Build, Don’t Buy

Enable Analytics, Machine Learning, and Forensics with Security Data Lake on AWS

Presenter:

Eric Gifford – Enterprise Security Architect, Cambia Health Solutions

Agenda

- Why build?
- What makes sense to build?
- Data Lake for InfoSec pros
- Security Data Lake?!
- Patterns for Machine Learning and Analytics

Cambia Health Solutions

Cambia → Born from an inspired idea

Cause → Transform health care into a **person-focused** and **economically sustainable** health care system

Embracing cloud innovation to provide **personalized & intuitive** experiences

On AWS: Web/Mobile applications, micro-services, data pipeline, data science

#whoami

Eric Gifford

- InfoSec Architecture & Engineering team
- Healthcare since 2008
- **Not** born cloud first
- Worked on traditional security challenges:
  - Multi-datacenter web gateway deployment
  - Web App Firewall implementation
  - Incident response processes
  - PCI-DSS consulting
- AWS Certified 2014

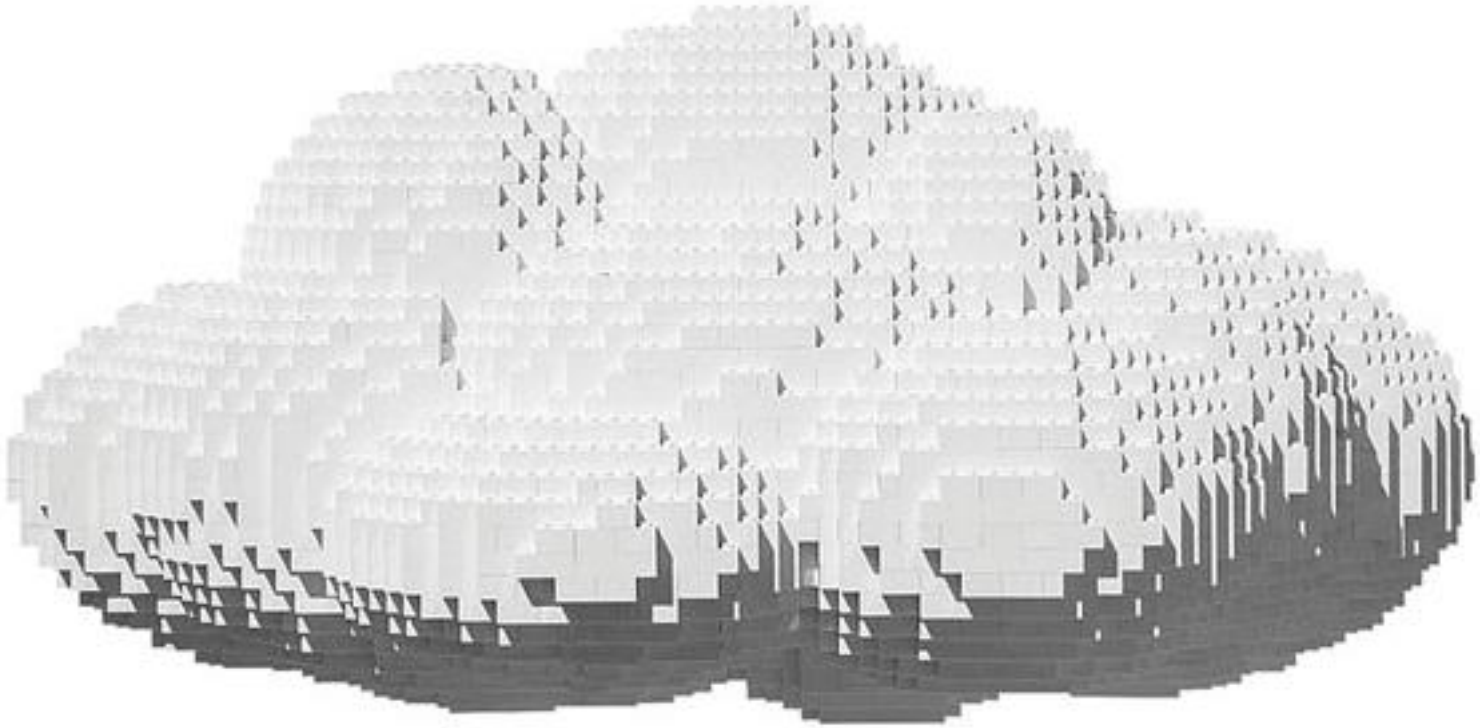
What cloud do you see?

Scary cloud?

Friendly cloud?

Silly cloud?

The cloud I see…



Why Build on cloud?

