



# Scalable, Automated Anomaly Detection with Amazon GuardDuty and SageMaker

(Machine Learning for Security)

Jeff Puchalski  
Senior Security Engineer  
AWS Security

Neal Rothleder  
Senior Security Architect  
AWS Professional Services

# What you will do

- Download sample AWS CloudTrail logs and Amazon GuardDuty findings
- Walk through Amazon GuardDuty findings to understand key data elements
- Use Amazon SageMaker to identify users of AWS accounts coming from anomalous IP addresses.
- Fuse those findings with those coming from Amazon GuardDuty to create an aggregated list of suspicious activity.

# What you will do



# What we will cover

## AWS CloudTrail logging

- Review CloudTrail data feeds for format and content
- Load test CloudTrail data of good and bad activities

## Amazon GuardDuty findings

- Review GuardDuty, finding types, formats
- Hands-on lab - Use GuardDuty findings to create an aggregated list of suspicious activity

## Amazon SageMaker

- High-level architecture review - building and deploying a model
- Hands-on lab – Use IP Insights algorithm to detect anomalous IP usage for GuardDuty findings, using CloudTrail log data for model training

# AWS CloudTrail



## AWS CloudTrail

Track user activity and API usage

### What can you do?

- Simplify compliance audits and incident response by automatically recording and storing activity logs for your AWS account
- Logs API calls made to AWS services
  - 90-day event history on by default
- Can create log “trails” stored to S3
  - Optional KMS encryption
  - Optional log file integrity validation
- Optional data-level event logging for S3 object calls and Lambda invokes
- Can route events to CloudWatch Events

# Intro to AWS CloudTrail

AWS CloudTrail logs activity for supported services to an AWS account

Example API events that are logged:

- iam:CreateUser
- s3>ListBucket

CloudTrail is enabled by default on all AWS accounts (as of August 2017)

- Event history of create, modify, and delete operations for last 90 days
- Viewable, searchable, and downloadable
- To access CloudTrail log files directly or for a longer time period, create a trail
- You can also use Athena to query the logs directly in S3 buckets

# AWS CloudTrail log files

Gzipped JSON files created in the trail's Amazon S3 bucket

Each file contains API event records covering ~5 minute time window

Log files encrypted by default (SSE with S3 or KMS managed key)

Optional log file integrity validation using signed digest file containing log-file hashes

# AWS CloudTrail events

Each record in a CloudTrail log file represents a single event

All records contain some common fields:

- **Timestamp**
- **Region**
- **Event name** (i.e., the API call)
- **Event source** (i.e., the service)
- **Source IP address**
- **User identity**

Event-specific request and response parameters may also be included for some events

# What do CloudTrail logs look like?

```
{"Records": [{"eventVersion": "1.0", "userIdentity": {"type": "IAMUser", "principalId": "EX_PRINCIPAL_ID", "arn": "arn:aws:iam::123456789012:user/Alice", "accountId": "123456789012", "accessKeyId": "EXAMPLE_KEY_ID", "userName": "Alice"}, "eventTime": "2017-11-29T11:29:42Z", "eventSource": "iam.amazonaws.com", "eventName": "CreateUser", "awsRegion": "us-east-1", "sourceIPAddress": "192.168.0.1", "userAgent": "aws-cli/1.3.2 Python/2.7.5 Windows/7", "requestParameters": {"userName": "Bob"}, "responseElements": {"user": {"createDate": "Nov 29, 2017 11:29:42 AM", "userName": "Bob", "arn": "arn:aws:iam::123456789012:user/Bob", "path": "/", "userId": "EXAMPLEUSERID"}}], "]}}
```

