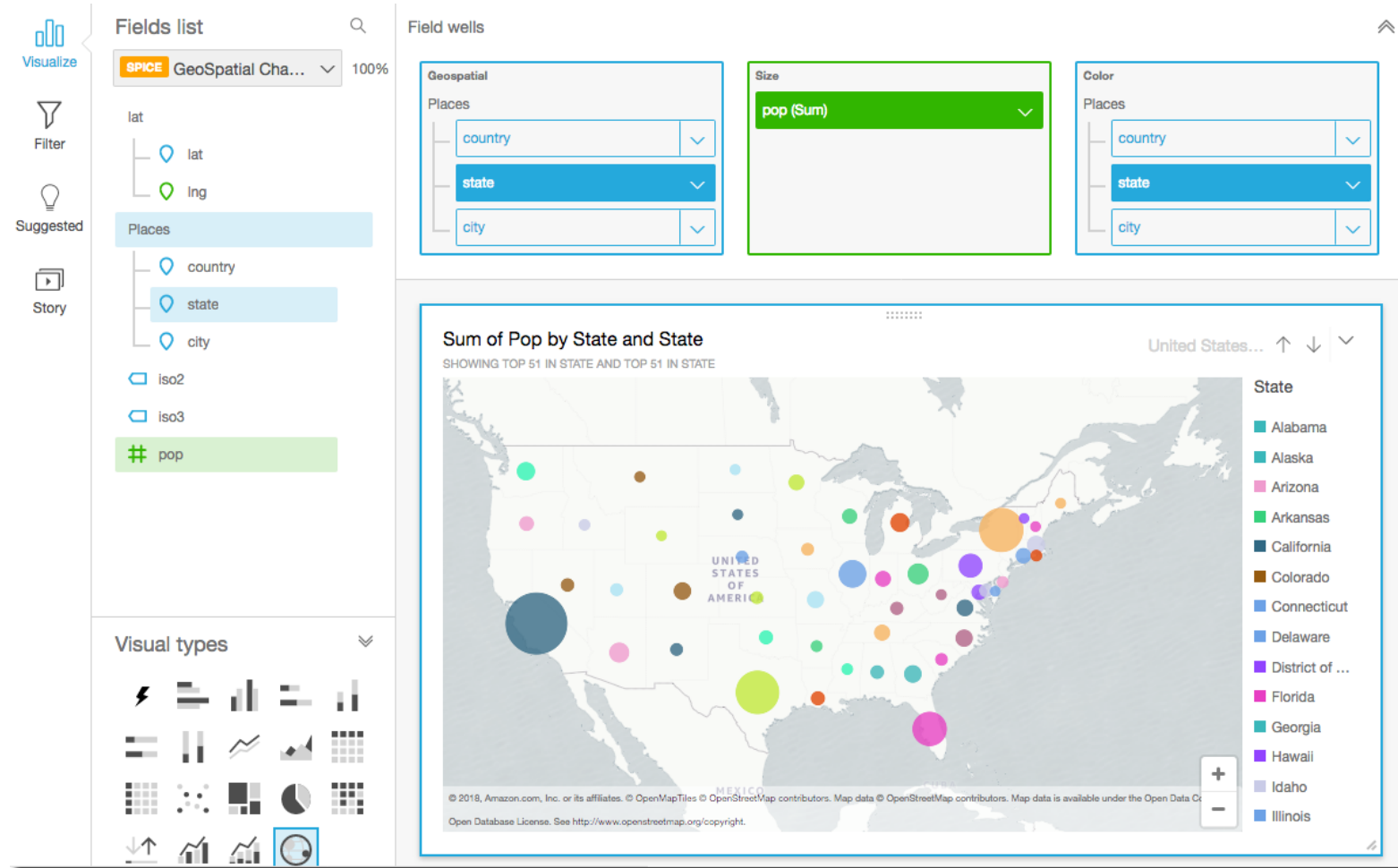
	A	B	C	D	E	F	G
1	Region	Gender	Style	Ship Date	Units	Price	Cost
2	East	Boy	Tee	1/31/2005	12	11.04	10.42
3	East	Boy	Golf	1/31/2005	12	13	12.6
4	East	Boy	Fancy	1/31/2005	12	11.96	11.74
5	East	Girl	Tee	1/31/2005	10	11.27	10.56
6	East	Girl	Golf	1/31/2005	10	12.12	11.95
7	East	Girl	Fancy	1/31/2005	10	13.74	13.33
8	West	Boy	Tee	1/31/2005	11	11.44	10.94
9	West	Boy	Golf	1/31/2005	11	12.63	11.73
10	West	Boy	Fancy	1/31/2005	11	12.06	11.51
11	West	Girl	Tee	1/31/2005	15	13.42	13.29
12	West	Girl	Golf	1/31/2005	15	11.48	10.67

Sum of Units	Ship Date ▼					
Region ▼	1/31/2005	2/28/2005	3/31/2005	4/30/2005	5/31/2005	6/30/2005
East	66	80	102	116	127	125
North	96	117	138	151	154	156
South	123	141	157	178	191	202
West	78	97	117	136	150	157
(blank)						
Grand Total	363	435	514	581	622	640

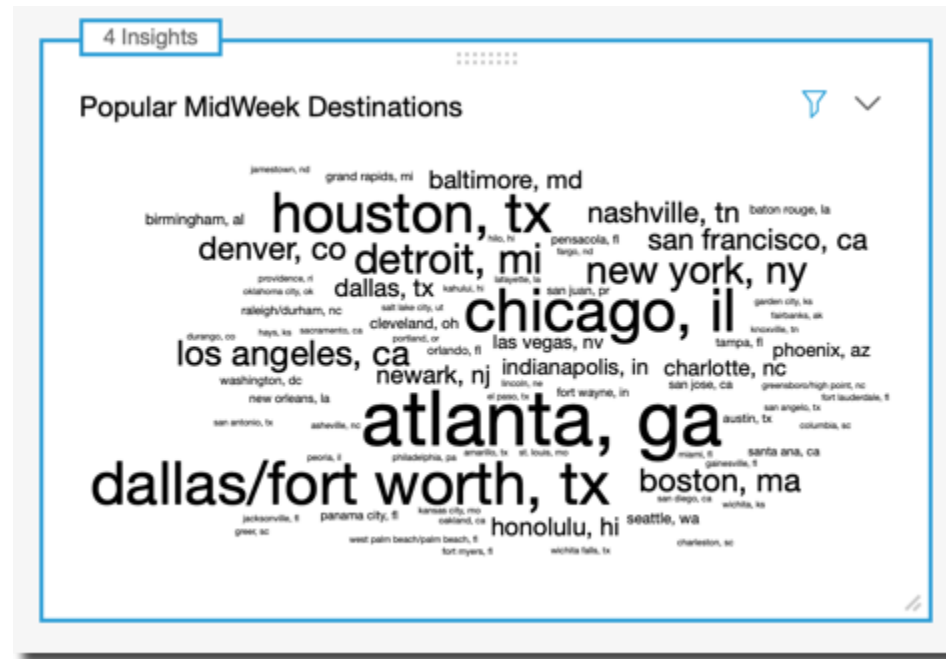
KPI's: compare key value to its target value