Cost

- If we can build something **simple** that works, why pay somebody else for it?
- Traditional **fixed-count licensing is wasteful** when compared to consumption-based pricing

Agility

- **Defer** long-term **commitment** and vendor lock-in
- **Keep pace** with the business
- Isn’t working out? Turn it off and **try something else**

Scalability

- Many cloud services are elastic and will scale out under load, so **your security needs to scale under load too**
- Security controls in your data center, are **not as reliable as security controls in the cloud** (e.g. uptime SLA).

What makes sense to build?

Start small and experiment

- Write **utility code** that sets up one type of centralized logging (e.g. AWS CloudTrail)
- Configure monitoring for “fresh logs” and **alert if they stop flowing**
- Take on the next log type, **keep iterating** with intense focus on delivery
- **Leverage open source** to avoid starting from scratch

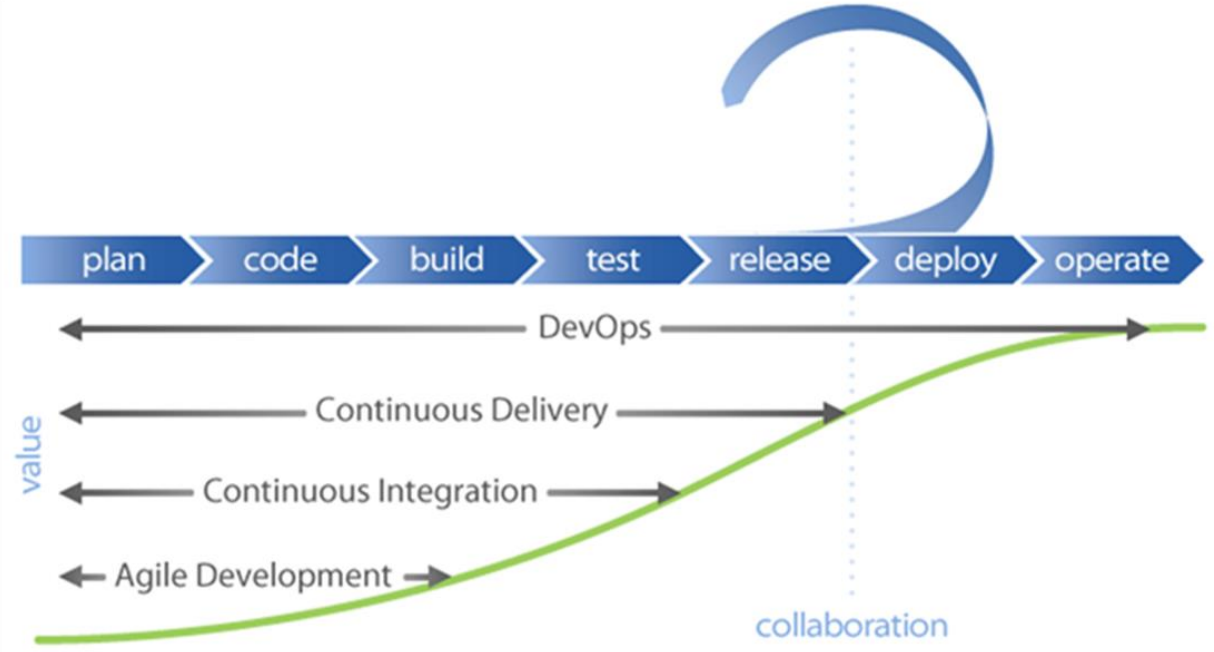
Cautiously consider ownership and support for any critical services

- Focus on learning and **build confidence** out of the gate
- Example of a critical service: Egress filtering from cloud via proxies
- **Be pragmatic**…Is your team prepared 24/7 to rapidly restore functionality to cloud environments
- If not, **work with cloud engineering partners** on design and implementation such that their team owns the service and support.

What to know before you build

You will need somebody to write "glue code"

- Python is fine
- Years of experience **not** necessary



Aligning to DevOps principles

- The people that **write it** will **support it**
- Leverage **automation** (e.g. Ansible playbooks)
- **Version control** the code (git)
- Keep a **backlog** (Kanban works well)
- Test changes in lower environments and **promote when tests pass**

Principles for building on cloud

Keep it Simple → **Complexity is the enemy** of security
Automate → Reduce quality deviations
Reuse → Invest in reusable patterns
Iterate → Patterns and process may need a “tweak”… often
Manage Less → Prefer serverless vs managing EC2 instances

A complex system that works is invariably found to have evolved from **a simple system that worked**. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system.
~John Gall (Gall’s Law)

Things Cambia InfoSec has built successfully on cloud

Account onboarding CloudFormation template → Lays down infrastructure and configuration for centralized collection of VPC Flow logs, Database logs, EC2 host events, S3 data events, etc.