# What do CloudTrail logs look like?

```
{"Records": [{"eventversion": "1.0", "userIdentity": {"type": "IAMUser", "principalId": "EX_PRINCIPAL_ID", "arn": "arn:aws:iam::123456789012:user/Alice", "accountId": "123456789012", "accessKeyId": "EXAMPLE_KEY_ID", "userName": "Alice"}, "eventTime": "2017-11-29T11:29:42Z", "eventSource": "iam.amazonaws.com", "eventName": "CreateUser", "awsRegion": "us-east-1", "sourceIPAddress": "192.168.0.1", "userAgent": "aws-cli/1.3.2 Python/2.7.5 Windows/7", "requestParameters": {"userName": "Bob"}, "responseElements": {"user": {"createDate": "Nov 29, 2017 11:29:42 AM", "userName": "Bob", "arn": "arn:aws:iam::123456789012:user/Bob", "path": "/", "userId": "EXAMPLEUSERID"}}, {"eventversion": "1.0", "userIdentity": {"type": "AWSIdentityProvider", "principalId": "EX_PRINCIPAL_ID", "arn": "arn:aws:sts::123456789012:assumed-role/test-role/test-user", "accountId": "123456789012", "accessKeyId": "EXAMPLE_KEY_ID", "userName": "test-user"}, "eventTime": "2017-11-29T11:29:42Z", "eventSource": "sts.amazonaws.com", "eventName": "AssumeRole", "awsRegion": "us-east-1", "sourceIPAddress": "192.168.0.1", "userAgent": "aws-cli/1.3.2 Python/2.7.5 Windows/7", "requestParameters": {"roleArn": "arn:aws:iam::123456789012:role/test-role", "durationSeconds": 3600}, "responseElements": {"tempCredentials": {"accessKeyId": "EXAMPLE_ACCESS_KEY_ID", "secretAccessKey": "EXAMPLE_SECRET_ACCESS_KEY", "sessionToken": "EXAMPLE_SESSION_TOKEN", "expiration": "2017-11-29T12:29:42Z", "accountId": "123456789012", "roleArn": "arn:aws:iam::123456789012:role/test-role", "sessionName": "test-session"}, "invokedRoleArn": "arn:aws:iam::123456789012:assumed-role/test-role/test-user"}}], "version": "1.0"}]
```

# What do CloudTrail logs look like?

```
{"Records": [{"eventVersion": "1.0", "userIdentity": {"type": "IAMUser", "principalId": "EX_PRINCIPAL_ID", "arn": "arn:aws:iam::123456789012:user/Alice", "accountId": "123456789012", "accessKeyId": "EXAMPLE_KEY_ID", "userName": "Alice"}, "eventTime": "2017-11-29T11:29:42Z", "eventSource": "iam.amazonaws.com", "eventName": "CreateUser", "awsRegion": "us-east-1", "sourceIPAddress": "192.168.0.1", "userAgent": "aws-cli/1.3.2 Python/2.7.5 windows/7", "requestParameters": {"userName": "Bob"}, "responseElements": {"user": {"createDate": "Nov 29, 2017 11:29:42 AM", "userName": "Bob", "arn": "arn:aws:iam::123456789012:user/Bob", "path": "/", "userId": "EXAMPLEUSERID"}}], "]}]
```