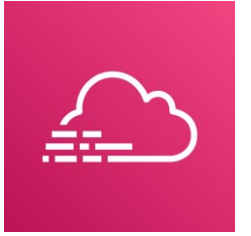
Geospatial Charts



Word Clouds: word or phrase frequency

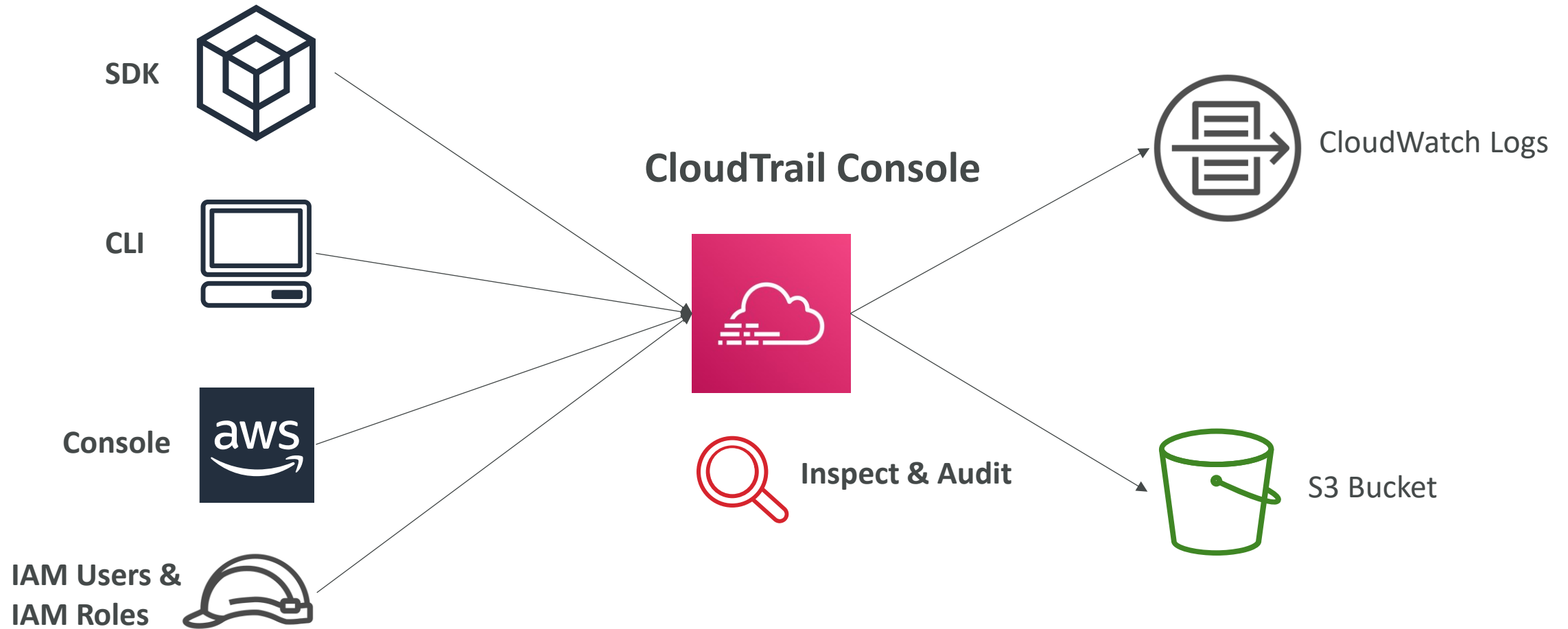


AWS CloudTrail

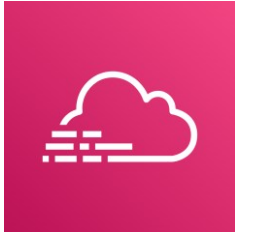


- **Provides governance, compliance and audit for your AWS Account**
- CloudTrail is enabled by default!
- **Get an history of events / API calls made within your AWS Account** by:
 - Console
 - SDK
 - CLI
 - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs or S3
- **A trail can be applied to All Regions (default) or a single Region.**

CloudTrail Diagram



CloudTrail Events



- **Management Events:**

- Operations that are performed on resources in your AWS account
- Examples:
 - Configuring security (IAM AttachRolePolicy)
 - Configuring rules for routing data (Amazon EC2 CreateSubnet)
 - Setting up logging (AWS CloudTrail CreateTrail)
- **By default, trails are configured to log management events.**
- Can separate **Read Events** (that don't modify resources) from **Write Events** (that may modify resources)

- **Data Events:**

- **By default, data events are not logged (because high volume operations)**
- Amazon S3 object-level activity (ex: GetObject, DeleteObject, PutObject): can separate Read and Write Events
- AWS Lambda function execution activity (the Invoke API)