Lambda "watchers"→ New VPC deployed? Our Lambda function discovers it and ensures logging is set correctly. New load balancers deployed? Lambda function discovers it and configures logging

ELK stack → Ad hoc log search and visualization for the above logs

Data Lake → Long-term persistence of enterprise security logs in raw form

Egress Proxy pool → Centrally managed Squid proxy cluster with URL whitelist

Incident Response API → Pass in AWS account number and instance ID to trigger automated evidence preservation and network containment

You can do it too!

Step 1 → Within 30 days **assess the state of logging** in your cloud environments

- Examples: Network logs, host logs, API logs, database logs, access logs, etc
- Identify gaps: Logging not occurring at all, not centrally collected, no visibility for infosec

Step 2 → Within 90 days **develop pattern(s) for central collection** of high priority logs

- Divide and conquer: API logs from cloud provider (e.g. AWS CloudTrail) is an easy place to start
- Prove success then iterate quickly

Step 3 → Within 180 days **provide code to run in each "monitored" account**

- CloudFormation works well for this
- Version control it and test thoroughly
- Periodically provide an updated template with new capabilities and improvements

Let’s briefly recap

- Cloud IaaS provides highly interoperable **building blocks that work well for security use cases** too
- Initially, **start small** and experiment; Defer building/owning critical services
- Low-complexity glue code is often enough; Align to DevOps principles
- Keep it **simple**, automate, and manage as few operating systems as possible
- Working with logs is a great “safe” first project

Let’s talk a bit more about logging

…the way we've always done it

- **Forecast** # of log producers and/or log volume and buy enough licensing
- Forecast storage needs and procure storage
- Forecast system requirements and build servers
- **Onboard** new log types, work through log format issues and field extraction
- **Nag** system owners to repoint to the new log destination, then nag again
- **Pay** for idle capacity during the implementation
- Purchase more infrastructure and licensing as log volume grows
- Do something else with the data? Haha! No. We've built a data jail

TL;DR: Traditional logging, log visibility, and log analytics projects take a long time and cost $$$$

What if…?

What if we could keep all logs on **cheap cloud storage** until we are ready to use them?

…and never again choose to not log something because it's too expensive.

What if we didn't have to format the logs when we write them?

…and instead **just apply a schema when we read** our logs.

What if we could break the "data jail" pattern to enable multiple downstream log consumers?

…and experiment in an agile manner?

Data Lake

Data Lake - The “generic” pattern

“A data lake is a method of storing data within a system or repository, **in its natural format**, that facilitates the collocation of data in **various schemata** and structural forms, usually object blobs or files. The idea of data lake is to have **a single store** of all data in the enterprise ranging from raw data (which implies exact copy of source system data) to transformed data which is used for various tasks including **reporting, visualization, analytics and machine learning**.”

~Wikipedia

Data Lake - The “generic” pattern

Data Lake is **not** Enterprise data warehouse (EDW)

- EDW houses transformed and structured data
- Data is generally not loaded into the EDW until the business has a purpose for it