# Amazon GuardDuty



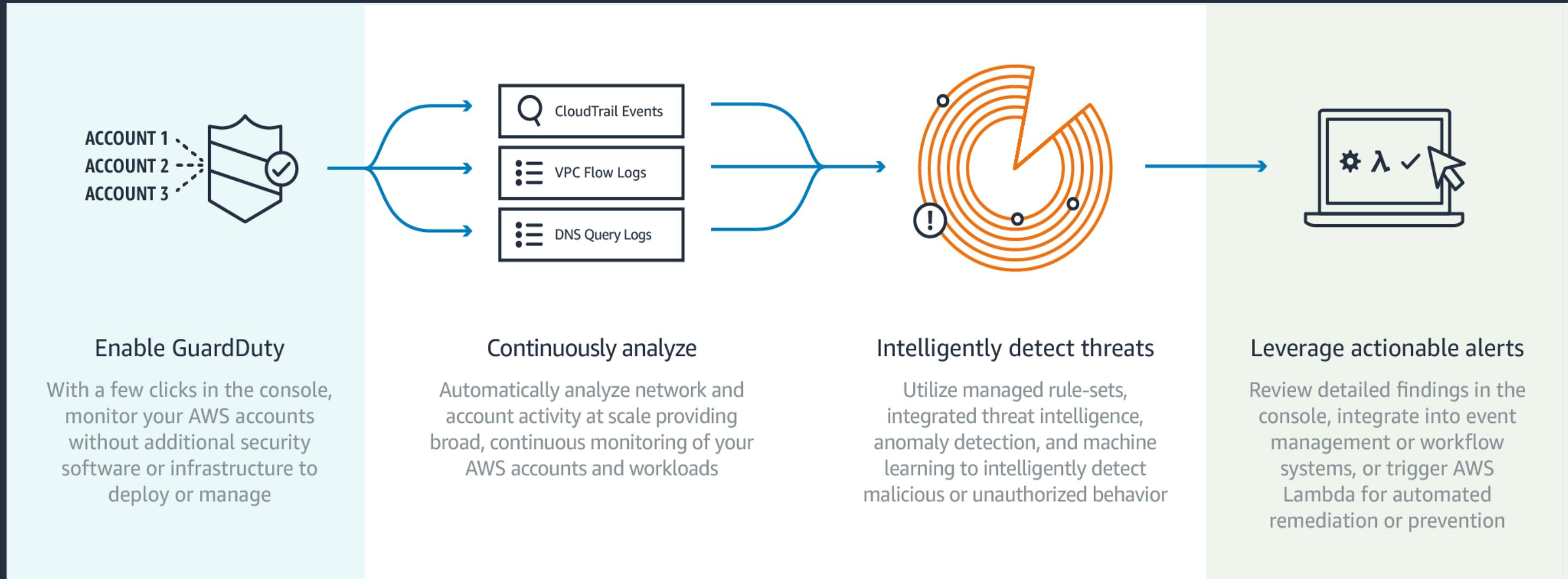
## Amazon GuardDuty

Intelligent threat detection  
and continuous monitoring to  
protect your AWS accounts  
and workloads

### What can you do?

- Continuous monitoring to rapidly detect threats (needle) to your environments in the sea of log data (haystack)
- Processes AWS CloudTrail logs, Amazon VPC flow logs, and DNS logs
- Analyzes billions of events across your AWS accounts for signs of risk
- Identifies unexpected and suspicious activity, such as privilege escalation, exposed creds, and communication with malicious IPs
- Can send findings to CloudWatch Events

# GuardDuty Threat Detection and Notification



# GuardDuty Data Sources

## VPC Flow Logs



- Flow Logs for VPCs do *not* need to Be Turned on to generate findings Data is consumed through independent duplicate stream.
- Suggested turning on VPC Flow Logs to augment data analysis (charges apply).

## DNS Logs



- DNS Logs are based on queries made from EC2 instances to known questionable domains.
- DNS Logs are in addition to Route 53 query logs. Route 53 is not required for GuardDuty to generate DNS based findings.

## CloudTrail Logs



- CloudTrail history of AWS API calls used to access the Management Console, SDKs , CLI, etc. presented by GuardDuty.
- Identification of user and account activity including source IP address used to make the calls.

# Trusted IP and Threat IP Lists

GuardDuty uses AWS developed threat intelligence and threat intelligence feeds from: CrowdStrike & Proofpoint

Expand Findings with Custom Trusted IP Lists and Known Threat Lists

- Trusted IP lists whitelisted for secure communication with infrastructure and applications
- No Findings will be presented for IP Addresses on trusted lists (no false positives!)
- Threat lists consist of known malicious IP addresses.
- GuardDuty generates findings based on threat lists.

Limits: 1 Trusted and 6 Threat Lists per Account

TRUSTED IP LISTS



KNOWN THREATS

# GuardDuty Findings

Describes threats by their primary purpose:

**ThreatPurpose:ResourceTypeAffected/ThreatFamilyName.ThreatFamilyVariant!Artifact**

**Backdoor:** resource compromised and capable of contacting source home

**Behavior:** activity that differs from established baseline

**Crypto Currency:** detected software associated with Crypto currencies

**Pentest:** activity detected similar to that generated by known pen testing tools

**Recon:** attack scoping vulnerabilities by probing ports, listening, database tables, etc.

**Stealth:** attack trying to hide actions / tracks

**Trojan:** program detected carrying out suspicious activity

**Unauthorized Access:** suspicious activity / pattern by unauthorized user

# GuardDuty Findings in Console

**UnauthorizedAccess:EC2/TorIPCaller** [QQ](#)

 EC2 instance i-99999999 is communicating with IP address 198.51.100.0 on the Tor Anonymizing Proxy network. [🔗](#)