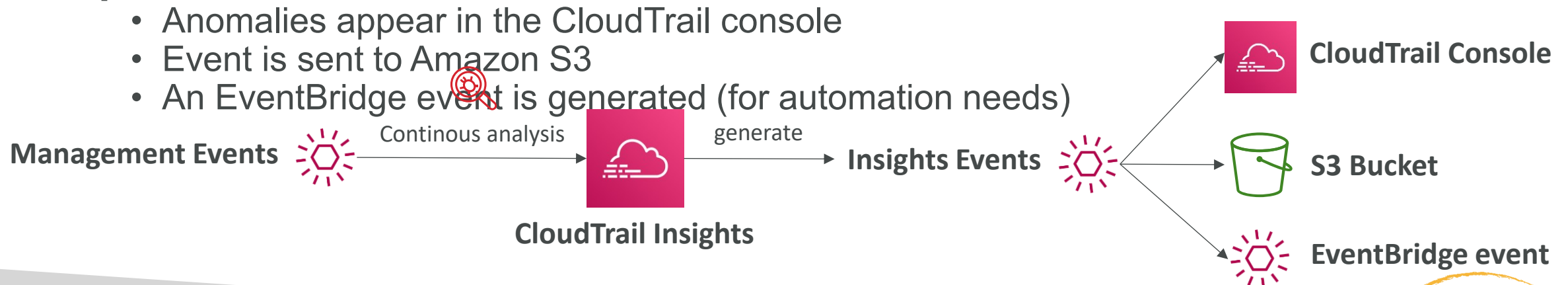
- **CloudTrail Insights Events:**

- See next slide 😊

CloudTrail Insights

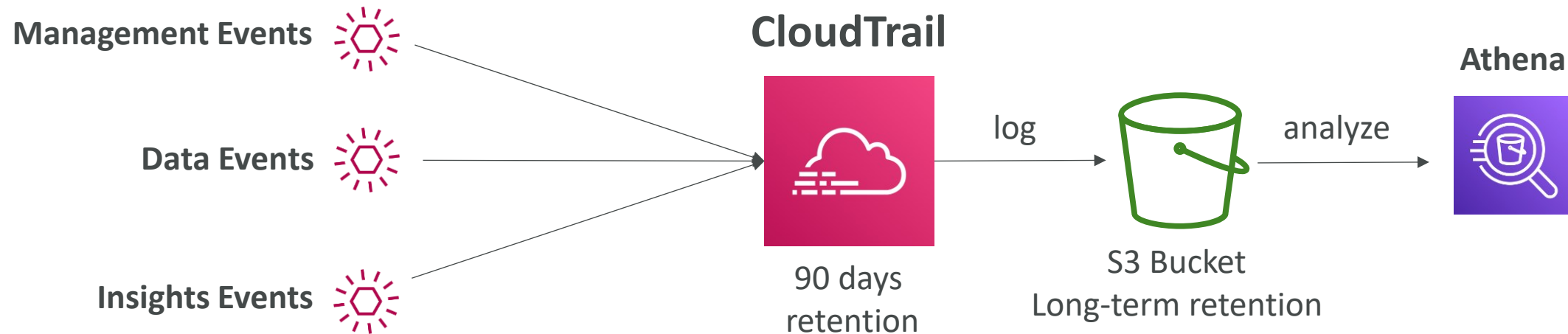


- Enable **CloudTrail Insights to detect unusual activity** in your account:
 - inaccurate resource provisioning
 - hitting service limits
 - Bursts of AWS IAM actions
 - Gaps in periodic maintenance activity
- CloudTrail Insights analyzes normal management events to create a baseline
- And then **continuously analyzes write events to detect unusual patterns**

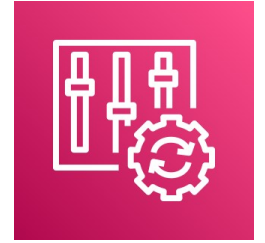


CloudTrail Events Retention

- Events are stored for 90 days in CloudTrail
- To keep events beyond this period, log them to S3 and use Athena



AWS Config



- Helps with auditing and recording **compliance** of your AWS resources
- Helps record configurations and changes over time
- Questions that can be solved by AWS Config:
 - Is there unrestricted SSH access to my security groups?
 - Do my buckets have any public access?
 - How has my ALB configuration changed over time?
- You can receive alerts (SNS notifications) for any changes
- AWS Config is a per-region service
- Can be aggregated across regions and accounts
- Possibility of storing the configuration data into S3 (analyzed by Athena)

Config Rules

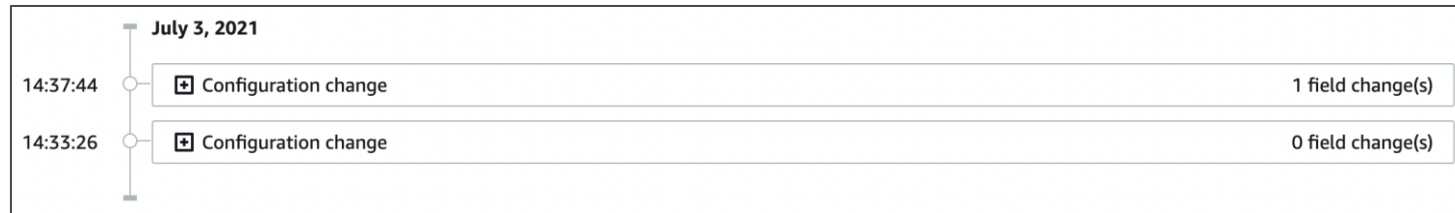
- Can use AWS managed config rules (over 75)
- Can make custom config rules (must be defined in AWS Lambda)
 - Ex: evaluate if each EBS disk is of type gp2
 - Ex: evaluate if each EC2 instance is t2.micro
- Rules can be evaluated / triggered:
 - For each config change
 - And / or: at regular time intervals
- **AWS Config Rules does not prevent actions from happening (no deny)**
- Pricing: no free tier, \$0.003 per configuration item recorded per region, \$0.001 per config rule evaluation per region

AWS Config Resource

- View compliance of a resource over time

☐	sg-077b425b1649da83e	EC2 SecurityGroup	✔ Compliant
☐	sg-0831434f1876c0c74	EC2 SecurityGroup	⚠ Noncompliant
☐	sg-09f10ed254d464f30	EC2 SecurityGroup	✔ Compliant