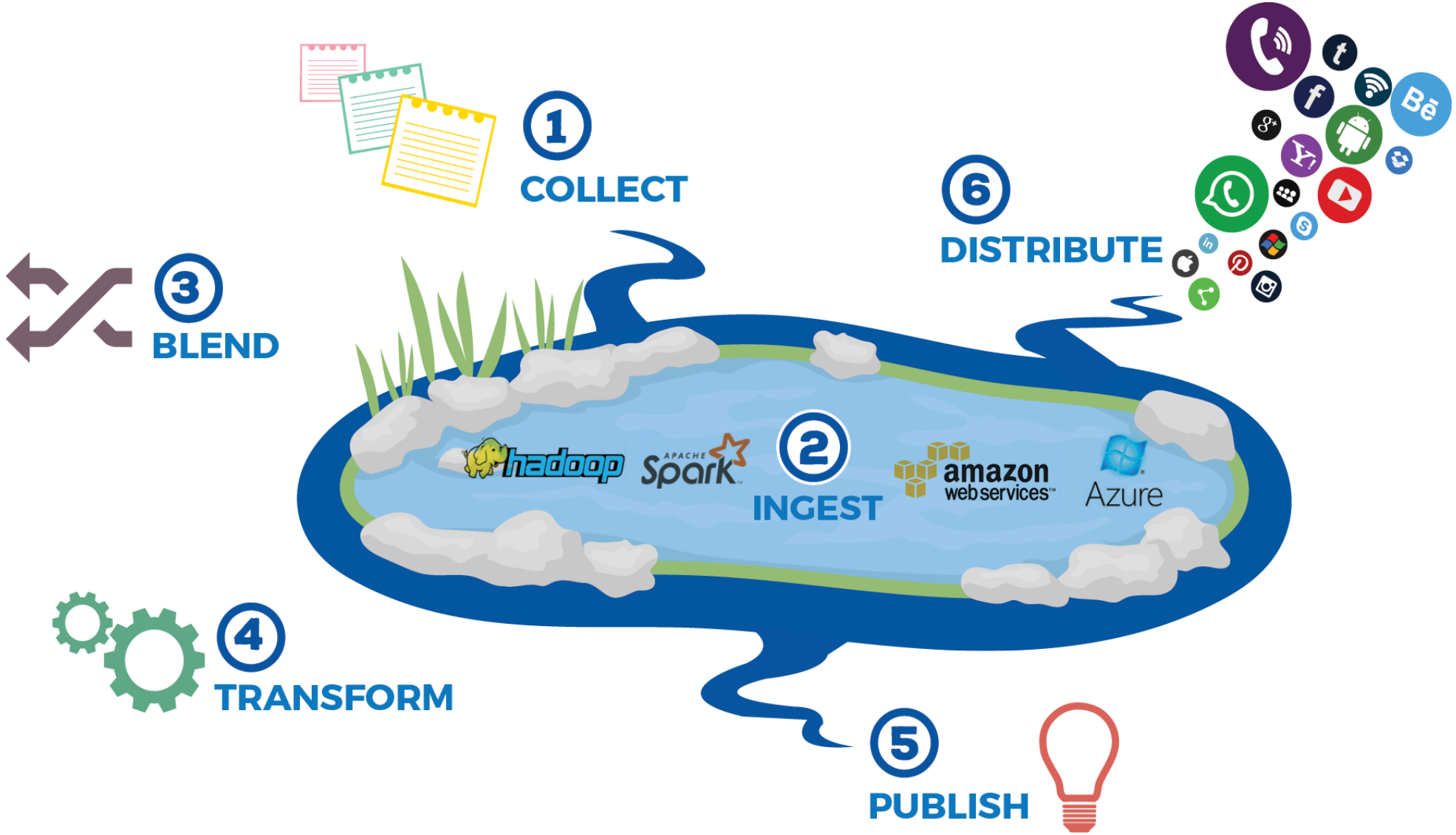
…because it’s too expensive to transform and load everything in advance

Data Lake is **not** Hadoop

- Hadoop is a data processing technology often used to run analytics against data
- Many data lake implementations have a related analytics capability like Hadoop

Data Lake is a pattern

Data Lake (the pattern)



Data Lake for InfoSec

1) **Produce Logs** – We are already good at this… (firewall, web proxy, etc)
2) **Ingest** – Get logs from log producer to the data lake (syslog, upload)
3) **Blend** – Combine with logs from other logs producers (AD, threat intel)
4) **Transform** – Perform analytics (machine learning models, rule-based queries)

~~Distribute~~

~~Publish~~

5) **Use the data and analytics** – Forensics, Reporting, Machine Learning, Insights

Data Lake for InfoSec

Let's revisit our what-ifs…

What if we could keep all logs on **cheap cloud storage** until we are ready to use them?

…and never again choose to not log something because it's too expensive.

What if we didn't have to format the logs when we write them?

…and instead **just apply a schema when we read** our logs.

What if we could break the "data jail" pattern to enable multiple downstream log consumers?

…and experiment in an agile manner?

Data Lake for InfoSec

Recommended storage for a data lake = AWS S3, Azure Blob Storage, Google Cloud Storage

Amazon S3

Google Cloud Storage

Microsoft Azure
Blob Storage

Why?

- Hold any type of data: Raw or Transformed
- Highly durable
- Low cost

Tips

- Use a **unique bucket per log type** (e.g. DNS logs bucket, WAF logs bucket) to facilitate schema on read
- **Partition the logs** in each bucket using **namespace** techniques