Severity	Region	Count
Medium <a href="#">QQ</a>	ca-central-1	1
<b>Account ID</b>	<b>Resource ID</b>	
254599117341 <a href="#">QQ</a>	i-99999999	

**Last seen**  
2018-03-13 08:04:16 (a few seconds ago)

**Resource affected**

Resource role	Resource type
TARGET	Instance <a href="#">QQ</a>
<b>Instance ID</b>	<b>Port</b>
i-99999999 <a href="#">QQ</a>	80 <a href="#">QQ</a>
<b>Image ID</b>	<b>Image description</b>
ami-99999999	GeneratedFindingInstaceImageDescription
<b>Launch time</b>	
2016-08-01 19:05:06	
<b>Instance profile</b>	
Arn: GeneratedFindingInstanceProfileArn	
ID: GeneratedFindingInstanceProfileId	

# GuardDuty Findings in Console

```
{  
  "version": "0",  
  "id": "c8c4daa7-a20c-2f03-0070-b7393dd542ad",  
  "detail-type": "GuardDuty Finding",  
  "source": "aws.guardduty",  
  "account": "254599117341",  
  "time": "2018-03-13T08:04:33Z",  
  "region": "ca-central-1",  
  "resources": [],  
  "detail": {  
    "schemaVersion": "2.0",  
    "accountId": "254599117341",  
    "region": "ca-central-1",  
    "partition": "aws",  
    "id": "16afba5c5c43e07c9e3e5e2e544e95df",  
    "arn": "arn:aws:guardduty:us-east-1:254599117341:detector/254599117341/finding/16afba5c5c43e07c9e3e5e2e544e95df",  
    "type": "UnauthorizedAccess:EC2/TorIPCaller",  
    "resource": { ... },  
    "service": { ... },  
    "severity": 3,  
    "createdAt": "2018-03-13T08:04:16.000Z",  
    "updatedAt": "2018-03-13T08:04:16.000Z",  
    "title": "UnauthorizedAccess:EC2/TorIPCaller",  
    "description": "UnauthorizedAccess:EC2/TorIPCaller"  
  }  
}
```

# GuardDuty Findings for this workshop

**Unauthorized Access:** suspicious activity / pattern by unauthorized user

- UnauthorizedAccess:IAMUser/ConsoleLogin
- UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B
- UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration
- UnauthorizedAccess:IAMUser/MaliciousIPCaller
- UnauthorizedAccess:IAMUser/UnusualASNCaller

# GuardDuty Findings for this Workshop

Threat UnauthorizedAccess:IAMUser/ConsoleLogin

- This IAM user has no prior history of login activity
- Your IAM user credentials might be compromised

Severity

- Medium
- Unknown confidence

# Strategy

# GuardDuty + Additional Machine Learning

GuardDuty severity scores are assigned by the type of finding  
They do not factor in context beyond their finding conditions

We can add context and start to build a more holistic confidence score by using a  
ML approach we can assign

*Let's combine the UnauthorizedAccess GuardDuty finding with an additional estimate of  
the likelihood of seeing this [User + SourceIP] to further evaluate how unusual the activity  
is.*