- View configuration of a resource over time

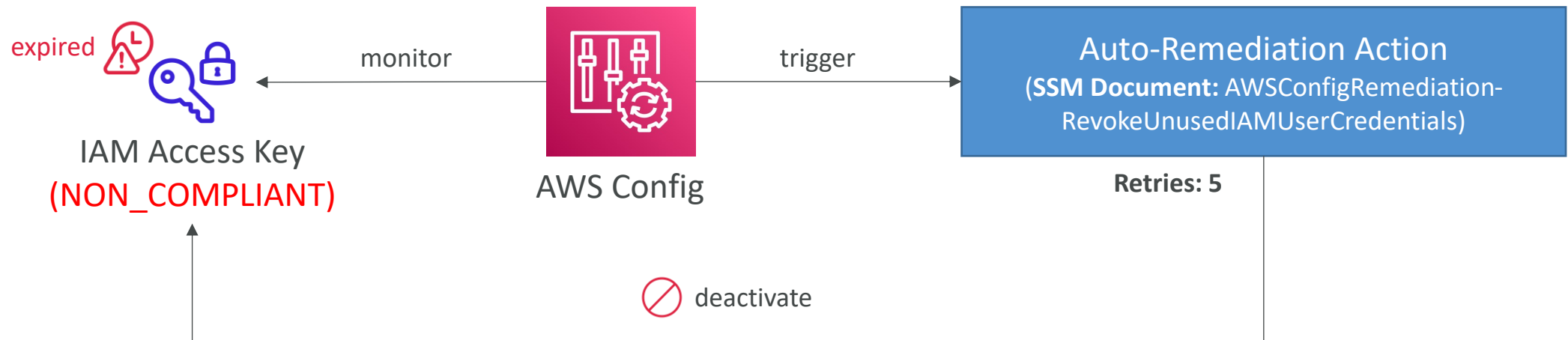


- View CloudTrail API calls of a resource over time



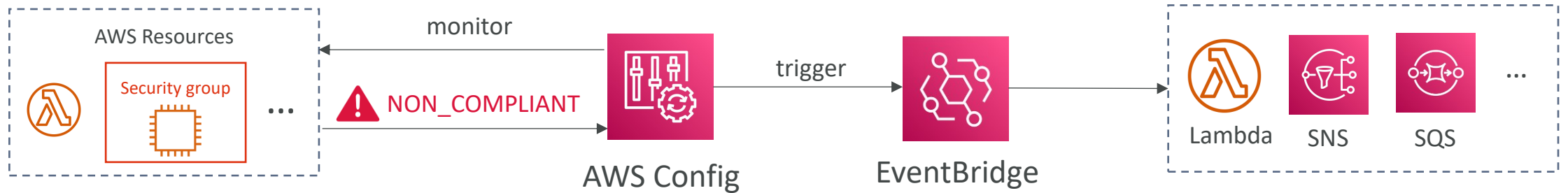
Config Rules – Remediations

- Automate remediation of non-compliant resources using SSM Automation Documents
- Use AWS-Managed Automation Documents or create custom Automation Documents
 - Tip: you can create custom Automation Documents that invokes Lambda function
- You can set **Remediation Retries** if the resource is still non-compliant after auto-remediation

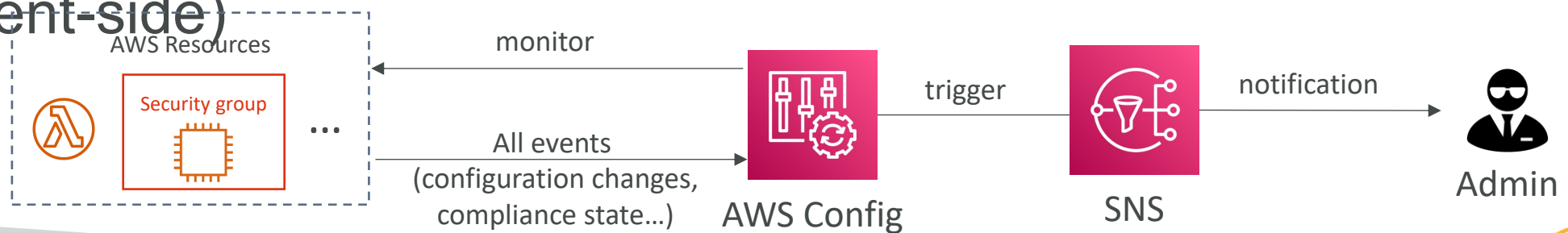


Config Rules – Notifications

- Use EventBridge to trigger notifications when AWS resources are non-compliant



- Ability to send configuration changes and compliance state notifications to SNS (all events – use SNS Filtering or filter at client-side)



CloudWatch vs CloudTrail vs Config

- CloudWatch
 - Performance monitoring (metrics, CPU, network, etc...) & dashboards
 - Events & Alerting
 - Log Aggregation & Analysis
- CloudTrail
 - Record API calls made within your Account by everyone
 - Can define trails for specific resources
 - Global Service
- Config
 - Record configuration changes
 - Evaluate resources against compliance rules
 - Get timeline of changes and compliance

For an Elastic Load Balancer

- CloudWatch:
 - Monitoring Incoming connections metric
 - Visualize error codes as % over time
 - Make a dashboard to get an idea of your load balancer performance
- Config:
 - Track security group rules for the Load Balancer
 - Track configuration changes for the Load Balancer
 - Ensure an SSL certificate is always assigned to the Load Balancer (compliance)
- CloudTrail:
 - Track who made any changes to the Load Balancer with API calls

AWS Budgets



- Create budget and **send alarms when costs exceeds the budget**
- 4 types of budgets: Usage, Cost, Reservation, Savings Plans
- For Reserved Instances (RI)
 - Track utilization
 - Supports EC2, ElastiCache, RDS, Redshift
- Up to 5 SNS notifications per budget
- Can filter by: Service, Linked Account, Tag, Purchase Option, Instance Type, Region, Availability Zone, API Operation, etc...
- Same options as AWS Cost Explorer!
- 2 budgets are free, then \$0.02/day/budget

Cost Explorer



- Visualize, understand, and manage your AWS costs and usage over time
- Create custom reports that analyze cost and usage data.
- Analyze your data at a high level: total costs and usage across all accounts
- Or Monthly, hourly, resource level granularity
- Choose an optimal **Savings Plan** (to lower prices on your bill)
- **Forecast usage up to 18 months based on previous usage**

Cost Explorer – Monthly Cost by AWS Service

Monthly costs by service



Download CSV

Instance Type	Oct 1, 2018	Nov 1, 2018	Dec 1, 2018	Jan 1, 2019
Total cost (\$)	1,312.71	1,328.54	1,125.99	1,129.65
t2.micro (\$)	486.75	475.89	405.63	409.27
c4.2xlarge (\$)	296.11	286.56	296.11	296.11

FILTERS

CLEAR ALL

Service	Include only
EC2-Instances	1
Linked Account	Include all
Region	Include all
Instance Type	Include all
Usage Type	Include all
Usage Type Group	Include all
Tag	Include All
API Operation	Include all
Charge Type	Include all