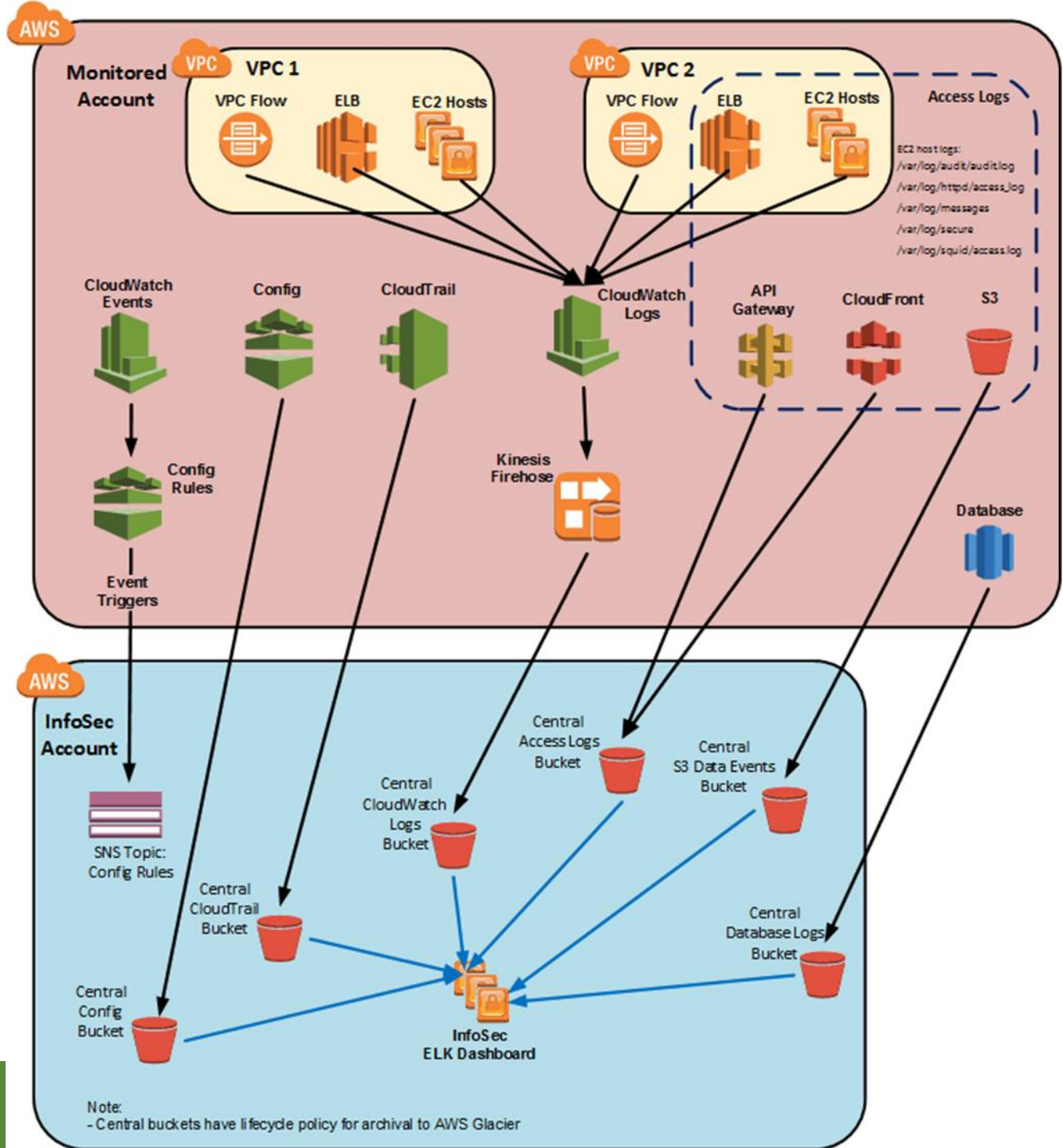
Data Lake for InfoSec

Send cloud logs to Data Lake

- Many AWS native logs can write directly to S3 (e.g. CloudTrail, Config, API Gateway, etc)
- Other AWS logs need some help to write to S3 but it's not hard (CloudWatch Logs + Kinesis)