# Amazon SageMaker



## Amazon SageMaker

Build, train, and deploy  
machine learning models at  
scale

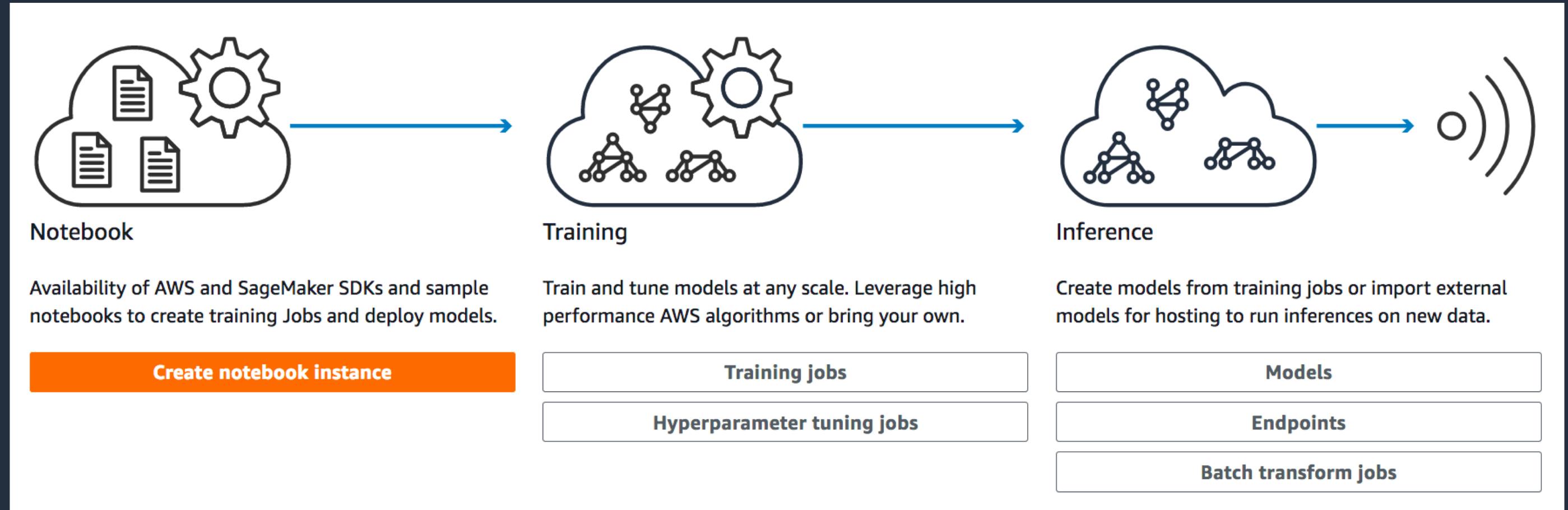
### What can you do?

- Rapidly build, train, and deploy ML models
  - Automated model training and tuning
- Supports all algorithms and frameworks
  - Many built-in optimized algorithms
  - Includes MXNet, TensorFlow, and others
- Experiment using hosted interactive notebooks
  - Jupyter notebooks that support multiple languages (e.g., Python, Scala)
- Built-in A/B testing capabilities

# The Amazon Machine Learning Stack



# SageMaker Workflow



# SageMaker Dashboard

Amazon SageMaker X

- Dashboard
- Search Beta
- Notebook**
  - Notebook instances**
  - Lifecycle configurations
  - Git repositories
- Training**
  - Algorithms
  - Training jobs
  - Hyperparameter tuning jobs
- Inference**
  - Model packages
  - Models
  - Endpoint configurations
  - Endpoints
  - Batch transform jobs

