CLOUD DATA PROCESSING

WPROWADZENIE
DO PRZETWARZANIA DANYCH W CHMURZE

Jakub Kasprzak
UAM, Big Data 2022

WPROWADZENIE DO PRZETWARZANIA DANYCH W CHMURZE
PLAN PRZEDMIOTU

1. Wprowadzenie do koncepcji „Jeziora Danych” (Data Lake)
2. Składowanie danych w data lake
3. Zasilanie jeziora danych
4. Terraform 101 IaaC
5. Budowanie systemów w oparciu o usługi zarządzane (Serverless)
6. Przetwarzanie danych w chmurze

WPROWADZENIE DO PRZETWARZANIA DANYCH W CHMURZE
PLAN GRY

- Wykład o Data Lake + demonstracje najważniejszych serwisów związanych z przetwarzaniem danych w chmurze AWS
- Mikroprojekt – ładowanie danych do Data Lake, organizacja warstw, tworzenie obiektów IaaC (Terraform) i podstawy przetwarzania danych (batch & real-time)

Zaliczenie

- udostępnienie kodów źródłowych z rozwiązaniami zadań / problemów (Terraform)
- Obecność + aktywność na zajęciach

python HashiCorp Terraform aws

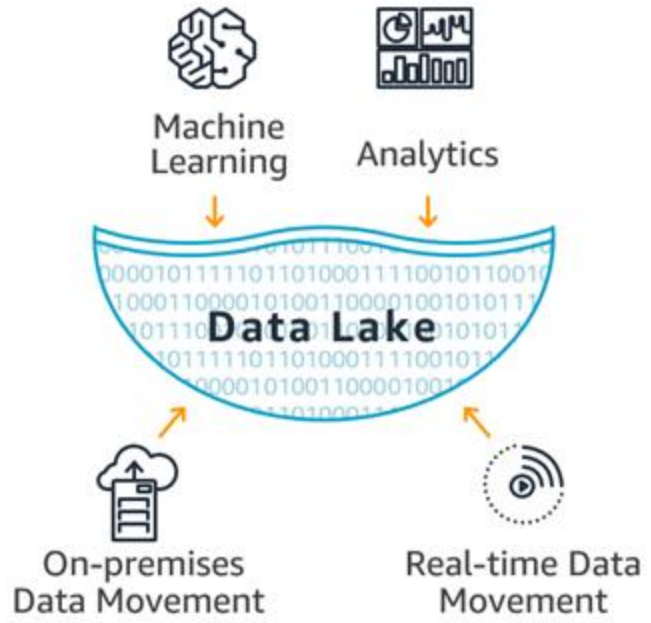
MODUŁ 1
WPROWADZENIE DO JEZIOR DANYCH

- Czym są jeziora danych ?
- Czym się różnią od hurtowni ?
- Jakie problemy rozwiązują ?
- Jak wyglądają jeziora danych?

DATA LAKES
CZYM SĄ JEZIORA DANYCH

DATA LAKES
CZYM SĄ JEZIORA DANYCH

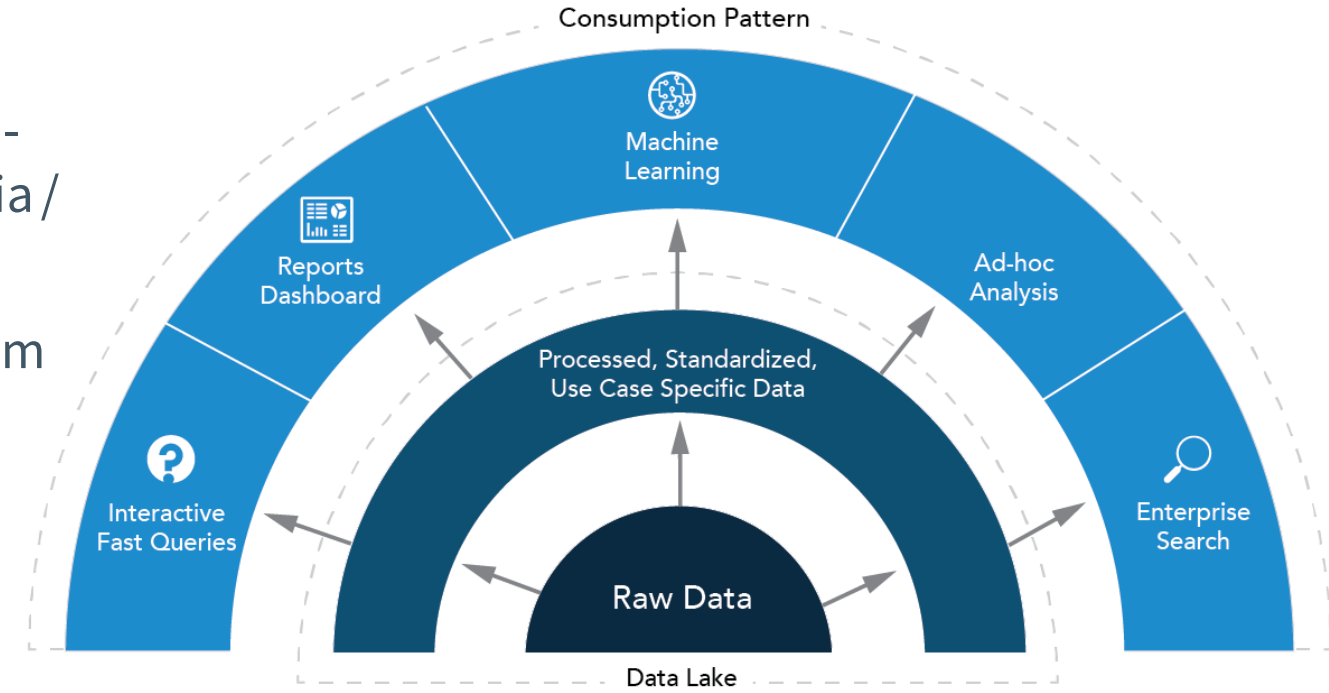
- **Przestrzeń** do składowania NIEPRZETWORZONYCH danych, **w oryginalnym formacie**
- Związane z pamięciami masowymi (Hadoop, object stores)
- Zintegrowane z mechanizmami ładowania danych, opisywania ich metadanymi oraz analityką
- Upraszczają klasyczne podejście do budowania systemów, nowoczesnych hurtowni, procesuów ETL/ETL. Otwierają drzwi do praktycznie nieograniczonej analityki danych
- Adresują najważniejsze założenia przetwarzania Big Data. Dowolne, jeszcze nieznane (przyszłe) potrzeby przetwarzania danych
- DL nie zastępuje DWH



Source: AWS

CZYM JEST DATA LAKE
GŁÓWNE ZAŁOŻENIA

- Separacja warstwy składowania danych od mocy obliczeniowej (niezależne skalowanie)
- Szybkość zmian i dostarczania danych - bycie gotowym na nieznane wymagania / potrzeby
- Self-service – biznes powienien móc sam odpowiadać na własne potrzeby, dostarczać rozwiązania bez zewnętrznych zależności
- Dostępność danych, transparentność (uwaga na security), wychodzenie z silosów



Source : Cloud Technologogy Partners

DATA WAREHOUSES VS DATA LAKES
PORÓWNANIE

DATA LAKE

- Cała domena biznesowa – wszystkie dane
- Schema-on-read (definiowana na podstawie przypadku użycia)
- ELT
- Efektywny kosztowo dla BigData, skalowalny
- Rozdzielenie warstwy obliczeniowej od pamięci masowej (skalowalność)
- Data scientists, inżynierowie

DATA WAREHOUSE

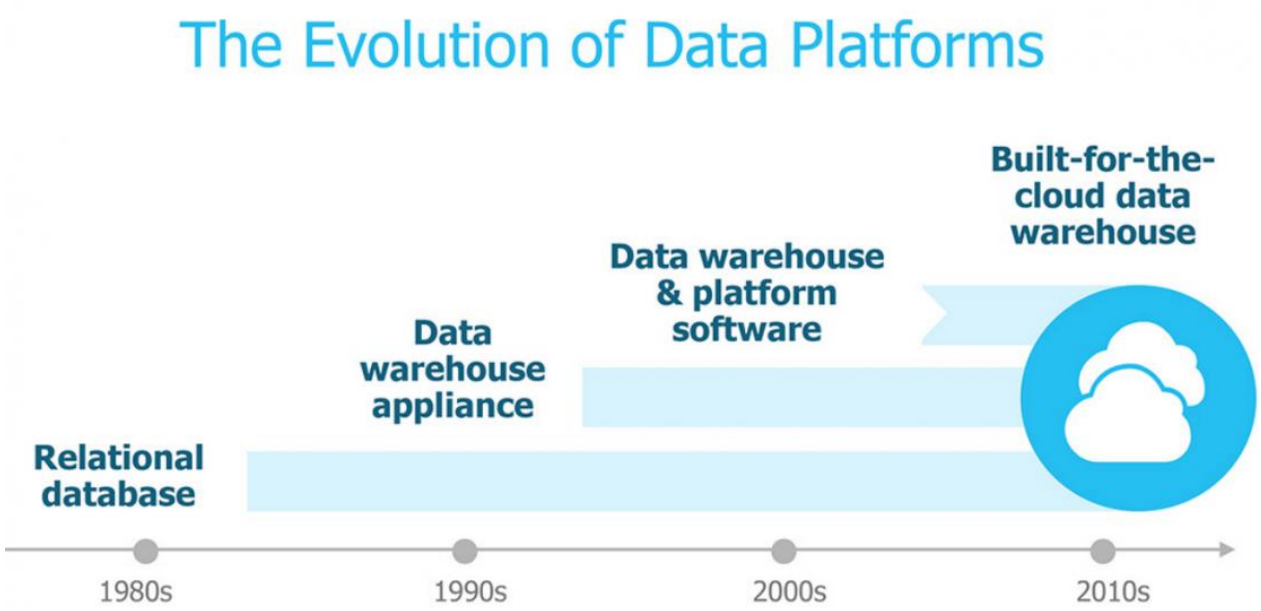
- Ustrukturyzowane, przetworzone, wybrane obszary
- Schema-on-write (wymaganie muszą być znane z góry, schemat zdefiniowany)
- ETL
- Koszt rośnie wraz z danymi
- Spójność danych, ściśle ustalony schemat
- Dobrze znane (30 lat ?)
- Analitycy biznesowi

EWOLUCJA SYSTEMÓW DO PRZTWARZANIA DANYCH
OD HURTOWNI DO DATA LAKE

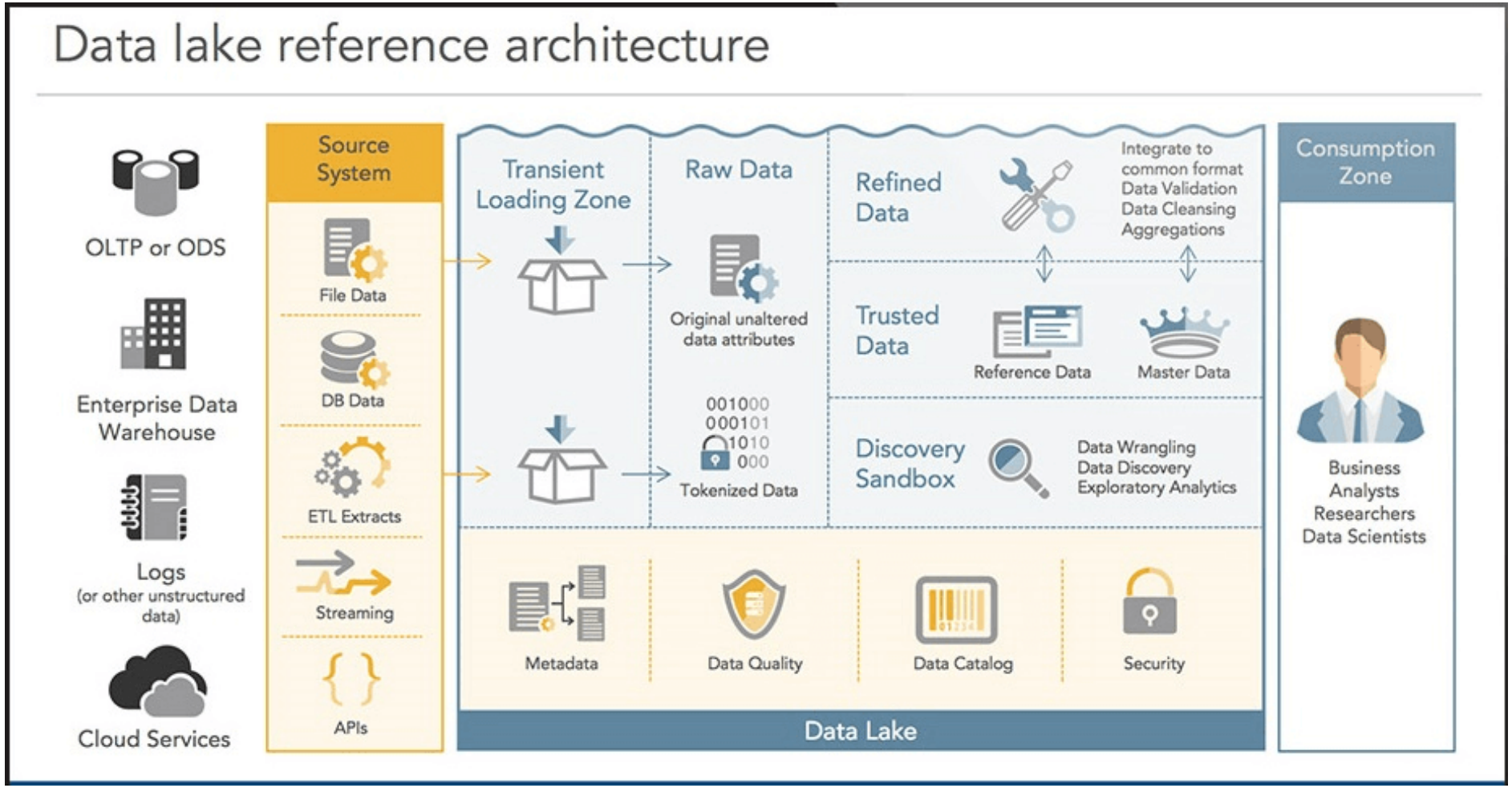
Hurtownie danych
- Inmon > Kimbal > Data Vault
- Cloud data warehouses (Redshift, Snowflake)