More filters

ADVANCED OPTIONS

i

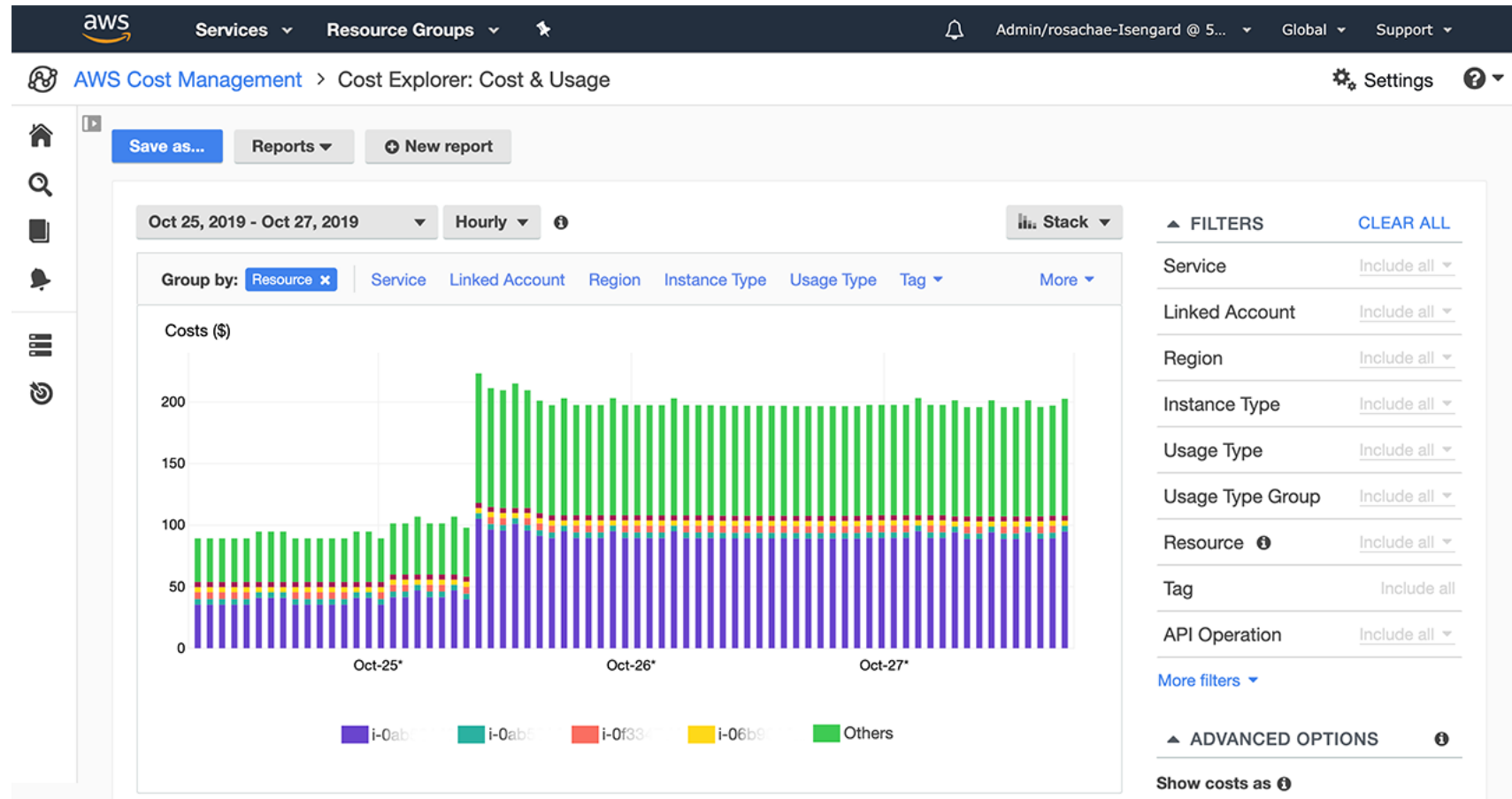
Show costs as

Unblended costs

Include costs related to

☐ Show only untagged resources

Cost Explorer– Hourly & Resource Level



Cost Explorer – Savings Plan Alternative to Reserved Instances

Recommendation options

Savings Plans type

☒ Compute

☐ EC2 Instance

Savings Plans term

☐ 1-year

☒ 3-year

Payment option

☒ All upfront

☐ Partial upfront

☐ No upfront

Based on the past

☐ 7 days

☐ 30 days

☒ 60 days

Recommendation: Purchase a Compute Savings Plan at a commitment of \$2.40/hour

You could save an estimated \$1,173 monthly by purchasing the recommended Compute Savings Plan.

Based on your past **60 days** of usage, we recommend purchasing a Savings Plan with a commitment of **\$2.40/hour** for a **3-year term**. With this commitment, we project that you could save an average of **\$1.61/hour** - representing a **40%** savings compared to On-Demand. To account for variable usage patterns, this recommendation maximizes your savings by leaving an average **\$0.04/hour** of On-Demand spend.

Before recommended purchase	After recommended purchase (based on your past 60 days of usage)	
Monthly On-Demand spend ⓘ \$2,955 (\$4.05/hour) Based on your On-Demand spend over the past 60 days	Estimated monthly spend ⓘ \$1,782 (\$2.44/hour) Your recommended \$2.40/hour Savings Plans commitment + an average \$0.04/hour of On-Demand spend	Estimated monthly savings ⓘ \$1,173 (\$1.61/hour) 40% monthly savings over On-Demand \$2,955 - \$1,782 = \$1,173

This recommendation examines your usage over the past 60 days (including your existing Savings Plans and EC2 Reserved Instances) and calculates what your costs would have been had you purchased the recommended Savings Plans. See applicable rates for Savings Plans [here](#). To generate this recommendation, AWS simulates your bill for different commitment amounts and recommends the commitment amount that provides the greatest estimated savings. [Learn more](#)

Recommended Compute Savings Plans

Download CSV

Add selected Savings Plan(s) to cart

×	Term	Payment option	Recommended commitment	Estimated hourly savings ⬆
☒	3-year	All upfront	\$2.40/hour	\$1.61 (40%)

*Average hourly spend and minimum hourly spend based on your current on-demand spend for the given instance family.

sundog-education.com

datacumulus.com

© 2024-2026 All Rights Reserved Worldwide

 DataCumulus

 Sundog
Education

Cost Explorer – Forecast Usage



Trusted Advisor



- No need to install anything – high level AWS account assessment
- Analyze your AWS accounts and provides recommendation on 6 categories:
 - Cost optimization
 - Performance
 - Security
 - Fault tolerance
 - Service limits
 - Operational Excellence
- **Business & Enterprise Support plan**
 - Full Set of Checks
 - Programmatic Access using AWS Support API

Checks	
▶	Amazon EBS Public Snapshots Checks the permission settings for your Amazon Elastic Block Store (EBS) snapshots. 0 EBS snapshots are marked as public.
▶	Amazon RDS Public Snapshots Checks the permission settings for your Amazon Relational Database Service (RDS) snapshots. 0 RDS snapshots are marked as public.
▶	IAM Use This check is intended to discourage the use of root access. At least one IAM user has been created for this account.