Amazon SageMaker > Notebook instances

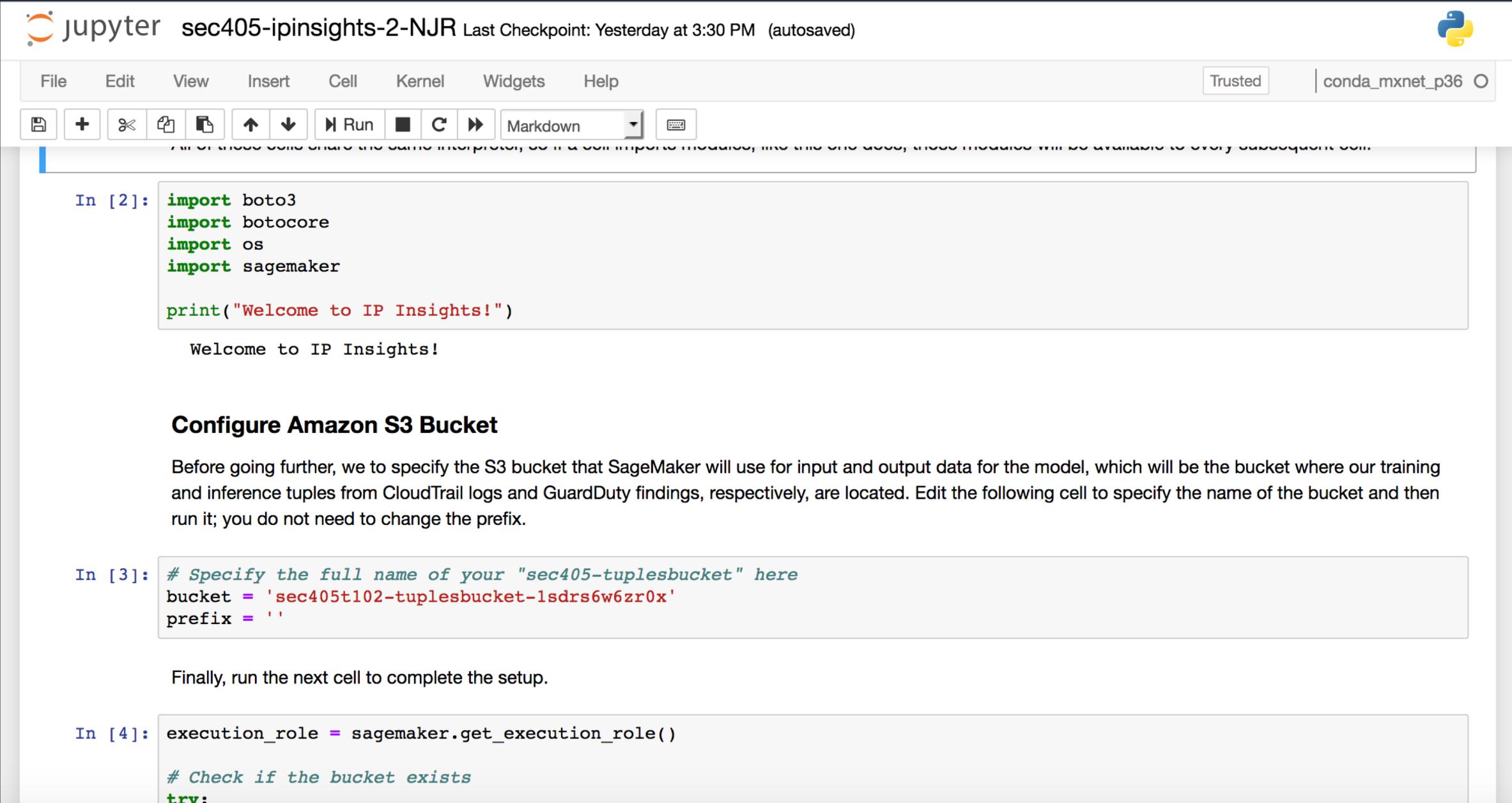
### Notebook instances

Actions Create notebook instance

Search notebook instances

Name	Instance	Creation time	Status	Actions
SEC405T102	ml.t2.medium	Jan 31, 2019 23:14 UTC	<span style="color: green;">InService</span>	<a href="#">Open Jupyter</a>   <a href="#">Open JupyterLab</a>

# SageMaker (Jupyter) Notebook



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter sec405-ipinsights-2-NJR Last Checkpoint: Yesterday at 3:30 PM (autosaved)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, conda\_mxnet\_p36
- Cell 2:** In [2]:

```
import boto3
import botocore
import os
import sagemaker

print("Welcome to IP Insights!")
```

Welcome to IP Insights!
- Section Header:** **Configure Amazon S3 Bucket**
- Text:** Before going further, we need to specify the S3 bucket that SageMaker will use for input and output data for the model, which will be the bucket where our training and inference tuples from CloudTrail logs and GuardDuty findings, respectively, are located. Edit the following cell to specify the name of the bucket and then run it; you do not need to change the prefix.
- Cell 3:** In [3]:

```
# Specify the full name of your "sec405-tuplesbucket" here
bucket = 'sec405t102-tuplesbucket-1sdrs6w6zr0x'
prefix = ''
```
- Text:** Finally, run the next cell to complete the setup.
- Cell 4:** In [4]:

```
execution_role = sagemaker.get_execution_role()

# Check if the bucket exists
try:
```

# ML Model

# IP Insights Model

Learn a measure of association between users and IP addresses using legitimate activity to answer the question of “How normal it is to see a given user using a given IP”

Statistical modeling and neural networks to capture associations

If the vector <principal ID, IP address> are close together, then it is likely for the principal to access the account from that IP address, even if it has never accessed it before