Specjalizowane rozwiązania (appliances)
- Netezza, Teradata, Exadata, Vertica

Rozwiąznania oparte o Hadoop / Data lake
- HDFS, Object store (S3, Azure Blob storage, GCS)



JAK WYGLĄDA DATA LAKE ?
PRZYKŁAD ARCHITEKTURY



Source : Zaloni reference architecture https://www.zaloni.com/

MODUŁ 2
SKŁADOWANIE DANYCH

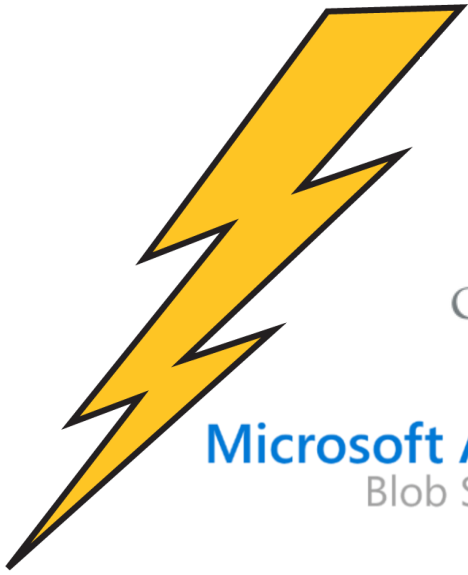
- Budowa jeziora danych - warstwy
- HDFS vs object store (S3)
- Formaty danych
- Partycjonowanie i organizacja danych w DL
- Bezpieczeństwo, tokenizacja danych PII, wyzwania

BUDOWA DATA LAKE
WARSTWY

- **RAW zone** (ingestion / landing zone)
  Tu lądują nasze dane w niezmienionej formie, często z PII. Główny cel – złapać dane tak szybko jak to możliwe w oryginalnej postaci (bez transformacji) Użytkownicy nie mają dostępu do tego miejsca (analogia do staging zone w DWH)
- **PROCESSED zone** (Standardized, Cleansed, Refined, Curated)
  Single source of truth - może składać się z kilku warstw. Kształt danych i ich przeznaczenie jest już znane – nie ma tu przypadkowych danych (vs RAW)
- **APPLICATION zone** (Trusted, Production, Master Data, Business)
  Single verision of truth – główne źródło danych dla użytkowników końcowych, wartswy raportującej, aplikacji.
- Inne (opcjonalne) : **DATA MARTS, SANDBOX**

SKŁADOWANIE DANYCH
HDFS VS OBJECT STORE

- HDFS (2000's) distributed file system - bazuje na file / block store
- Object store (S3, Azure Blob Storage, Google Cloud Storage)
- Składowanie plików (hierarchia) vs składowanie obiektów (skalowalność metadanych)
- Skalowalność object store (w teorii nieskończona),
- Cena 3-5-10 (?) x niższa od HDFS
- Trwałość object store : S3 99.999999999% vs HDFS 99.9999 % (szacunkowa)

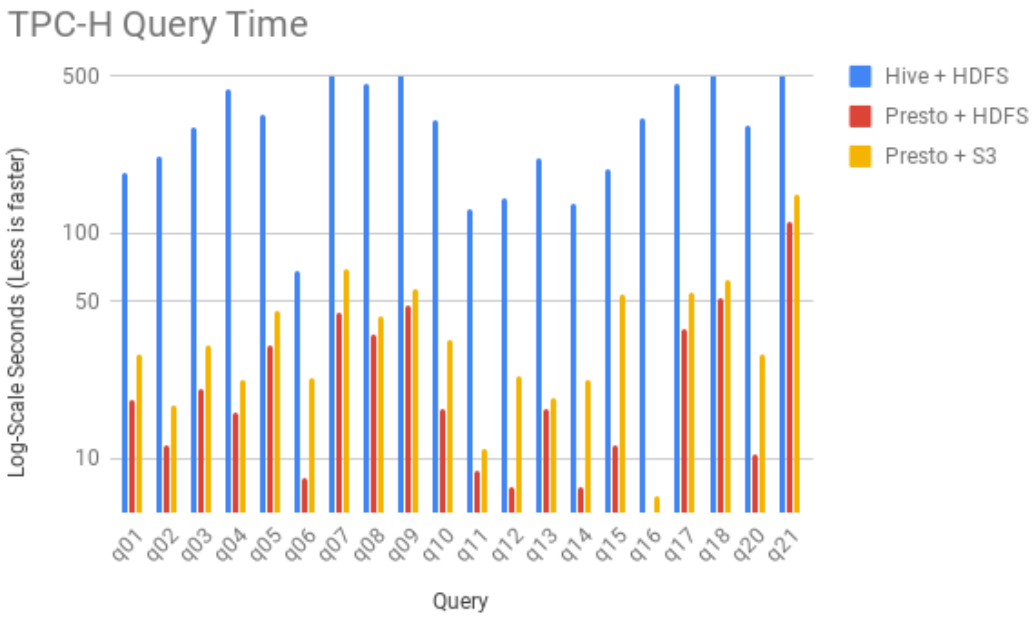


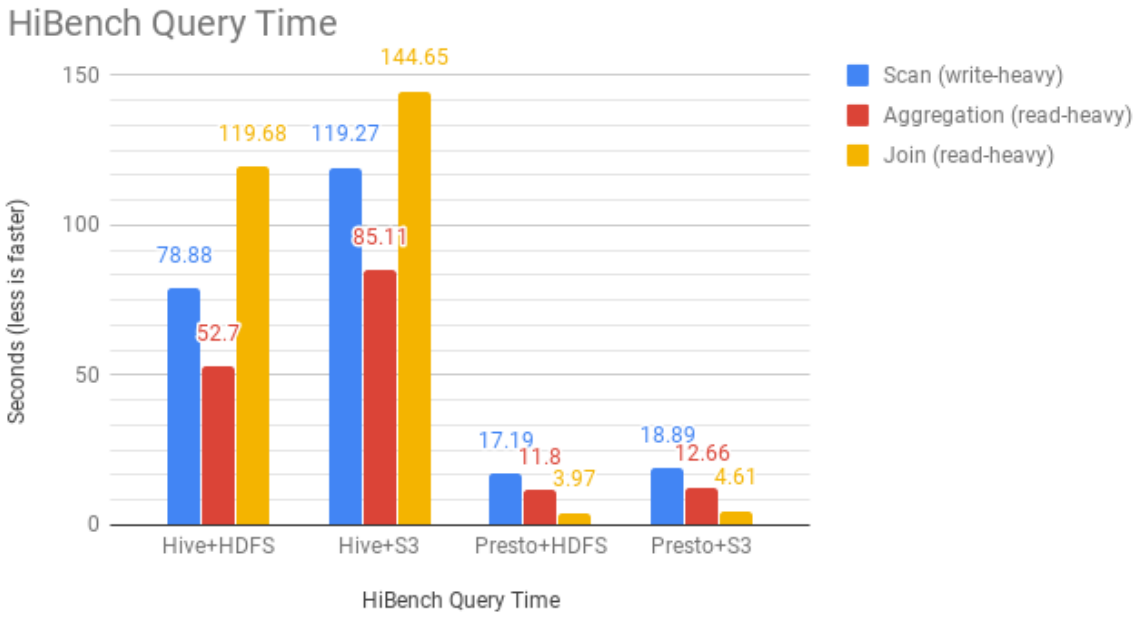
Google Cloud Storage

Microsoft Azure
Blob Storage

HDFS VS OBJECT STORE
WYDAJNOŚĆ

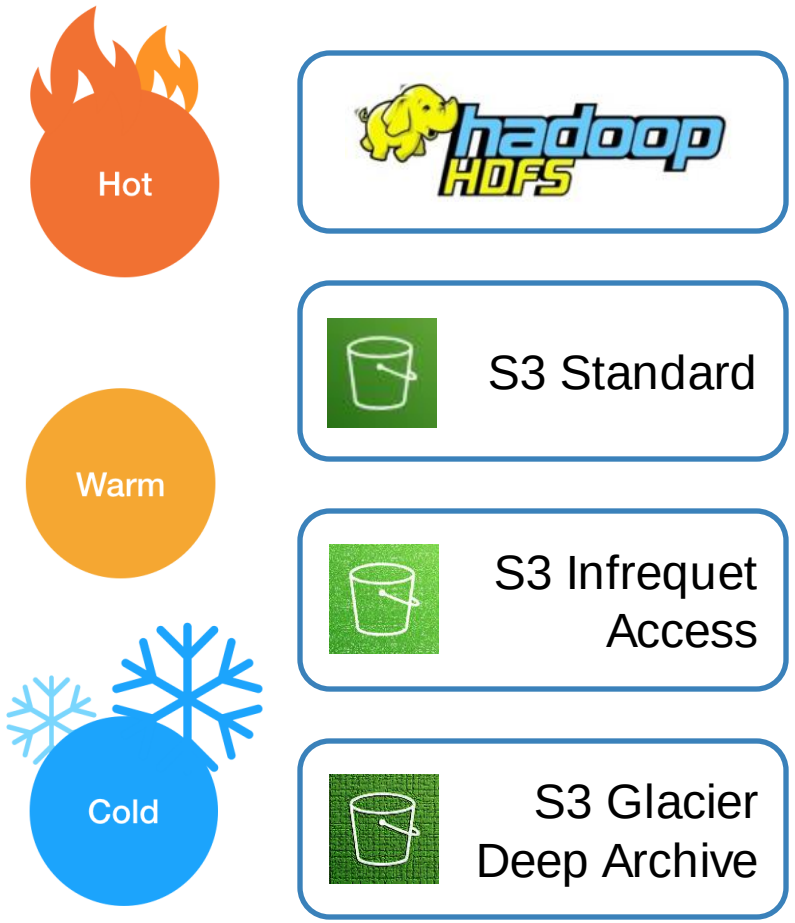
- TPC-H (866 mln rek ~100GB)
- HiBench (Hive-QL) 11 mln rek. (~1.8 GB)





Source : Cloudian.com, Datanami.com [Can On-Prem S3 Compete with HDFS for Analytic Workloads?]