Tip

- Always persist logs first before sending to a downstream analysis engine



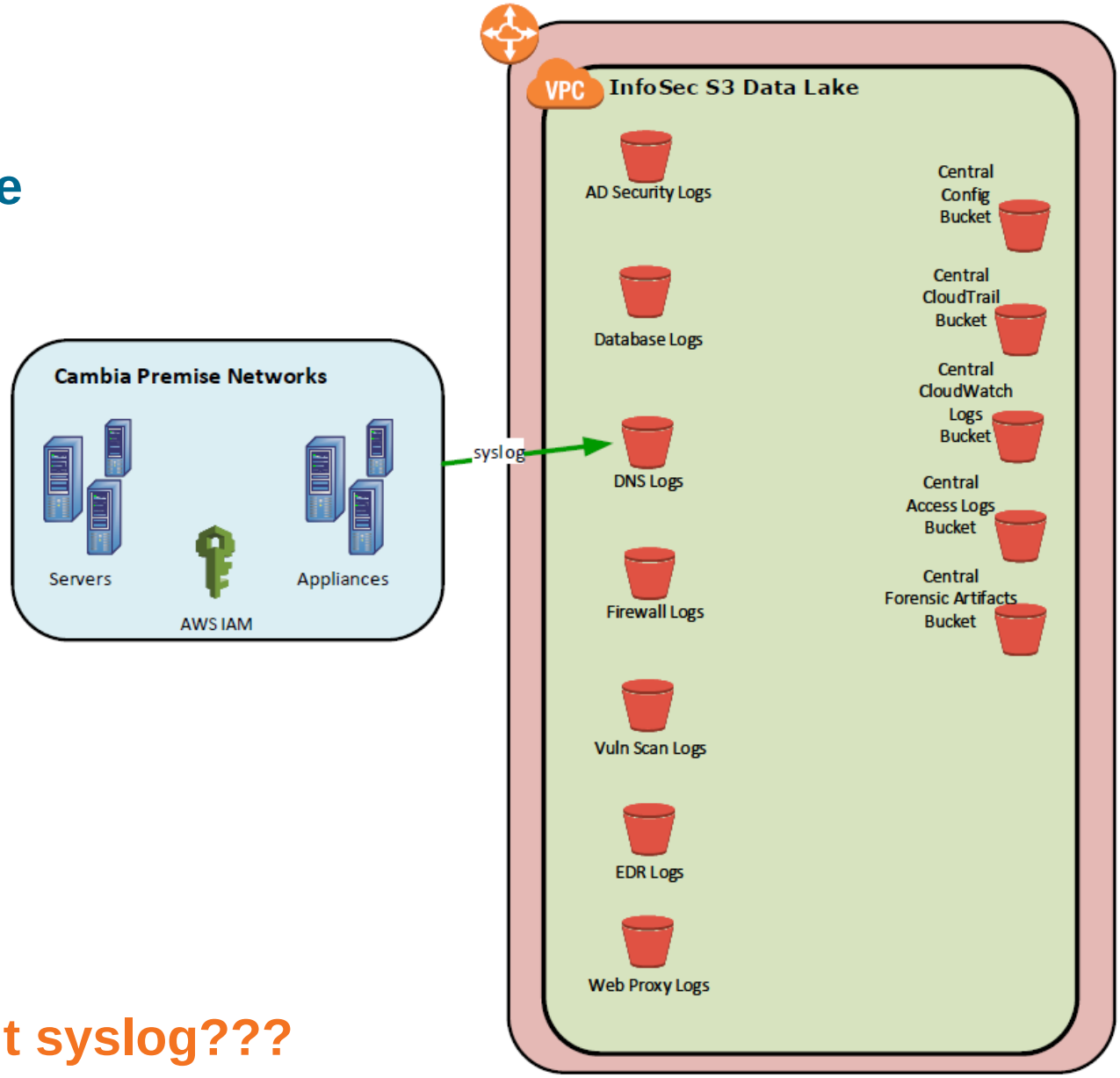
Data Lake for InfoSec

Send on-prem logs directly to Data Lake

- Logs can be uploaded directly to S3 with a bit of code and an AWS access key (cron job, scheduled task)

Tip

- **Always persist logs first** before sending to a downstream analysis engine



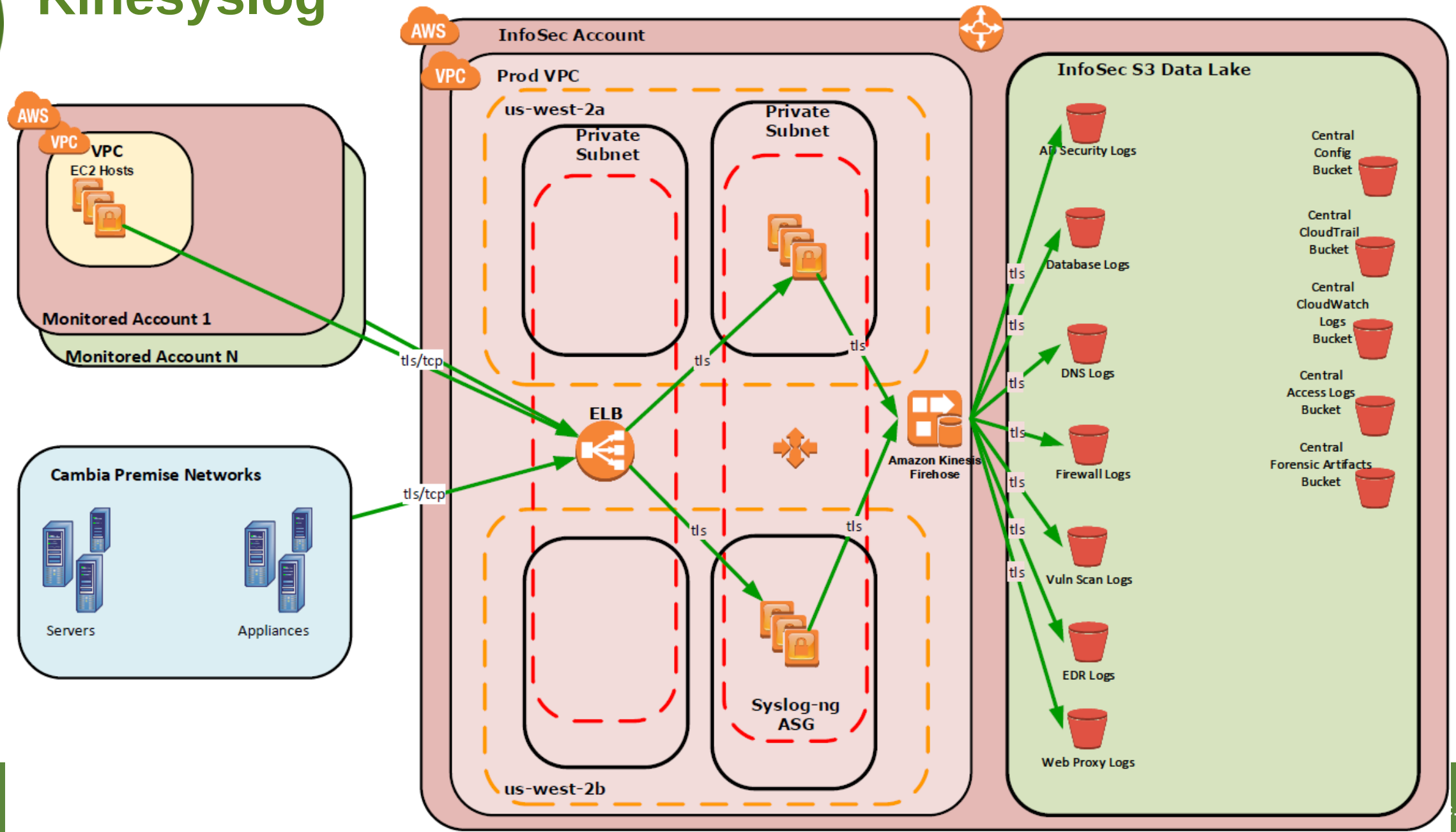
What about syslog???