ML and AI System Architecture

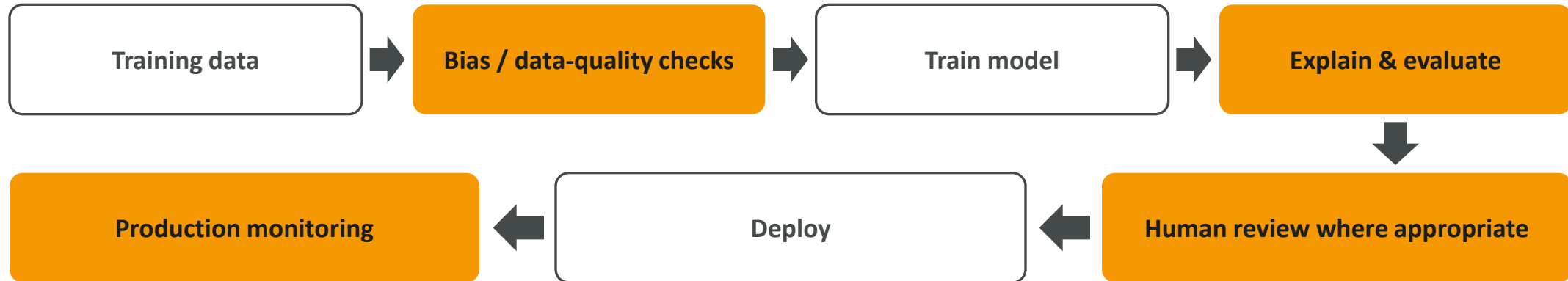
Best practices for designing machine learning systems

Responsible AI: Core Dimensions

- Fairness
- Explainability
- Privacy and Security
- Safety
- Controllability
- Veracity and Robustness
- Governance
- Transparency



Responsible AI for Traditional ML



- **Bias & fairness metrics** — standardized metrics quantify disparities in datasets and model outcomes (the individual metrics are covered elsewhere in this course)
- **SHAP** — explains individual or aggregate predictions using feature-attribution values
- **SageMaker AI MLflow** — tracks experiment / model metrics and artifacts for reproducibility, comparison, and auditability
- **Human review** — a general workflow pattern when consequences, uncertainty, or risk justify it (not tied to any single service)
- **Production monitoring** — CloudWatch, QuickSight, and AWS-published monitoring solutions

Exam tip: SageMaker Clarify, Model Monitor, A2I & Role Manager are in maintenance mode — AWS now recommends the approaches shown here.

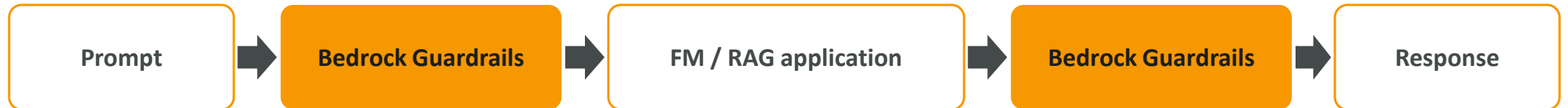
Responsible AI for Generative AI

EVALUATION — measure model & application behavior



Automated metrics, human evaluation, or LLM-as-a-judge — assess correctness, relevance, safety, and bias for your workload

RUNTIME SAFEGUARDS — enforce safety & policy on every request



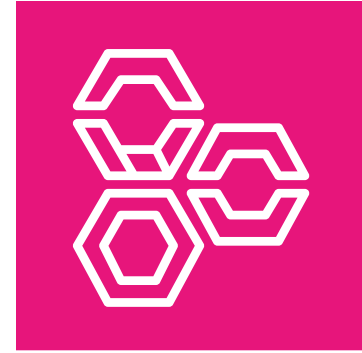
Content filters, denied topics, sensitive-information protection & other safety controls — applied in the app flow, independent of evaluation

Evaluation measures behavior. Guardrails enforce runtime policy. Responsible AI needs both — one does not replace the other.

Keep monitoring in production: quality, safety & bias shift as models, prompts, retrieved data, user behavior, or app context change. Amazon Macie can discover sensitive data in S3.

AWS Well-Architected Machine Learning (ML) Lens

- The principles of this section are embodied in a “custom lens” for the AWS Well-Architected Tool
- This is, in practice, a JSON file that leads you through an interview about how you are using ML
 - And guides you on how you might do it better.
- Associated white paper:
 - <https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/machine-learning-lens.html>