ROZWIĄZANIA HYBRYDOWE HDFS + OBJECT STORE
TEMPERATURA DANYCH - WARSTWY



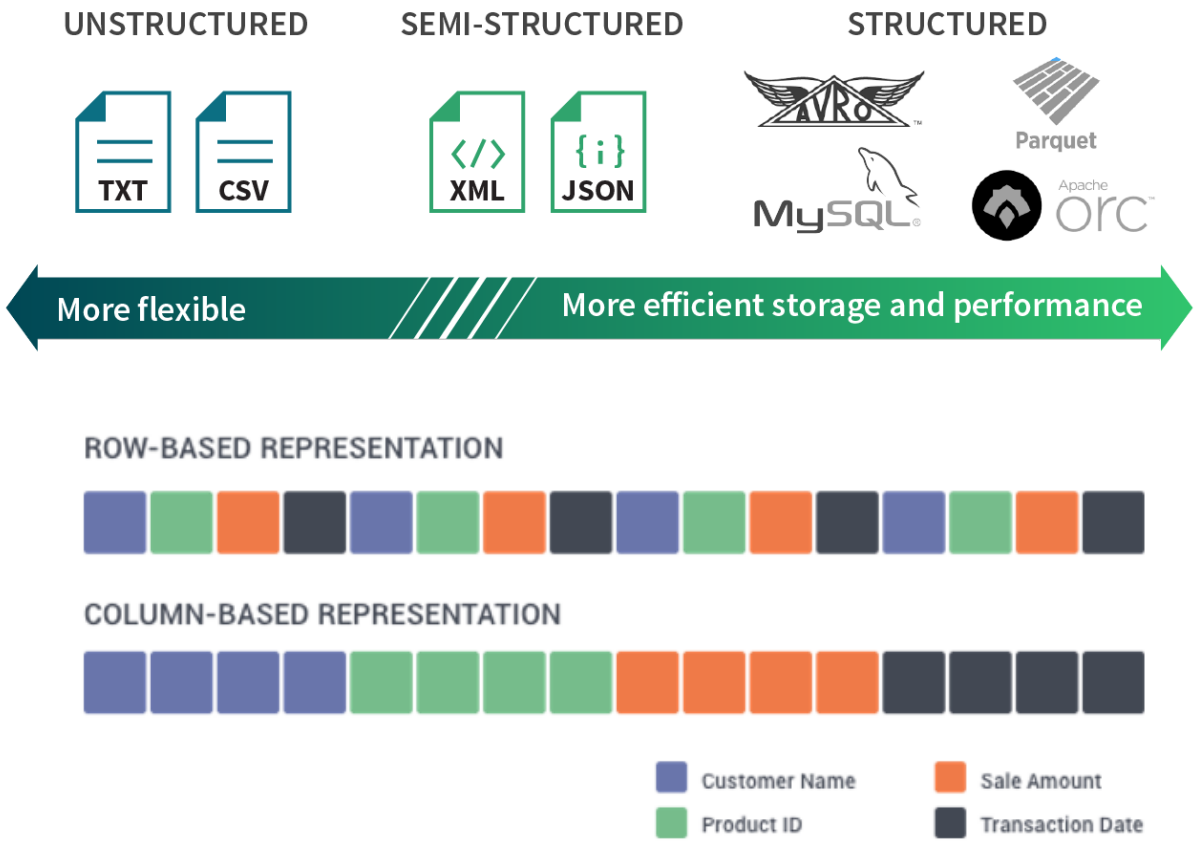
- Dane gorące
  Najświeższe / najważniejsze dane (często odpytywane) na **Hadoop / EMR** z lokalnym HDFS danych (+ in-memory computing). Wysoka wydajność = wysoki koszt
- Dane ciepłe / chłodne
  dedykowane warstwy **object store** np. S3 (Standard, Infrequent Access) – capacity optimized
- Zimne dane
  archiwum (Glacier, Deep Archive)
- Duża złożoność systemu
  Wszystkie benefity i wady obydwu rozwiązań), wysoki próg wejścia (wiedza, specjaliści) i utrzymania

SKŁADOWANIE DANYCH
ISTOTNE CZYNNIKI

- Rozmiary plików
  HDFS domyślny rozmiar 128 MB, wszystko co mniejsze powinno być postrzegane jako małe (API calls)
- Formaty danych
  Różne wymagania w zależności od przeznaczenia - szybki zapis / odczyt / przesyłanie. AVRO, PARQUET, JSON etc.
- Kompresja
  Snappy / LZO / GZIP – kompromis pomiędzy efektywnością a funkcjonalnością. Co to znaczny że pliki są splittable ?
- Bezpieczeństwo, kontrola dostępu, szyfrowanie
- Utrzymanie (automatyczna archiwizacja danych)

SKŁADOWANIE DANYCH
WYBÓR ODPOWIEDNIEGO FORMATU

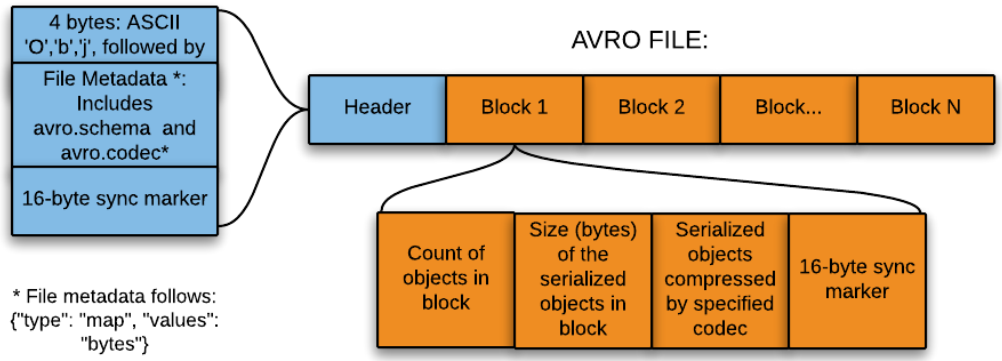
- Wybór **formatu** bazuje na sposobie w jaki dane są przetwarzane
- Organizacja **wierszowa vs kolumnowa**
  Czy potrzebujemy czytać całe rekordy czy analizować wybrane atrybuty ?
- **Ewolucja schematów**
- Podzielność plików (**splitability**)
- Możemy przechowywać dane w różnych formatach dla różnych zastosowań - osobne warstwy jeziora.
  Uwaga na dodatkowy narzut – konwersja typów, modyfikowania schematów, łączenia plików



Souce : https://databricks.com/

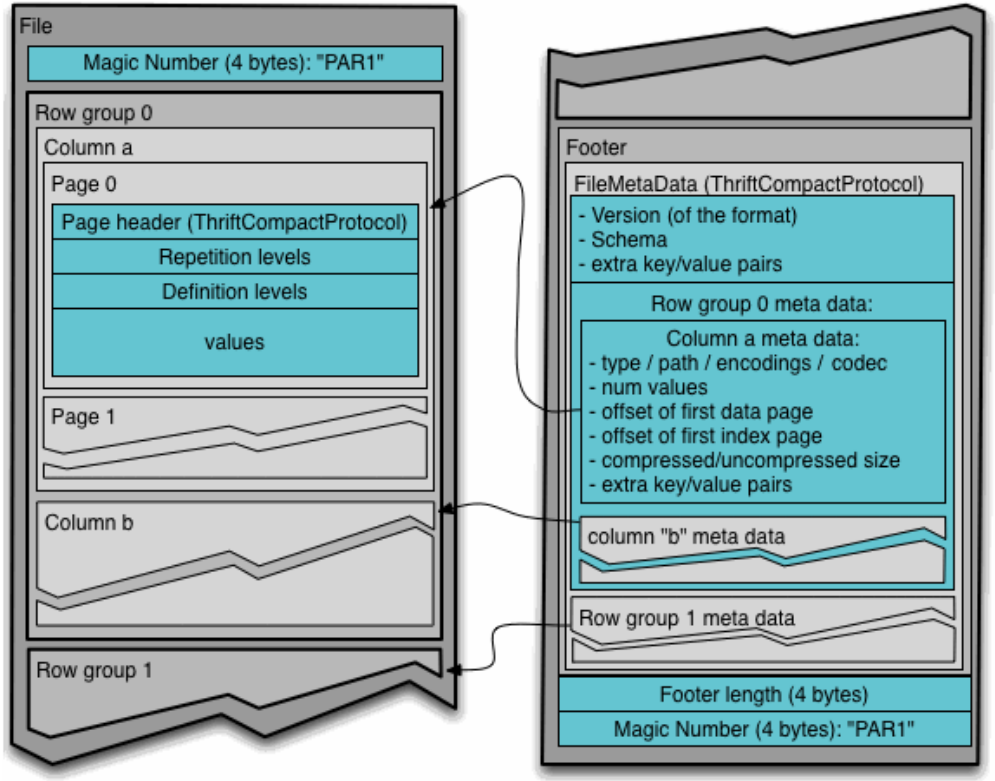
FORMATY DANYCH
AVRO

- Row based format – czytamy zawsze cały obiekt (rekord), snappy compression
- Ściśle zdefiniowany schemat
- Pełne wsparcie dla schematów zagnieżdżonych, złożonych (mapy, listy, obiekty)
- Standard wspiera **ewolucje schematów**
  Backward / Forward / Full transitive compatibility
- Powszechnie używany w systemach opartych o **Kafkę** (+ Schema Registry)
- Dobry do landing zone z uwagi na brak konieczności konwersji typów danych



FORMATY DANYCH
PARQUET, ORC

- Formaty składowania danych kolumnowo
- Składowanie kolumnowe = efektywna kompresja, optymalizacja kosztów - szczególnie w chmurze publicznej !
- Idealny do analityki, czytamy tylko te atrybuty które chcemy
- Uwaga na typy danych i konieczność konwersji pomiędzy formatami (szczególnie w raw zone)
- Schemat dołączonny do danych (self-describing)
- Wspiera mechnizmy predicate-pushdown

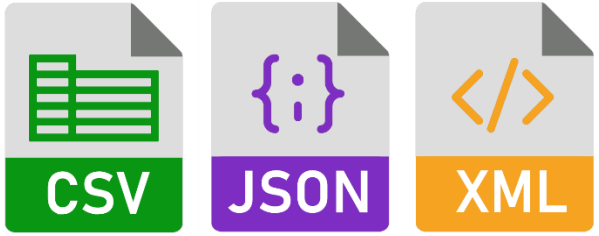


Parquet

Apache orc

FORMATY DANYCH
CSV, JSON, XML

- Formaty unstructured / semi-structured - wygodne dla użytkowników końcowych – łatwość analizy w dowolnych narzędziach bez konieczności używania dodatkowych bibliotek
- **CSV** – dobry do przesyłania danych między systemami w postaci wierszowej, niski narzut na dodatkowe metadane (co jest zaletą i wadą). Prosty, płaski schemat. Nie ma standardowego rozwiązania na przesyłanie danych binarnych. Uwaga na separatory.
- **JSON** – wspiera struktury złożone (mapy, listy, zagnieżdżenia). Większość języków ma parsery. Możliwość definiowania i parsowania schematu
- **XML** - narzut na procesowanie / parsowanie, duża nadmiarowość formatu. Historycznie b. popularny obecnie żadko wybierany z uwagi na dużyt narzut w przetwarzaniu i złożoność



SKŁADOWANIE DANYCH
PARTYCJONOWANIE, ORGANIZACJA DANYCH

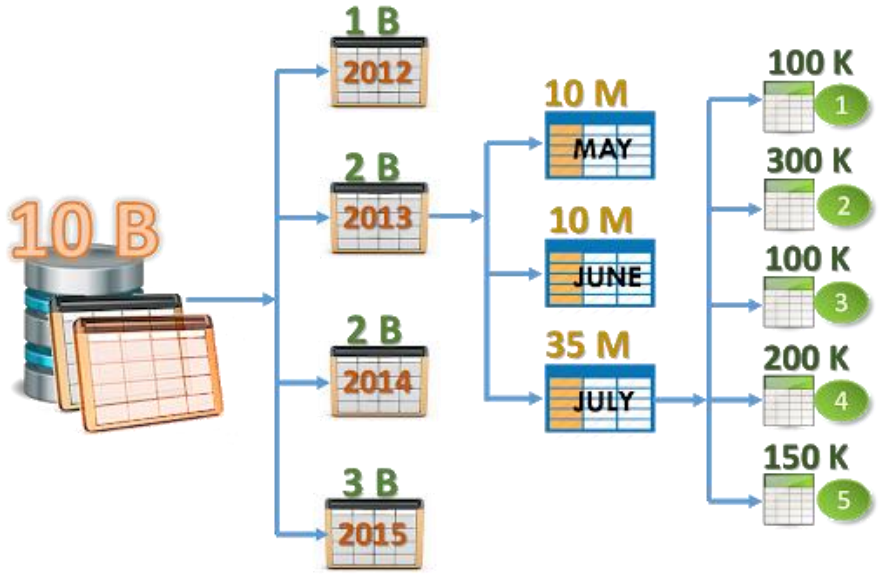
- Przykład 1 : zasilanie strumieniowe z Kafki
  Kafka + AVRO format jako źródło, wsparcie schema evolution, separacja wiadomości z kafki per topic, składowanie danych w godzinnych partycjach

  s3://bucket-name/{kafka-topic-name}/{full-event-name}/year={year}/ month={month}/day={day}/hour={hour}/sub_version={sub_version}/

  s3://RAW/test-topic/test.event.v1/**year**=2020/**month**=06/ **day**=01/**hour**=09/sub_**version**=1.0.1/file.avro

- Przykład 2 : zasilanie batchowe
  Pliki CSV/JSON, separacja po źródle, wersjonowanie