What about Syslog? → Kinesyslog

We wrote a Syslog and Graylog Extended Log Format (GELF) relay to AWS Kinesis Firehose

- Supports UDP, TCP, and TLS; RFC2164, RFC5424, RFC5425, RFC6587, GELF v1.1
- Syslog events are received and stored in an in-memory buffer
- Every 4 MB or 60 seconds, message buffer is compressed and sent to Kinesis
- Kinesis Firehose writes the compressed JSON payload to S3
- Can run on-premise or on EC2

Kinesyslog


AWS
VPC
VPC
EC2 Hosts
Monitored Account 1
Monitored Account N
tls/tcp
Cambia Premise Networks
Servers
Appliances
tls/tcp
AWS
InfoSec Account
VPC
Prod VPC
us-west-2a
Private Subnet
Private Subnet
ELB
tls
tls
tls
tls
Amazon Kinesis Firehose
Syslog-ng ASG
us-west-2b
InfoSec S3 Data Lake
tls
tls
tls
tls
tls
tls
AD Security Logs
Database Logs
DNS Logs
Firewall Logs
Vuln Scan Logs
EDR Logs
Web Proxy Logs
Central Config Bucket
Central CloudTrail Bucket
Central CloudWatch Logs Bucket
Central Access Logs Bucket
Central Forensic Artifacts Bucket

A Good Start???

Persist logs

Ad hoc search with ELK

Hunt for IOCs

Run daily reports

Perform Forensics

Automate IOC detection

Near real-time alerting

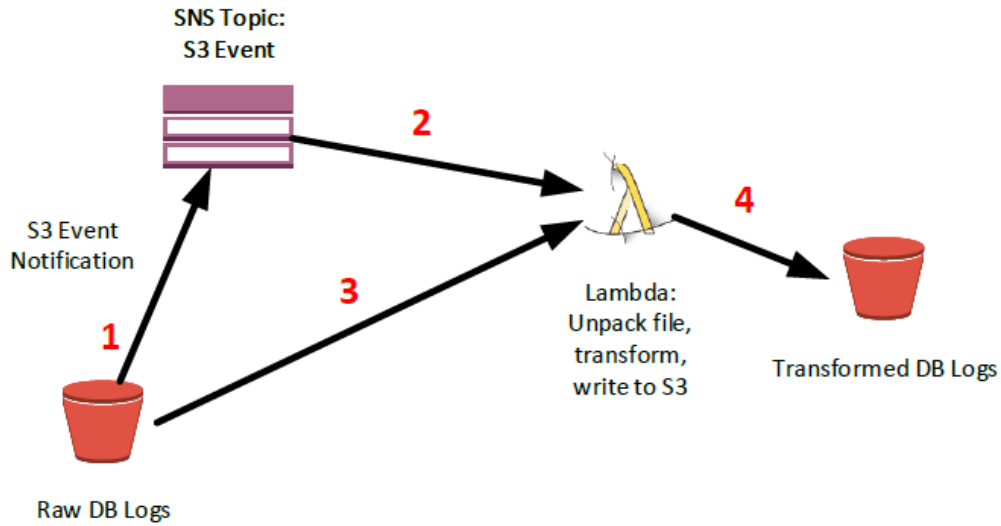
Train ML models

Generate security insights

What else can we do with the data?