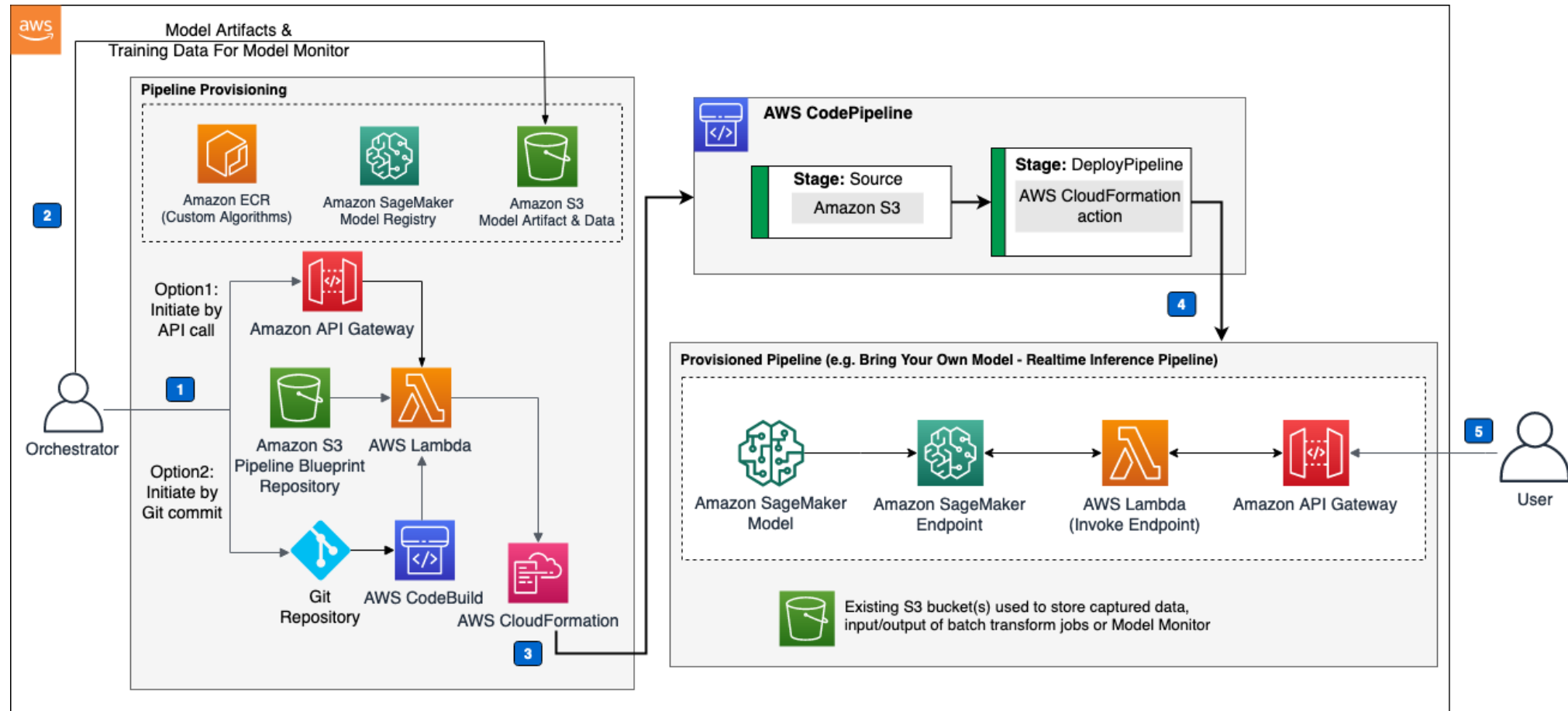
AWS Well-Architected Tool

MLOps Workload Orchestrator on AWS



- From AWS Solutions Library
- CloudFormation template
- Can set up training, inference, monitoring
- Currently 12 pipelines to pick from
 - Training with SageMaker and hyperparameter tuning or autopilot
 - “Bring-your-own model” (BYOM) real-time or batch inference with SageMaker
 - Custom algorithm pipeline for your own Docker / ECR images
 - Model monitor pipelines on inference for data quality, model quality, bias, and explainability

MLOps Workload Orchestrator



Exam Preparation

More Prep Resources

AWS Skill Builder

- The free resources are likely sufficient
 - “Standard Exam Prep Plan” includes Amazon’s own training resources
 - “Official Practice Question Set” – 20-question practice test
- <https://skillbuilder.aws/category/exam-prep/machine-learning-engineer-associate-MLA-C02>

The screenshot shows the AWS Skill Builder website for the AWS Certified Machine Learning Engineer - Associate exam. The page features a purple header with the AWS logo and navigation links. A green banner at the top promotes a subscription offer. The main content area includes a section for the exam prep plan, a sidebar with a subscription offer, and a table of courses.

EXAM PREP
AWS Certified Machine Learning Engineer - Associate

Guidance from start to AWS Certified Machine Learning Engineer - Associate. Get the most out of your study time with AWS Skill Builder's four steps through exam prep, for seamless learning when and where you need it. This exam validates technical ability to implement ML workloads in production and operationalizing them. Boost your career profile and credibility, and position yourself for in-demand machine learning job roles.

[Download the exam guide](#)

Get hands-on with an Individual Subscription

Monthly Plan **Annual Plan**

\$29/month

1,000+ lab experiences, enhanced exam prep, solution building, and more

[Subscribe now](#)

Step 1: Get to know the exam with exam-style questions
Review the exam guide. Take the AWS Certification Official Practice Question Set to understand exam-style questions.

Course	Description	Type	Cost	Duration
Standard Exam Prep Plan: AWS Certified Machine Learning Engineer - Associate (MLA-C01 - English)	Fundamental The Standard Exam Prep Plan includes free resources for Steps 1–3 of the 4-step plan created by AWS experts. Learners will be enrolled in all courses included in the Exam Prep Plan. We recommend that you select and work through the exam prep content that meets your specific skill and knowledge needs. Then, you can progress through the material at your own pace.	Exam Preparation	Free	7 hours 30 minutes
Enhanced Exam Prep Plan: AWS Certified Machine Learning Engineer - Associate (MLA-C01 - English)	Fundamental The Enhanced Exam Prep Plan includes resources for each step of the 4-step plan created by AWS experts. Learners have access to everything in the Standard Exam Prep plan plus access to more robust role-based training, including game-based learning and experiential learning. Learners will be enrolled in all courses.	Exam Preparation	Paid	41 hours 15 minutes

Practice Exams (including Amazon's)

aws

AWS Certified Machine Learning Engineer - Associate Official Practice Question Set (MLA-C01)

Dashboard Notes Flags [Go to AWS Skill Builder](#)

Dashboard

0.0% Complete

N/A

0 of 1 Products Taken

00:00:00 Avg. Answer Time

00:00:00 Avg. Correct Answer Time

00:00:00 Avg. Incorrect Answer Time

Products Reports

Product Name	Product Length	Product Time	% Correct	Status
AWS Certified Machine Learning Engineer - Associate Official Practice Question Set (MLA-C01 - English)	20 Questions	Unlimited		See more

Previous Score Reports

Product Name	Questions Taken	% Correct	Time Elapsed	Attempt Date	Product State
--------------	-----------------	-----------	--------------	--------------	---------------