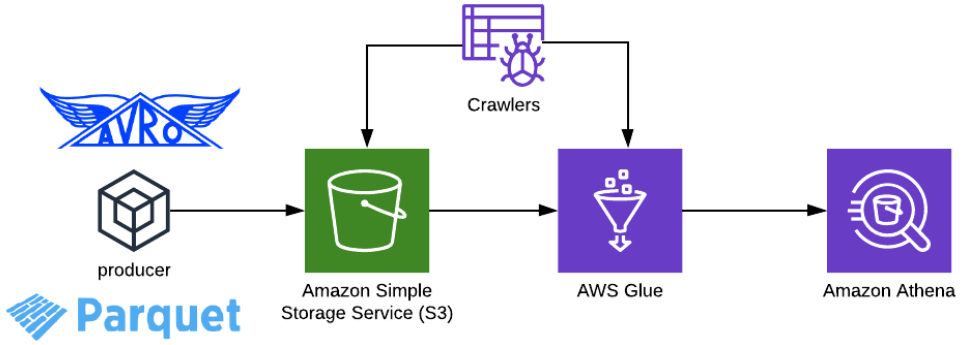
  s3://BC/{domena}/{source-name}/year={year}/ month={month}/ day={day}/hour={hour}/version={version}



Souce : http://www.hadooptechs.com/hive/dynamic-partition-in-hive

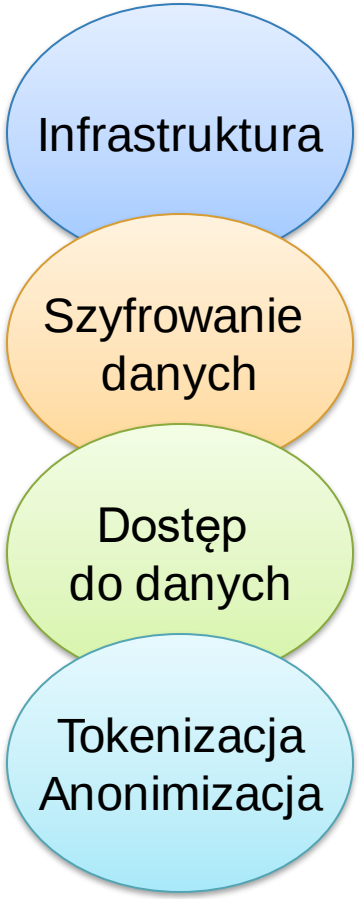
AVRO / PARQUET
DEMO

- Demonstracja formatów danych AVRO / PARQUET
- Omówienie schematów AVRO
- Porównanie wydajności
- Wstęp do analizy danych w AWS
- Wstęp do Glue Data Catalogue
- Wstęp do Athena (serverless Presto)



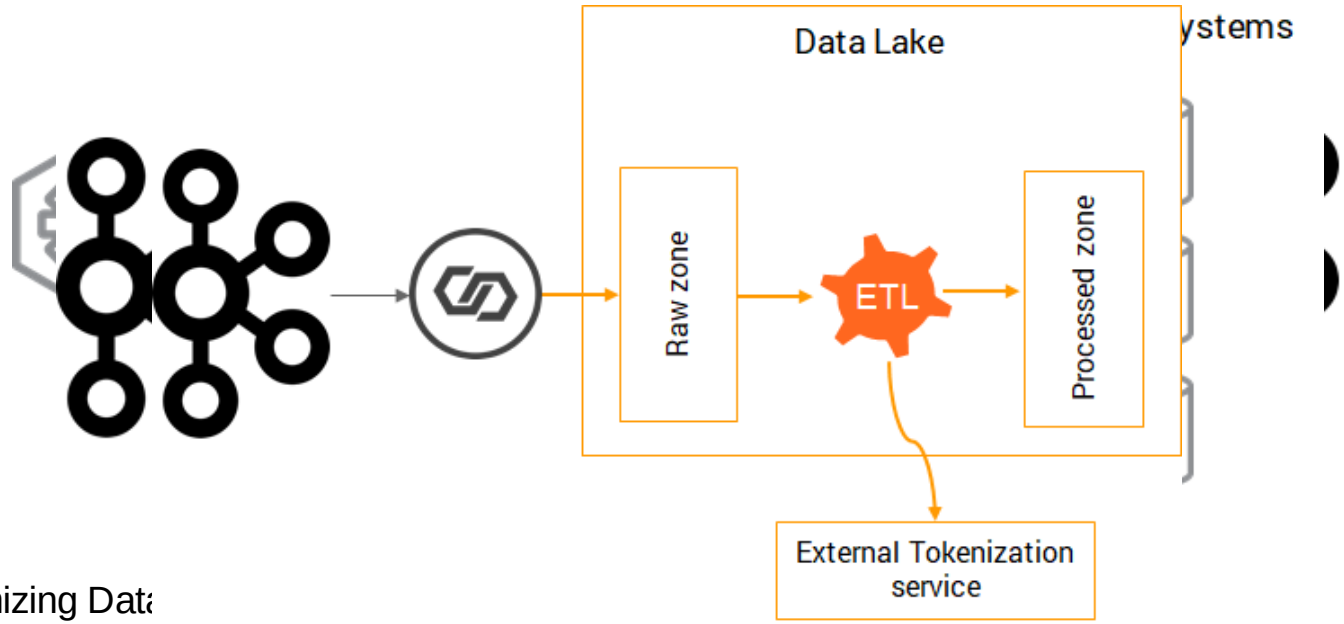
SKŁADOWANIE DANYCH
BEZPIECZEŃSTWO

- Zabezpieczenie infrastruktury
- Rozdzielenie logiczne obszarów biznesu (bounded-context) zgodnie z zasadami Domain Driven Design
- Separacja i decentralizacja całej infrastruktury odpowiedzialnej za przetwarzanie danych
- Role Based Access Control vs Attribute Based Access Control
- Szyfrowanie danych (in transit and at rest = KMS, TLS)
- Virtual Private Clouds – tworzenie bezpiecznych odseparowanych stref (bez dostępu publicznego)



BEZPIECZEŃSTWO DANYCH
TOKENIZACJA PII / MASKOWANIE

- Wymagania GDPR, regulacje związane z danym sektorem (finanse, ochrona zdrowia, inne wrażliwe obszary)
- Tokenizacja / maskowanie / anonimizacja danych umożliwiających identyfikację Klientów (**Personal Identifiable Information**) przykłady implementacji :



Source : Masking & Tokenizing Data

OBJECT STORE
SIMPLE STORAGE SERVICE

- Nieograniczony, wysoko dostępny, elastyczny, trwały,
  Trwałość 99.999999999%, stałe opóźnienia, wysoka przepustowość łączy, tani ☺
- Model spójności - *Consistency model* – ***STRONG****!!!*
  Prev :Read after write consistency (for new objects) and eventual consistency for DELETES and PUTS of the existing objects
- Zdarzenia (on PUT/POST/COPY etc..)
  Lambda, SNS, SQS – budowanie systemów opartych o zdarzenia (event driven)
- Wersjonowanie
- Bezpieczeństwo szyfrowanie danych *in transit and at rest*
- Możliwa replikacja pomiędzy różnymi regionami

OBJECT STORE
SIMPLE STORAGE SERVICE

- Koncepcja obiektów
  Object Key – pełna ścieżka do obiektu), „wirtualne foldery”
- Globalnie unikalna nazwa bucketu
- Max rozmiar pliku 5 TB, jeśli ładujesz więcej niż 5GB – multi-part uploads
- Metadane, S3 Inventory, indeksowanie + SimpleDB
- Wersjonowanie + life cycle policy
  Polityki zmiany klasy / archiwizacji danych w zależności od temperatury danych. Poprzednie wersje mogą być automatycznie wrzucane w Glacier po określonym czasie.
- Praca z plikami S3 – przetwarzanie wielowątkowe
- Wydajność 3500 / 5500 requests per sec / prefix
  (nie ma potrzeby dodawania salts w nazwie obiektów) AWS docs

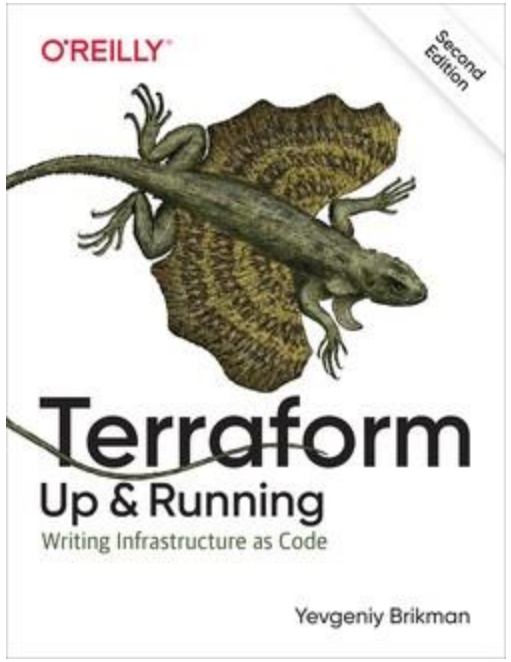
Amazon Simple Storage Service (S3)

OBJECT STORE
S3 STORAGE TIERS

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

WSTĘP DO LABORATORIUM
TERRAFORM 101

Infrastruktura jako kod, **CloudFormation** czy **Terraform** ?

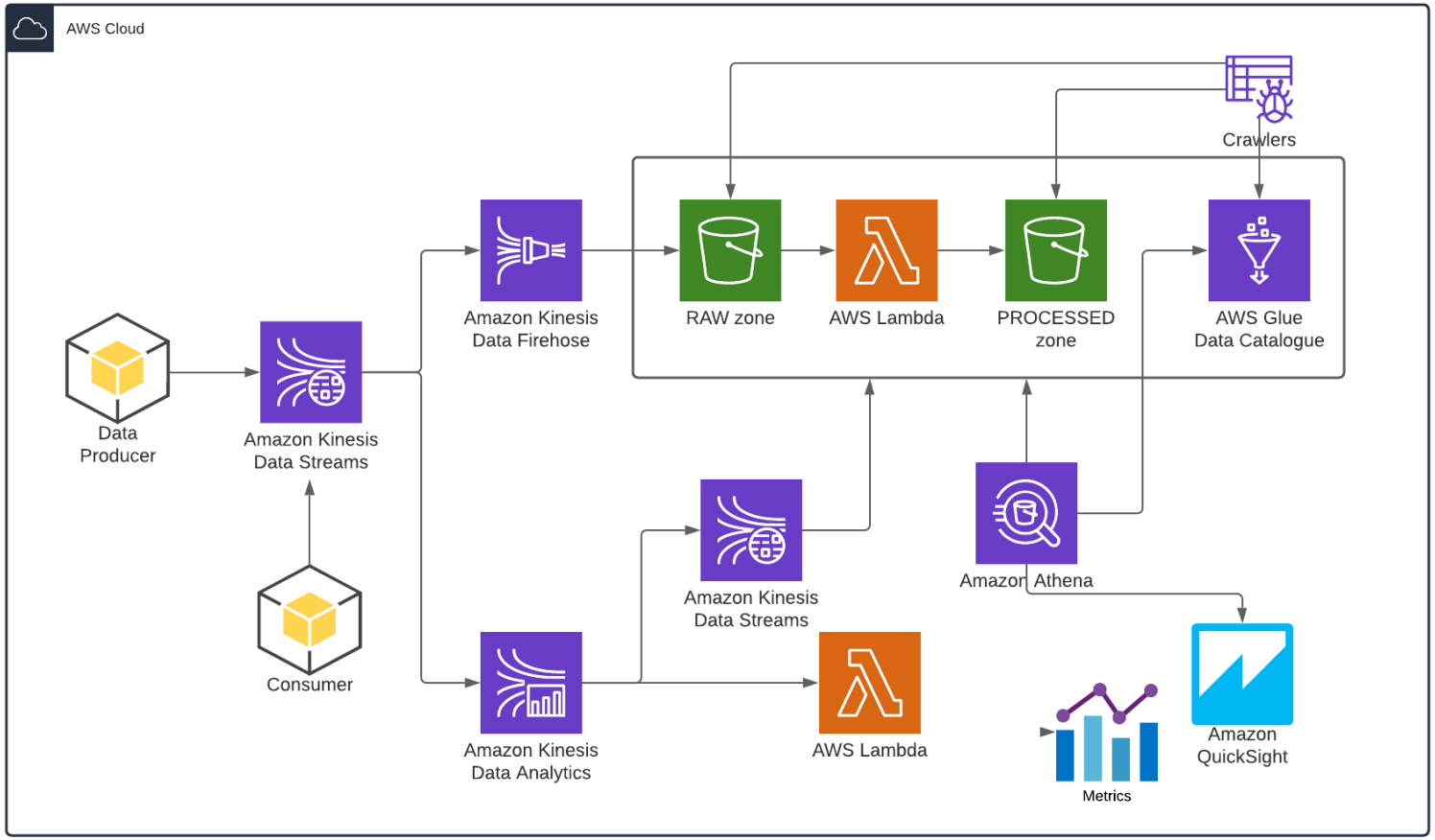


ABC Terraform

- **INIT** – inicjalizacja środowiska, providerów, stanu Terraform
- **PLAN** – wyznaczenie delty (nowe obiekty)
- **APPLY** – wdrożenie planowanych zmian
- **DESTROY** – usuwanie wszystkich obiektów, które zostały stworzone (I są śledzone w plikach stanu)