If compressed JSON isn't your thing…transform to your desired format

1) Fire S3 event to AWS SNS topic
2) Invoke Lambda function when log file is written to S3
3) Decompress and transform to desired format
4) Write back to S3 bucket



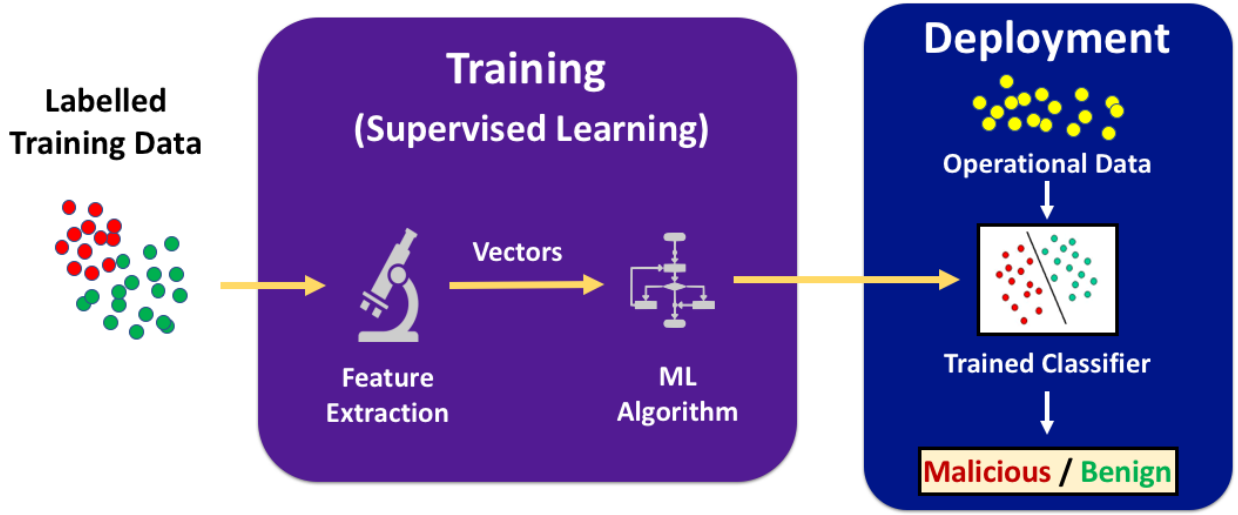
Put it in a database and query with SQL? Don't!

- Unless you need to query frequently, **logs in S3 are as good as logs in a database**
- Raw logs in S3 can be encrypted at rest, compressed, and still be queried with SQL
- Method is called "schema-on-read" → Table definition is applied to the data at query time
- Take a look at AWS Redshift Spectrum, Athena, and S3 Select

What about Machine Learning?

We looked at a few options

- Elastic X-pack for ML
- Splunk Analytics for ML
- Niddel Magnet (bought by Verizon)
- Cisco Stealthwatch Cloud



Two dominant patterns emerged

- Develop and train your own ML models (tried this first; it's hard)
  - Elastic X-pack & Splunk Analytics
- Feed data to somebody else's pre-developed ML models (easy)
  - Niddel Magnet & Cisco Stealthwatch

What about Machine Learning?

As we looked further at ML vendors we found something that surprised us…

The typical pattern

Agree on an evaluation period (**30 days**?)

Point log feeds at the solution and let it "**soak**" as product as it trains it's classifiers

After soak time has completed → Trained models generate **security insights**

Our Security Data lake pattern

- Grant vendor "cross-account" access to Data Lake S3 buckets →**12 months of data**
- Run **parallel ML "bakeoffs"** based on exactly the same historic and real-time events
- If a vendor is unable to use cross-account roles we **can always replay our logs** in to their API

Better than traditional SIEM?

- **Raw data persisted** without lossy or costly transformation
- Storage is **cheap and durable**
- Perform **ad hoc search** and **forensics** in ELK
- Transform logs into **any desired format in near real-time**, if needed
- **Never** have to **re-implement logging infrastructure** due to a new vendor
- **SQL Query** all security logs in S3 without loading to a database
- Quickly **evaluate promising ML technology** with years of real data

Questions and Comments