SageMaker AI Developer Guide

<https://docs.aws.amazon.com/sagemaker/latest/dg/>

The screenshot shows the AWS SageMaker AI Developer Guide page. The top navigation bar includes the AWS logo, social media icons, a language selector set to 'English', and a 'Sign In to the Console' button. The left sidebar contains the 'Amazon SageMaker Developer Guide' title, a search bar, and a list of topics: 'What Is Amazon SageMaker?', 'How It Works', 'Set Up Amazon SageMaker', 'Get Started', 'Using Notebook Instances', 'Build a Model', 'Train a Model', 'Deploy a Model', 'Use Your Own Algorithms or Models', 'Amazon SageMaker in AWS Marketplace', 'Manage Machine Learning Experiments', 'Use Machine Learning Frameworks with Amazon SageMaker', 'Reinforcement Learning with Amazon SageMaker RL', 'Monitoring', 'Security', 'Amazon SageMaker Ground Truth', and 'Limits and Supported Regions'. The main content area features a breadcrumb trail 'AWS Documentation » Amazon SageMaker » Developer Guide » What Is Amazon SageMaker?'. A message box states: 'The AWS Documentation website is getting a new look! Try it now and let us know what you think. [Switch to the new look >>](#) You can return to the original look by selecting English in the language selector above.' Below this is the section 'What Is Amazon SageMaker?' with a paragraph describing the service. A 'Are You a First-time User of Amazon SageMaker?' section follows, recommending four steps: 1. Read 'How Amazon SageMaker Works', 2. Read 'Get Started', 3. Try a model training exercise, and 4. Explore other topics. The footer contains 'Terms of Use | Privacy | © 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.', a link to 'Have a question? Try the Forums.', a 'Did this page help you?' poll with 'Yes' and 'No' options, and a 'Feedback' button.

aws

English Sign In to the Console

Amazon SageMaker
Developer Guide

Documentation - This Guide

Search

☐ What Is Amazon SageMaker?

☒ How It Works

☒ Set Up Amazon SageMaker

☒ Get Started

☒ Using Notebook Instances

☒ Build a Model

☒ Train a Model

☒ Deploy a Model

☒ Use Your Own Algorithms or Models

☒ Amazon SageMaker in AWS Marketplace

☐ Manage Machine Learning Experiments

☒ Use Machine Learning Frameworks with Amazon SageMaker

☒ Reinforcement Learning with Amazon SageMaker RL

☒ Monitoring

☒ Security

☒ Amazon SageMaker Ground Truth

☐ Limits and Supported Regions

AWS Documentation » Amazon SageMaker » Developer Guide » What Is Amazon SageMaker?

The AWS Documentation website is getting a new look!
Try it now and let us know what you think. [Switch to the new look >>](#)
You can return to the original look by selecting English in the language selector above.

On this page:

[Are You a First-time User of Amazon SageMaker?](#)

What Is Amazon SageMaker?

Amazon SageMaker is a fully managed machine learning service. With Amazon SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment. It provides an integrated Jupyter authoring notebook instance for easy access to your data sources for exploration and analysis, so you don't have to manage servers. It also provides common machine learning algorithms that are optimized to run efficiently against extremely large data in a distributed environment. With native support for bring-your-own-algorithms and frameworks, Amazon SageMaker offers flexible distributed training options that adjust to your specific workflows. Deploy a model into a secure and scalable environment by launching it with a single click from the Amazon SageMaker console. Training and hosting are billed by minutes of usage, with no minimum fees and no upfront commitments.

This is a HIPAA Eligible Service. For more information about AWS, U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA), and using AWS services to process, store, and transmit protected health information (PHI), see [HIPAA Overview](#).

Are You a First-time User of Amazon SageMaker?

If you are a first-time user of Amazon SageMaker, we recommend that you do the following:

1. Read [How Amazon SageMaker Works](#) – This section provides an overview of Amazon SageMaker, explains key concepts, and describes the core components involved in building AI solutions with Amazon SageMaker. We recommend that you read this topic in the order presented.
2. Read [Get Started](#) – This section explains how to set up your account and create your first Amazon SageMaker notebook instance.
3. Try a [model training exercise](#) – This exercise walks you through training your first model. You use training algorithms provided by Amazon SageMaker. For more information, see [Get Started](#).
4. Explore other topics – Depending on your needs, do the following:

Terms of Use | Privacy | © 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Have a question? Try the Forums.

Did this page help you? Yes No

Feedback

Amazon's Exam Guide

<https://docs.aws.amazon.com/aws-certification/latest/machine-learning-engineer-associate-02/machine-learning-engineer-associate-02.html>

AWS Certified Machine Learning Engineer - Associate (MLA-C02)

The AWS Certified Machine Learning Engineer - Associate (MLA-C02) exam validates a candidate's ability to build, operationalize, deploy, and maintain AI and ML solutions and pipelines by using the AWS Cloud. The exam validates ML engineering skills and the ability to work with traditional ML models and foundation models (FMs).

Topics

- [Introduction](#)
- [Target candidate description](#)
- [Exam content](#)
- [Content outline](#)
- [Content Domain 1: Data Preparation for ML and AI](#)
- [Content Domain 2: ML Model and Foundation Model \(FM\) Development](#)
- [Content Domain 3: Deployment and Orchestration of ML and AI Workflows](#)
- [Content Domain 4: Operating, Monitoring, and Securing ML and AI Solutions](#)
- [Mentions of AWS services on the exam](#)
- [In-scope AWS services and features](#)
- [Out-of-scope AWS services and features](#)
- [Comparison of MLA-C01 and MLA-C02](#)
- [Survey](#)

Introduction

The AWS Certified Machine Learning Engineer - Associate (MLA-C02) exam validates a candidate's ability to build, operationalize, deploy, and maintain AI and ML solutions and pipelines by using the AWS Cloud.

The exam validates ML engineering skills and the ability to work with traditional ML models and foundation models (FMs).

Introduction

1

Consider taking the Certified AI Practitioner exam first



What to Expect

Sitting for the exam

- ~2 hours (130 minutes) (Beta is 170)
 - But you'll probably only need a little over 2.
 - 65 questions (85 during beta)
 - TAKE THE TIME TO FULLY UNDERSTAND THEM
- Use the flag capability
 - You'll have time to go back and double check things.
- You can't bring anything into the room
 - You'll be given note paper or dry-erase board, earplugs, and that's it.
- Schedule smartly
 - As soon as possible after you've mastered the practice exams
 - Choose a time of day that matches your highest energy level – you need stamina
 - Remember different testing providers may have different availability



Prepare

- Use the bathroom before you head in
- Eat something (but not right beforehand)
- Get a good night's sleep
- Review things you need to memorize before going in
 - Recall, Precision, F1
 - Regularization techniques
- Be alert
 - Whatever works for you...
- Make sure you're ready
 - You've got \$300 on the line
 - Take the practice exams! Repeatedly!



Strategies

- Don't rush
 - For each question, take the time to understand:
 - What you are optimizing for
 - Requirements
 - The system as a whole
- Your pace should be about 2 – 2 ½ minutes per question
 - If you're taking longer, make your best guess, flag it and come back to it later
- Use the process of elimination
 - Even if you don't know the correct answer, you can often eliminate several.
- Keep calm
 - You're not going to get 100%. You don't have to.



Congratulations