HashiCorp Terraform

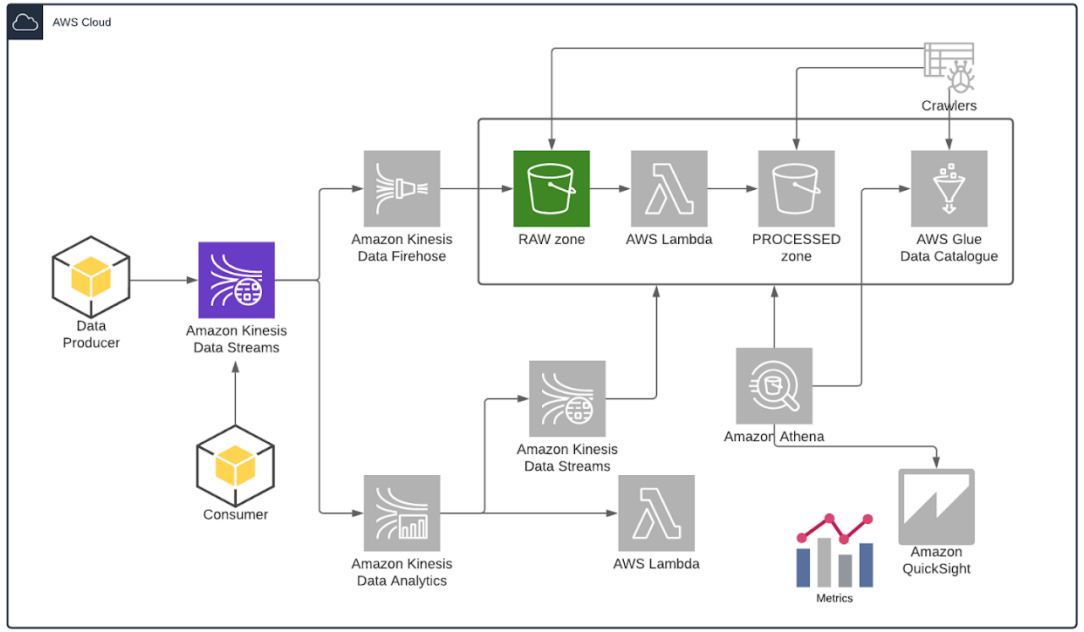
WSTĘP DO LABORATORIUM
OMÓWIENIE SCENARIUSZA



LABORATORIUM
ĆWICZENIE 1 : PODSTAWOWE OBIEKTY

Zakres i cel ćwiczenia

- Pierwsze kroki z Terraform
- Utworzenie podstawowych struktur – S3 Bucket + Kinesis Data Stream
- Generowanie danych testowych
- Czytanie danych z Kinesis Data Stream



MODUŁ 3
ZASILANIE DATA LAKE

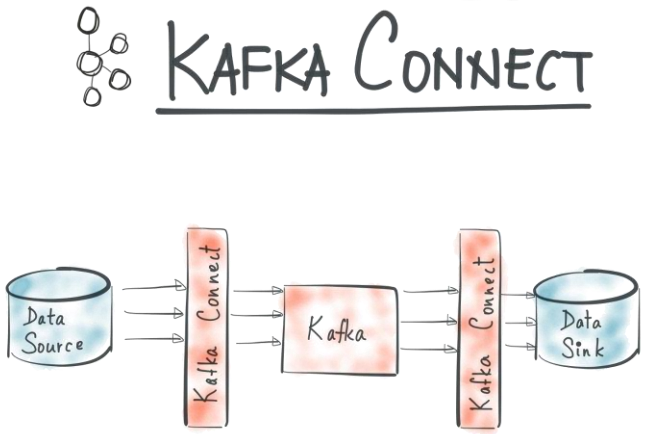
- Dane strumieniowe podejście tradycyjne (Kafka)
- Przetwarzanie strumieniowe z wykorzystaniem serverless (Kinesis Streams / Firehose)
- Przetwarzania batchowe, źródła zewnętrzne, zadania ETL
- Synchronizacja zadań na przykładzie Airflow

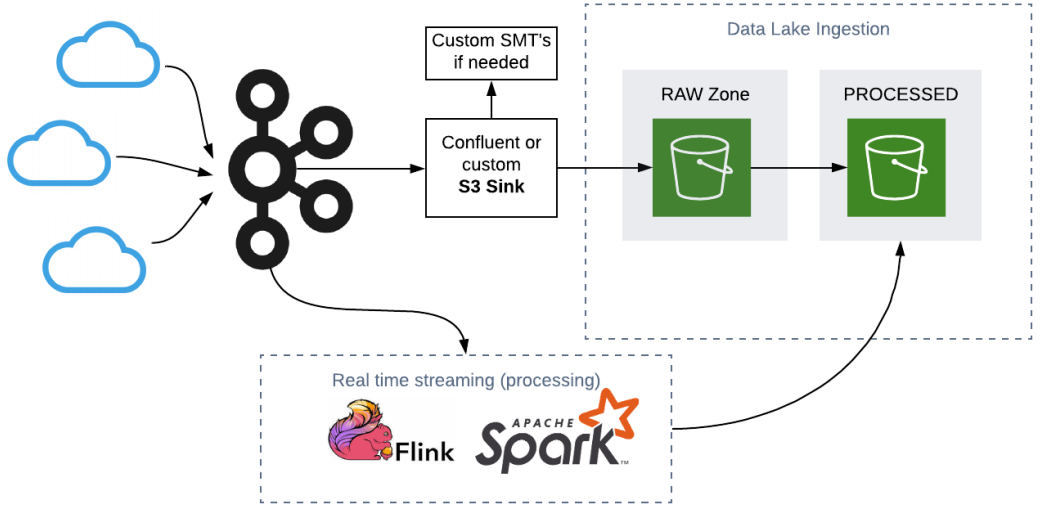
ZASILANIE DATA LAKE
SCENARIUSZE I WYMAGANIA

- Różne wymagania ze względu na opóźnienia. Jakie są rzeczywiste potrzeby biznesu? Co to znaczy real time?
- **Real time** streaming – natychmiastowe działania, opóźnienia liczone w ms
- **Near-real time** – podejście reaktywne, akceptowalne opóźnienia rzędu minut
- **Batch processing** – przetwarzania danych historycznych, duże zbiory, zewnętrzne źródła
- Konieczność synchronizacji zadań (*orchestration and scheduling*)

ZASILANIE DATA LAKE
DANE STRUMIENIOWE + MSG BROKER (KAFKA)

- Klasyczne podejście
  Apache Kafka (lub MSK) + (jeśli AVRO to Schema Registry) + (Kafka Connect S3 Sink / inna technologia RT)
- Przetwarzanie czasu rzeczywistego (analityka)
  Spark Streaming / Flink / Kafka Streams / KSQL
- Zasilanie DL danymi w oryginalnej postaci

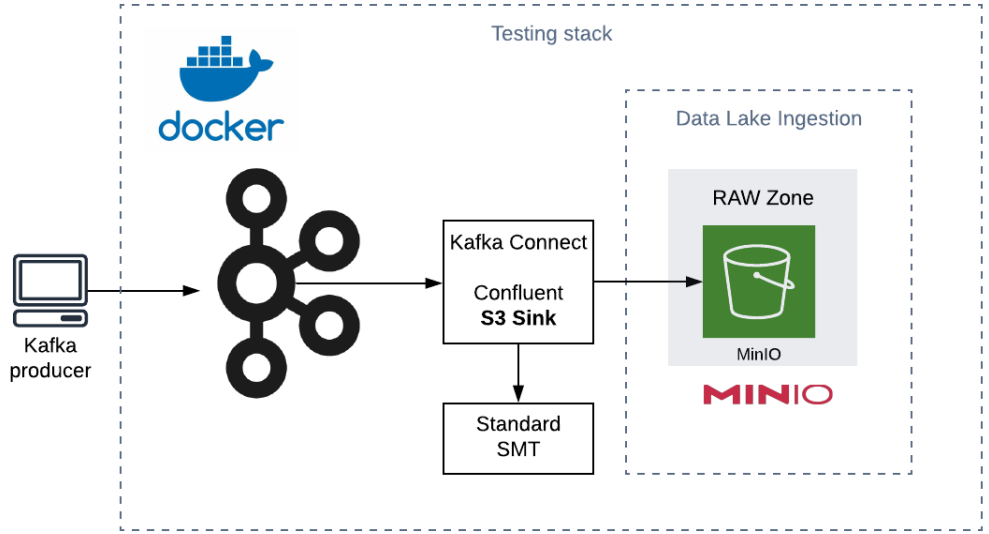




CONFLUENT

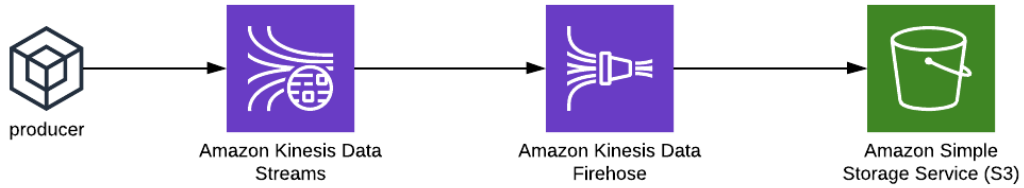
KAFKA CONNECT / ZASILANIE JEZIORA DANYCH
DEMO

- Klasyczne podejście do zasilania DL strumieniem danych w oparciu o Kafka
- Omówienie Kafka Connect S3 Sink
- Single (simple) Message Transformations



KINESIS DATA STREAMS + FIREHOSE
DEMO

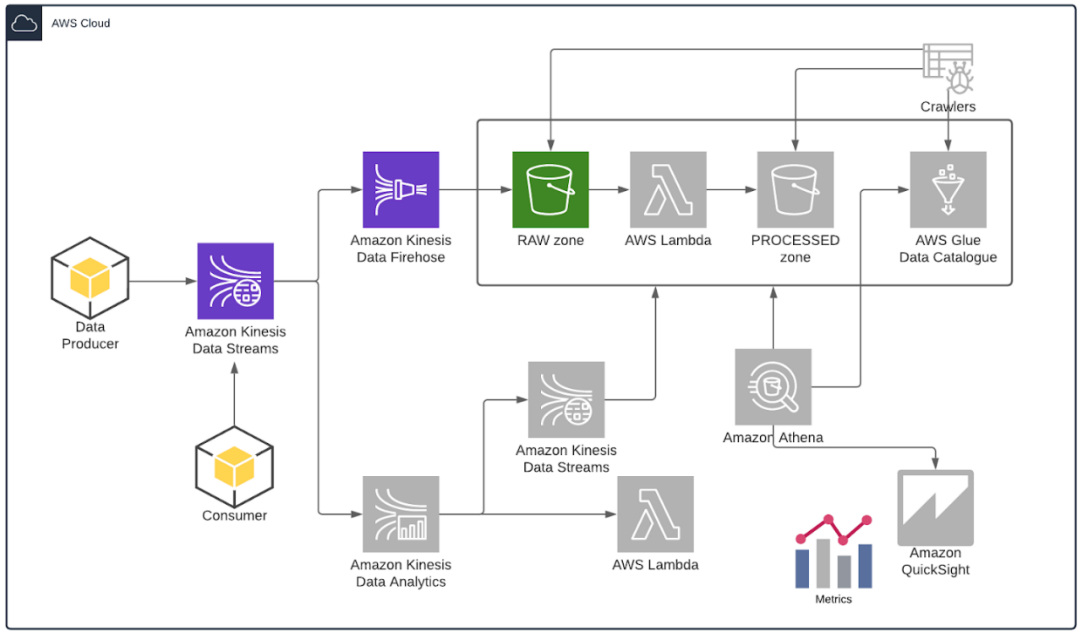
- Tworzenie Kinesis Data Streams, wysyłanie i konsumowanie wiadomości
- Omówienie Streams + Firehose w scenariuszu ładowania do S3



LABORATORIUM
ĆWICZENIE 2 : ZASILANIE DATA LAKE

Zakres i cel ćwiczenia

- Utworzenie Kinesis Firehose
- Utworzenie podstawowych struktur – S3 Bucket + Kinesis Data Stream
- Generowanie danych testowych
- Czytanie danych z Kinesis Data Stream

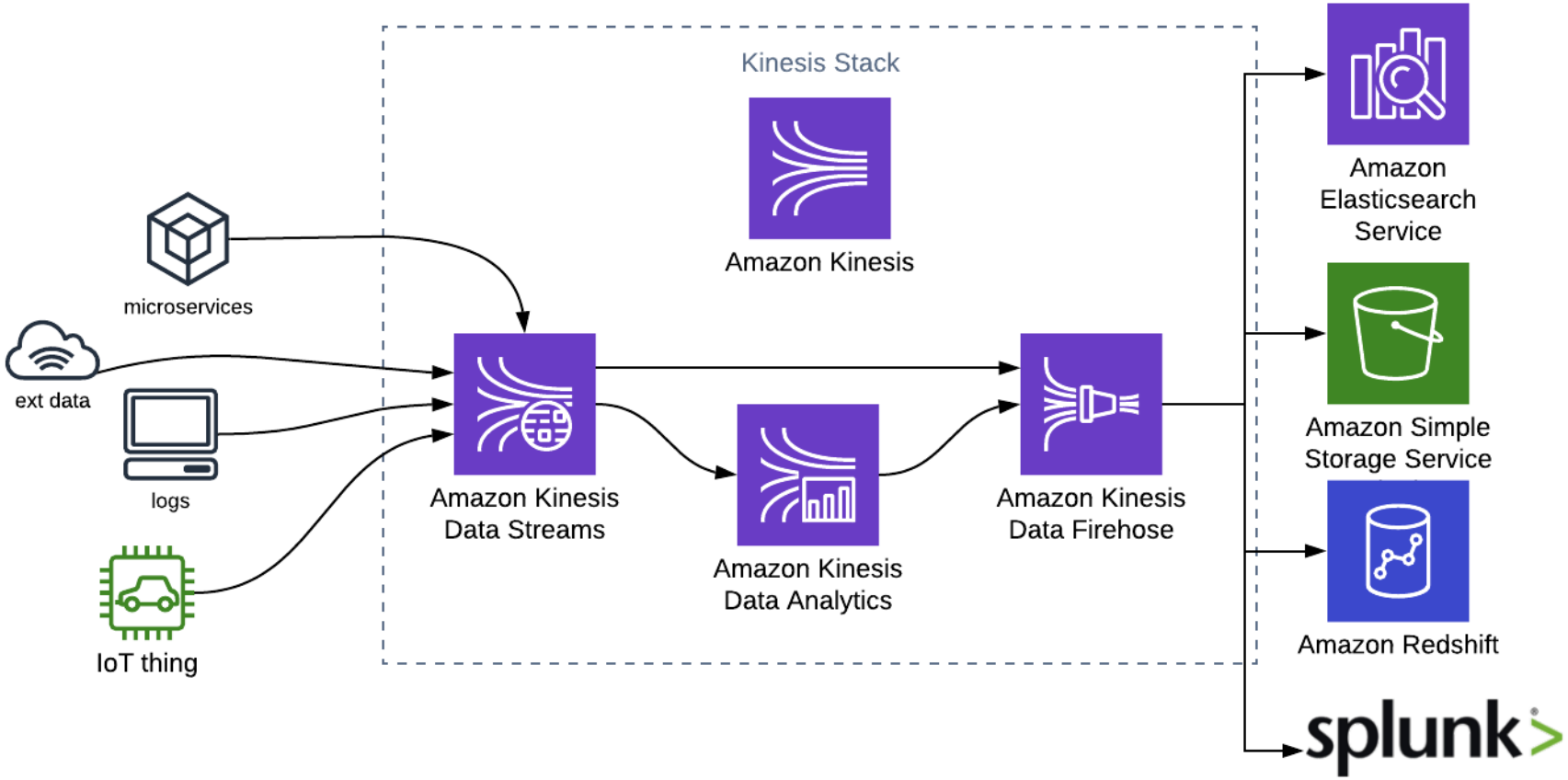


ZASILANIE DATA LAKE
DANE STRUMIENIOWE + SERVERLESS KINESIS

- **Kinesis** to zbiór usług **serverless**
  - Kinesis **Streams** (Kafka) – streaming, low latency
  - Kinesis **Firehose** (S3 sink) – ładowanie danych m.in do S3
  - Kinesis **Analitycs** (Flink) – analityka real-time
- **Streams** w domyśle zastępuje Kafkę
  jednak jest dużo uboższy jeśli chodzi o funkcjonalność
- **Streams** do przetwarzania danych real time (append only stream)
  IoT, logi aplikacji, clickstreams, można budować w oparciu o nią mikroserwisy
- Dobra **integracja** z innymi technologiami (Spark, NiFi etc)
- Wysoka dostępność - dane są automatycznie replikowane na 3 AZ
  nie musimy martwić się o skalowanie, dostępność (HA mamy out of the box)

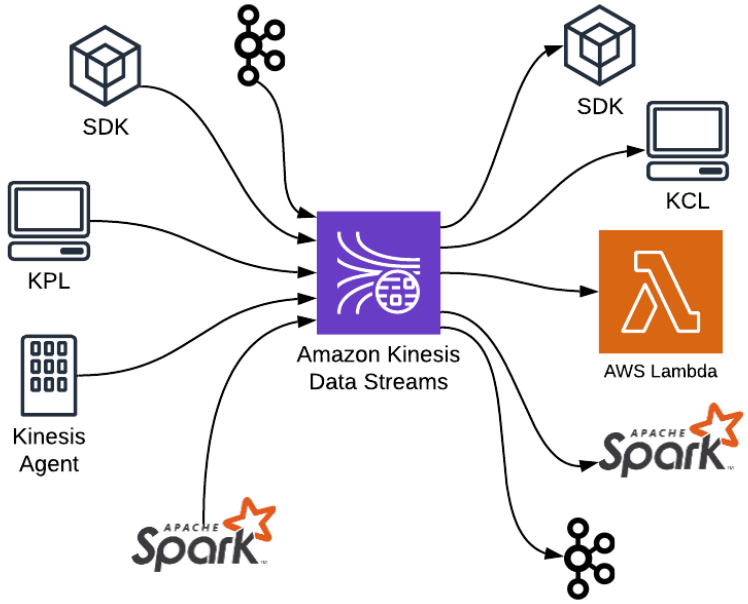
Amazon Kinesis

AWS KINESIS STACK
STREAMS, ANALITYCS, FIREHOSE



KINESIS STREAMS
PRODUCERS & CONSUMERS

- Wysyłanie wiadomości do Kinesis (**producers**)
  1MB lub 1000 rekordów per shard
  SDK (API PutRecord(s) np. Lambda, low throughput, latency)
  KPL (C++/JAVA) low latency / high throughput, agregacje, kolekcje rekordów (RecordMaxBufferTime), sync / async (auto-retries)
  Kinesis Agent – do monotirowania logów (Java based)
- Konsumowanie rekordów (**consumers**)
  2MB/s 5 API calls / per SHARD
  SDK (API GetRecords)
  KCL – Klient umożliwiający realizację consumer groups (checkpointing na DynamoDb)
  Lambda – do budowy lekkich procesów ETL (batch size
  3rd Party – Spark, Kafka Connect
  Enhanced consumer (Push również działa z Lambda)
  2MB/s per shard / per consumer + 70ms latency



KINESIS STREAMS VS KAFKA
PORÓWNANIE

Kinesis Streams to uproszczona wersja Kafka topics. W pełni serverless ale z ograniczeniami (kompromis pomiędzy funkcjonalnością a serwisem zarządzalnym)

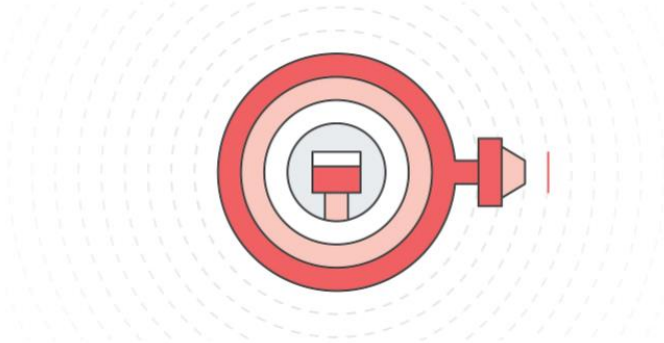
- Brak headerów, event key jest stringiem (do partycjonowania), Nie ma transakcji
- event timestamp jako LogAppendTime (ApproxArrivalTimestamp)
- 200 ms minimalne latency na standardowym konsumerze (5 API calls/s) lub ~70ms na enhanced
- Możliwość kompresji / przetwarzania mikrobatch

	Apache Kafka	AWS Kinesis Streams
Główna jednostka alokacji danych	Topic	Stream
Składowanie danych	Partycja	Shard
Zapewnienie kolejności	Na poziomie partycji	Na poziomie Shard'a
Przechowywanie danych (retention)	Konfigurowalne (brak MAX)	24h (default) do 365 dni
Rozmiar danych per msg	1MB (by default)	1MB
Modyfikacja partycji	UP	UP/Down z ograniczeniami
Replikacja danych (HA)	Mirroring	Automatyczna 3 AZ
Zależności	Zookeeper*	DynamoDB for checkpointing
Unique ID	Offset	Sequence number
Złożoność / narzut administracyjny	Duży	Serverless

KINESIS FIREHOSE
ZASILANIE NEAR-REAL TIME

- Automatycznie skalowalna usługa do zasilanie near-real time
  ograniczenia dla Direct PUT 1,000 rec/sec, 1,000 req/second, and 1 MiB/second)
- Konfigurowalne mikrobatche
  definiowane poprzez “BufferSize” lub ”BufferTime” (min. 60 sec) po osiągnięciu których następuje zapis rekordów (podobna zasada działania co Kafka-Connect S3 sink)
- Proste transformacje
  Formatów np do Parquet oraz spięcie Firehose z Lambdą do pre/post processingu
- Ładowanie danych do S3, Redshift, ElasticSearch, Splunk

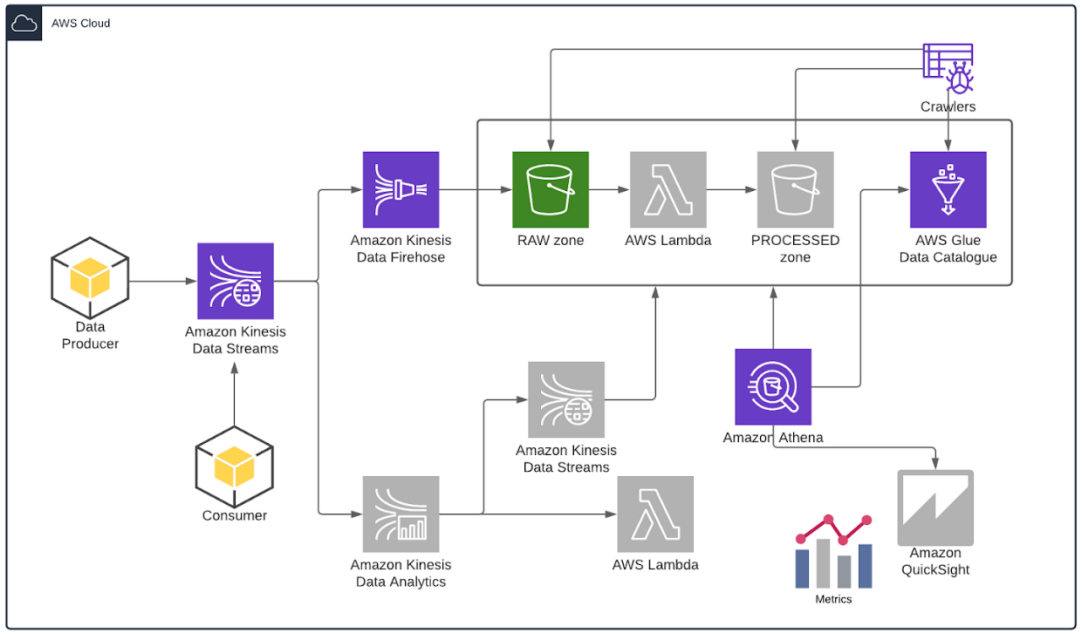
Amazon Kinesis Data Firehose



LABORATORIUM
ĆWICZENIE 3 : GLUE DATA CATALOGUE & ATHENA

Zakres i cel ćwiczenia

- Rejestrowanie table w Glue Data Catalogue
- Tworzenie Crawlerów i inne metody rejestrowania struktur w Glue
- Analiza danych za pomicą SQL w Athena (Presto)

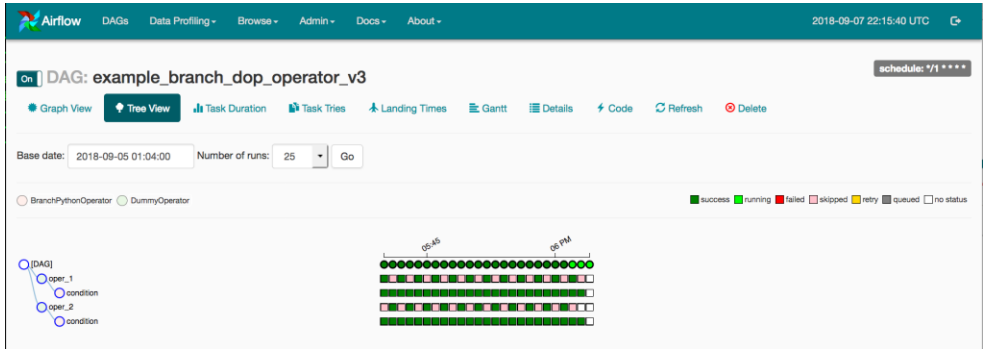


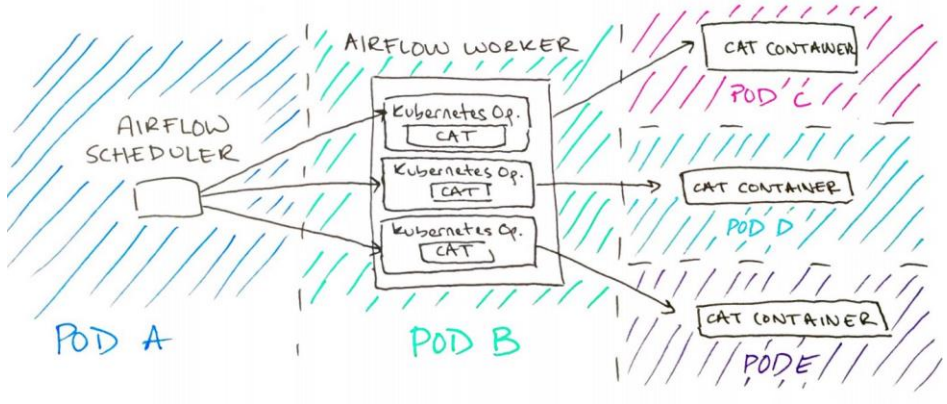
MODUŁ 4
PRZETWARZANIE DANYCH W DL

- Wykorzystanie natywnych usług w chmurze
  - Funkcje Lambda
  - Glue Spark / Python
  - EMR
  - Athena (Presto)
  - Przetwarzanie strumieniowe z Kinesis Analytics (Flink)

PRZETWARZANIE DANYCH
SYNCHRONIZACJA I PLANOWANIE ZADAŃ – AIRFLOW

- Synchronizacja, planowanie i wykonywania zadań cyklicznych **ETL** (batch)
- Skalowalne, elastyczne, gotowe rozwiązanie oparte o kolekcje operatorów, connectorów, sensorów
- Różne modele executorów (Celery, Dask, Kubernetes)
- Oparte o DAG's (Directed Acyclic Graphs)
- Wyzwania w serverless

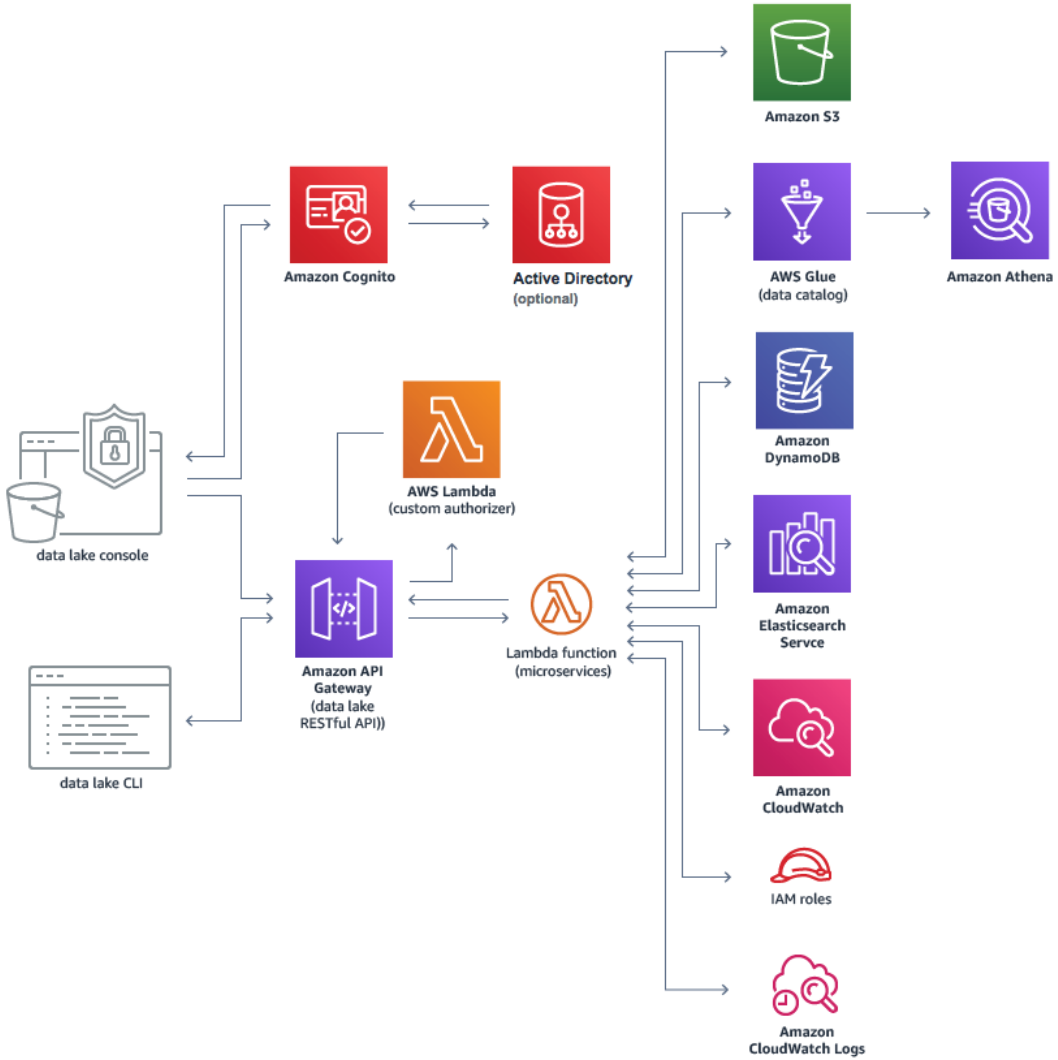




Source : Bluecore - [We're All Using Airflow Wrong and How to Fix It]

PRZETWARZANIE SERVERLESS
AWS LAMBDA

- Przetwarzanie danych serverless **event-driven data processing**
  Uruchmiamy kod wykonywalny (nie zależny)
- Klej który skleja ze sobą usługi sieciowe
  Wyzwalanie funkcji poprzez zdarzenia na S3, DynamoDB, Kinesis, SQS, Step functions pozwalają modelować algorytmy przepływów etc.
- Idealne do małych, krótkich zadań
  limity : max 3GB memory, 900s, 1000 concurent executions (soft limit), layers 5/250MB
- Typowe przypadki użycia
  Continuous ETL's, przetwarzanie danych z IoT backends, aplikacje Web, API endpoints



Source https://aws.amazon.com/solutions/data-lake-solution/

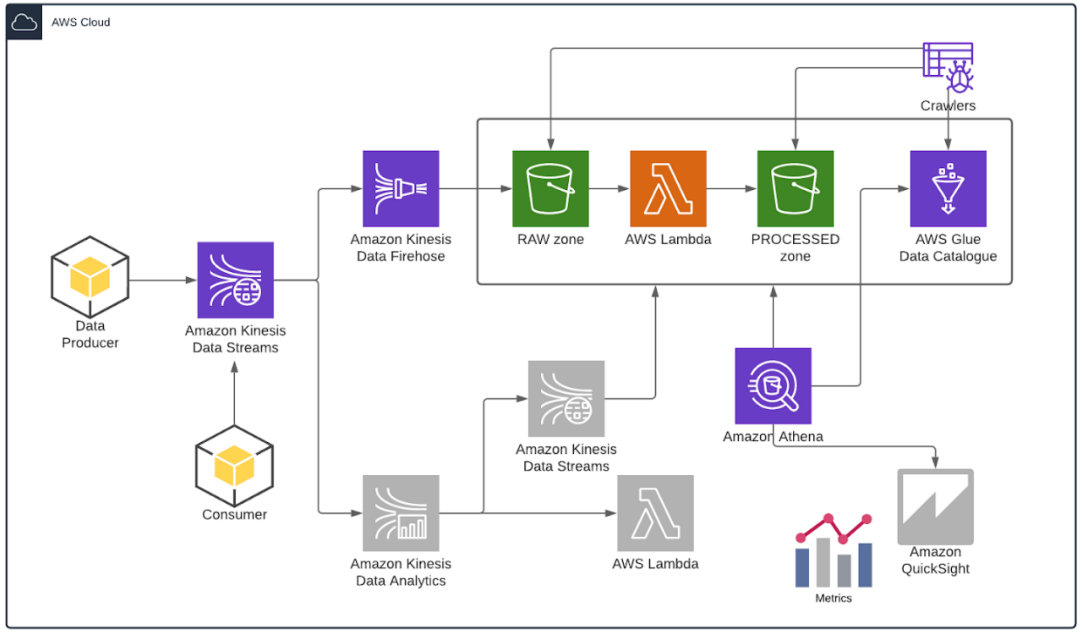
PRZETWARZANIE SERVERLESS
AWS ATHENA

- Bazuje na **Presto 0.217 (v2)** (uwaga na ograniczenia np. ts ms only)
- Wygodny interfejs (**ANSI SQL**) do analizy BigData na S3 (bazuje na Glue DC) – alterantywa dla klasycznych hurtowni – **no ETLs** !
- **Ograniczenia** co do ilości równoczesnych zapytań - API limits + limity samej technologii (128GB RAM per node)
- Ograniczone możliwości tunningu kwerened. Podstawowe techniki to eliminacja partycji, kompresja i optymalizacja ze względu na formaty danych (uwaga kolejność JOIN ma znaczenie – join distribution, duża tabela z lewej !)
- Model kosztowy oparty o ilość przeskanowanych danych
- Wspiera różne formaty danych (AVRO, JSON, CSV, PARQUET, ORC) widoki
- **JDBC / ODBC** integruje się z innymi narzędziami

LABORATORIUM
ĆWICZENIE 4 : CIĄGŁY PROCES ETL (LAMBDA)

Zakres i cel ćwiczenia

- Implementacja ciągłego przetwarzania danych za pomocą funkcji Lambda i wyzwalaczy S3 (RAW > PROCESSED zone)
- Rejestrowanie table w Athena za pomocą SQL
- Mechanizmy porównywania danych w poszczególnych warstwach Data Lake
- Aktualizowanie metadanych w Hive Metastore (ciągłe)



KINESIS ANALYTICS
PRZETWARZANIE REAL TIME SERVERLESS

- Strumieniowy ETL bazujący na **Flink**
- Łatwość tworzenia standardowych (prostych) procesów analitycznych bazujących na rozszerzonej składni SQL
- Możliwość tworzenia przepływów analitycznych reagujących na zdarzenia (responsive analytics)
- **Nie jest tani** – płacimy za czas wykonywania się naszej aplikacji (Java/SQL)
- Serverless, automatycznie skalowalny, możliwość pre-processowania danych, dołączania danych referencyjnych

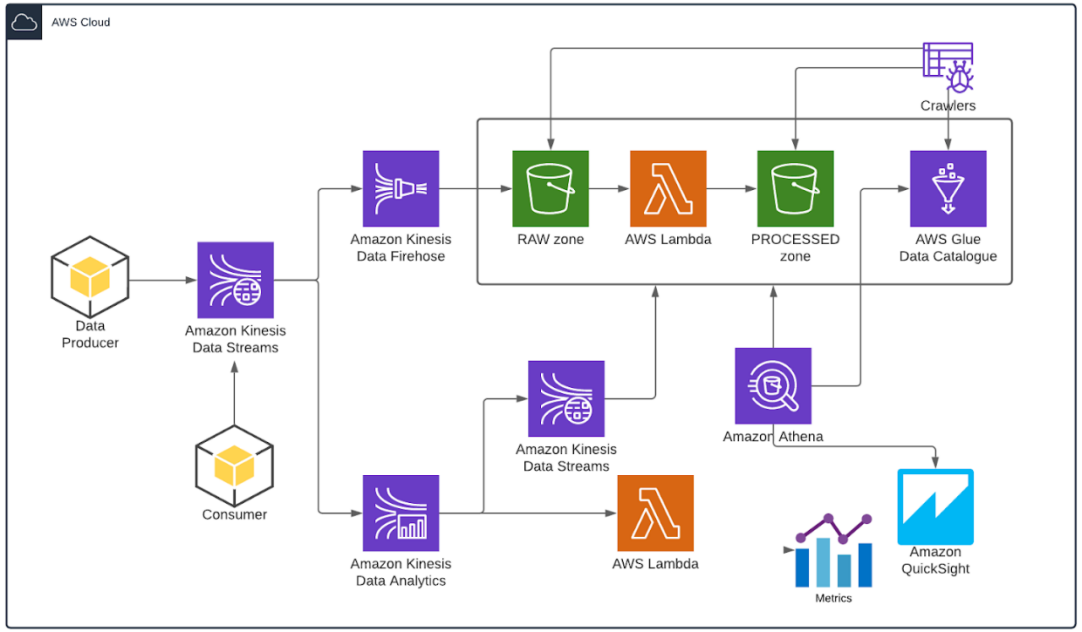
Amazon Kinesis Data Analytics

Apache Flink

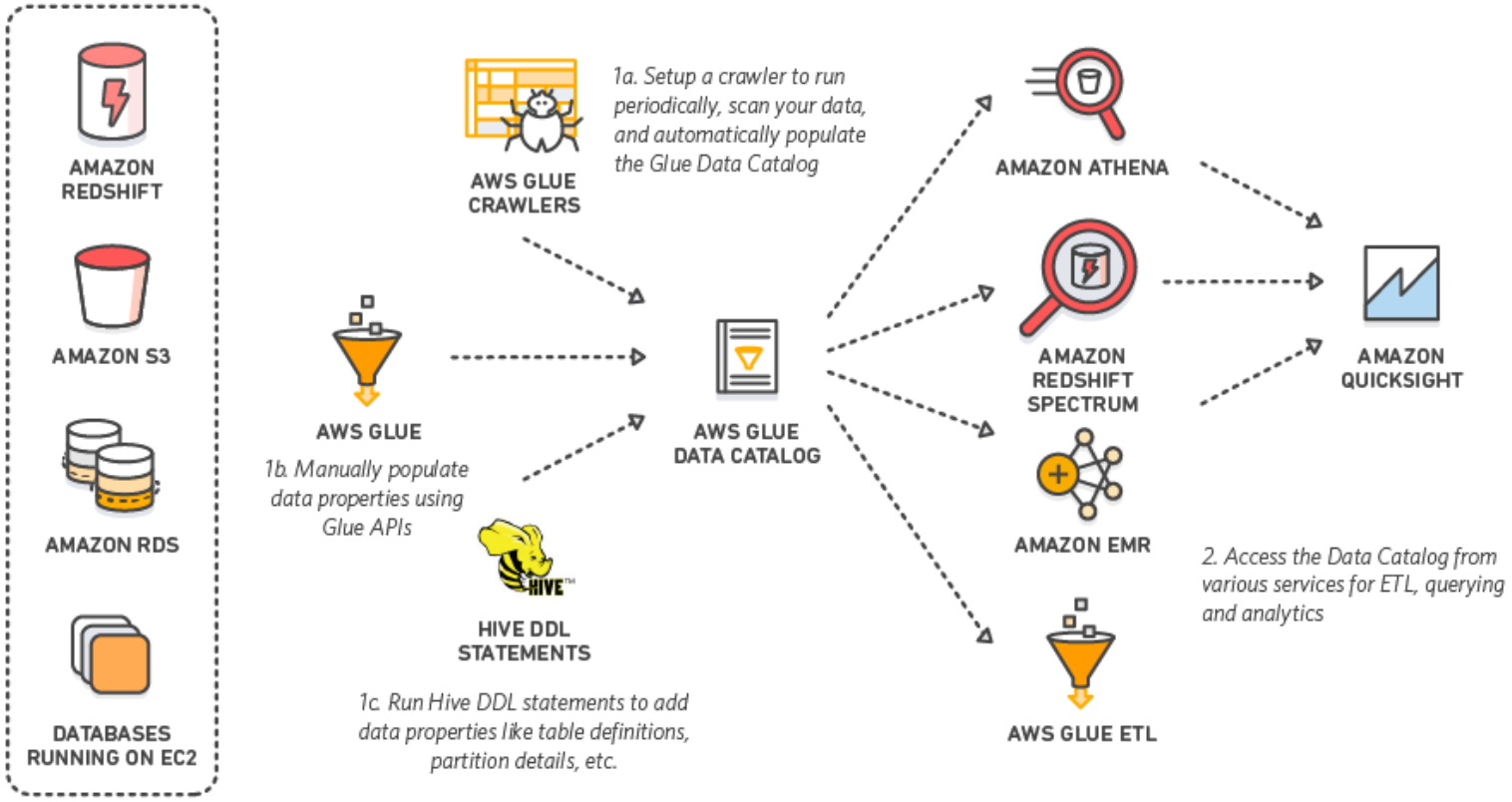
LABORATORIUM
ĆWICZENIE 5 : ANALITYKA REAL TIME

Zakres i cel ćwiczenia

- Kinesis Analytics – real-time jako serwis.
- Wyzwalanie alertów w przypadku wystąpienia zdarzeń (analityka czasu rzeczywistego)
- Real time dashboards

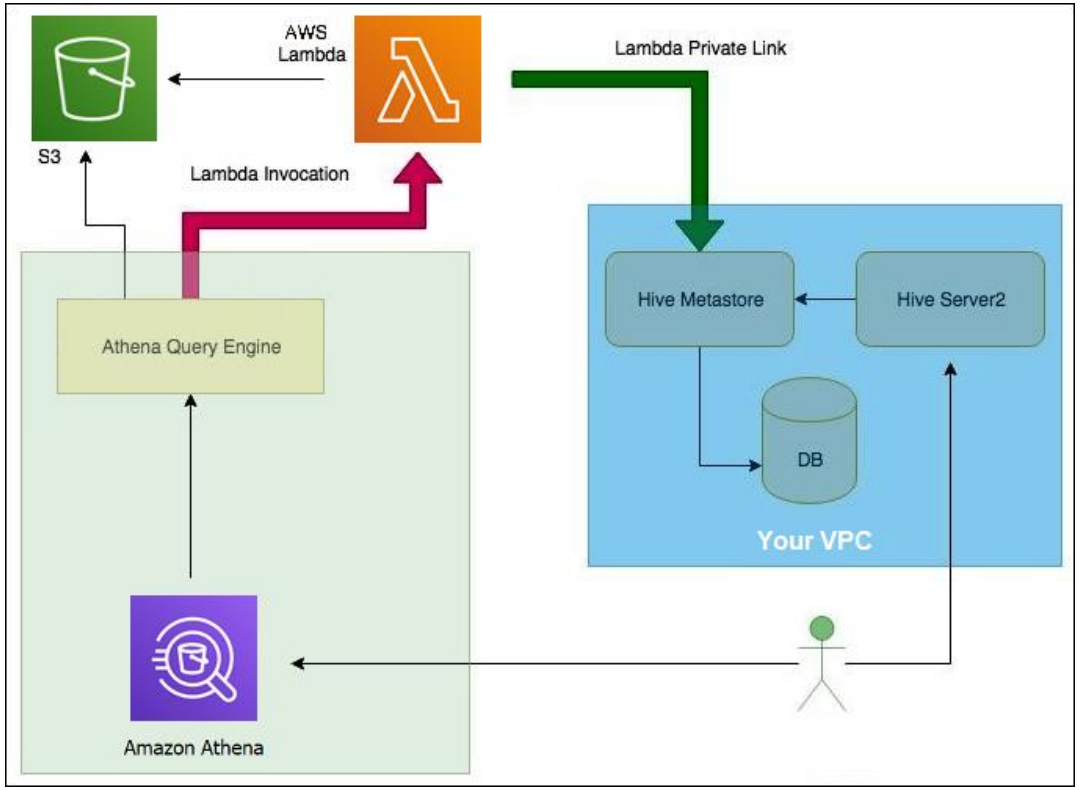


HIVE METASTORE
GLUE DATA CATALOG



Source: AWS docs https://docs.aws.amazon.com/athena/latest/ug/connect-to-data-source-hive.html

ATHENA
EXTERNAL METASTORE



Source: AWS docs https://docs.aws.amazon.com/athena/latest/ug/connect-to-data-source-hive.html

PRZTWARZANIE BATCH
EMR, GLUE

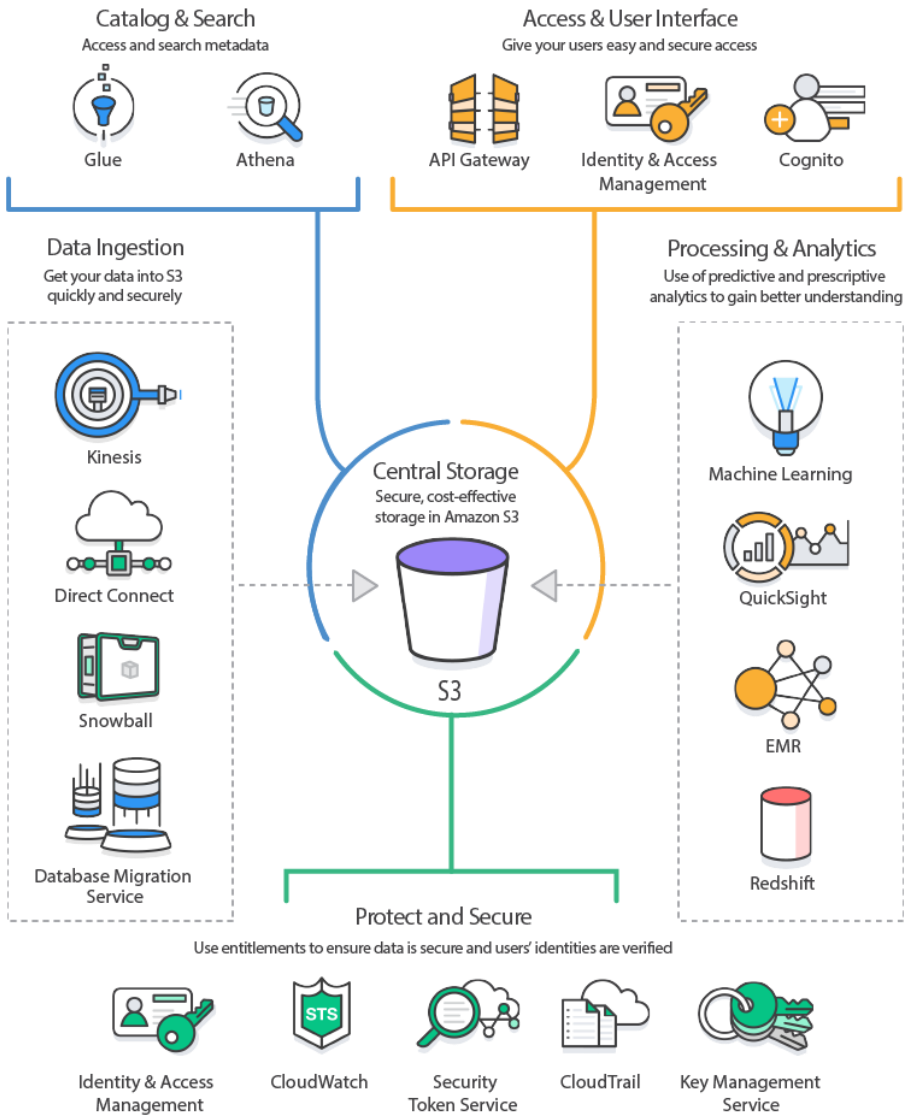
- **Elastic Map Reduce**
  Gotowa platforma z pre-instalowanym i skonfigurowanymi frameworkami do przetwarzania BigData
- Nowoczesny sposób użycia **Hadoop**
  klaster może być uruchamiany na żądanie (czas setupu ~minuty). Nie martwimy się o konfiguracje, zestawienie wszystkich komponentów Hadoop. Uwaga na ograniczenia - versions compatibility. Skalowalny.
- Szeroki wachlarz technologii w jednym miejscu
  Spark, Flink, Presto, Pig, Scoop, Hadoop, HBase, Hue, JupyterHub, Zeppelin, Mahout, Mxnet, TensorFlow and many others
- **Glue** – w pełni serverless framework do przetwarzania danych
  Wsparcie dla Python/Scala do tworzenia zadań ETL tasks (**Spark or Python** jobs)

EMR

Glue

BIG DATA IN THE CLOUDS
PRZYKŁADY UŻYCIA

- **Data Lake**
  W teorii nieskończona przestrzeń do składowania danych, liniowa skalowalność, stałe opóźnienia, HA, bardzo dobra integracja z innymi usługami
- **Przetwarzanie danych z IoT**
  Może wspierać miliardy urządzeń, przetwarzań nieograniczone (w teorii) ilości zdarzeń, analizować w czasie rzeczywistym niezawodnie i bezpiecznie
- **Uczenie maszynowe / AI (Sagemaker)**
  W pełni zintegrowane środowisko (IDE) dedykowane dla uczenia maszynowego, tworzenia, trenowania i tuningu modeli (hyper parametrization)



Questions ?