# Solution

# Exercise Workflow

- Using our “normal” CloudTrail data (User, IP address pairs), train SageMaker – using the IP Insight ML algorithm - to build a baseline model of normal IP usage
- As GuardDuty alerts come in, feed the alerting <User, IP> pairs to our model to determine a fit score. Low scores → Anomalous
- *[Blend anomaly score with findings from Amazon GuardDuty to create an aggregated list of suspicious activity] – See the IPInsights tutorial*

# Building Blocks



AWS CloudTrail

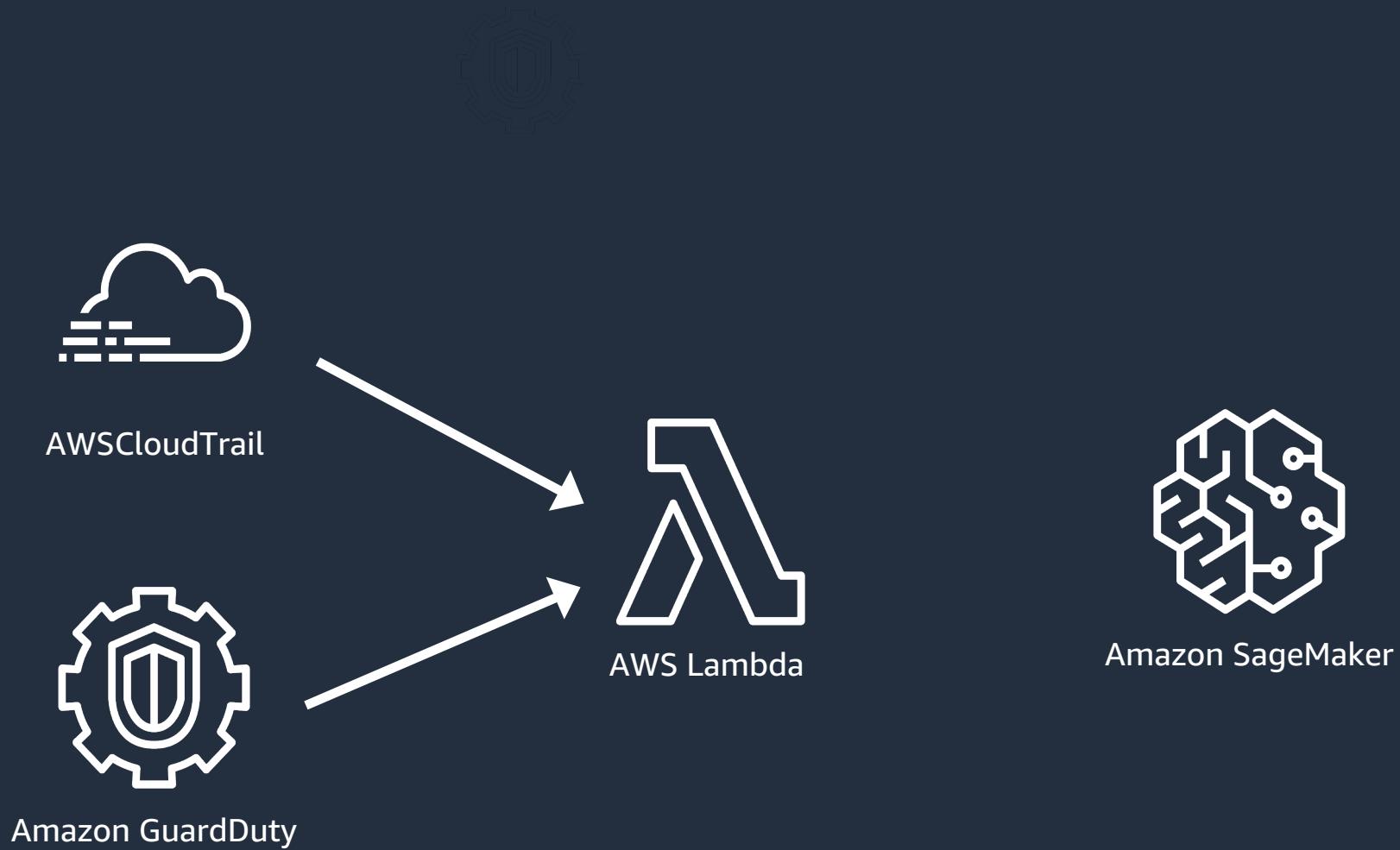


AWS Lambda

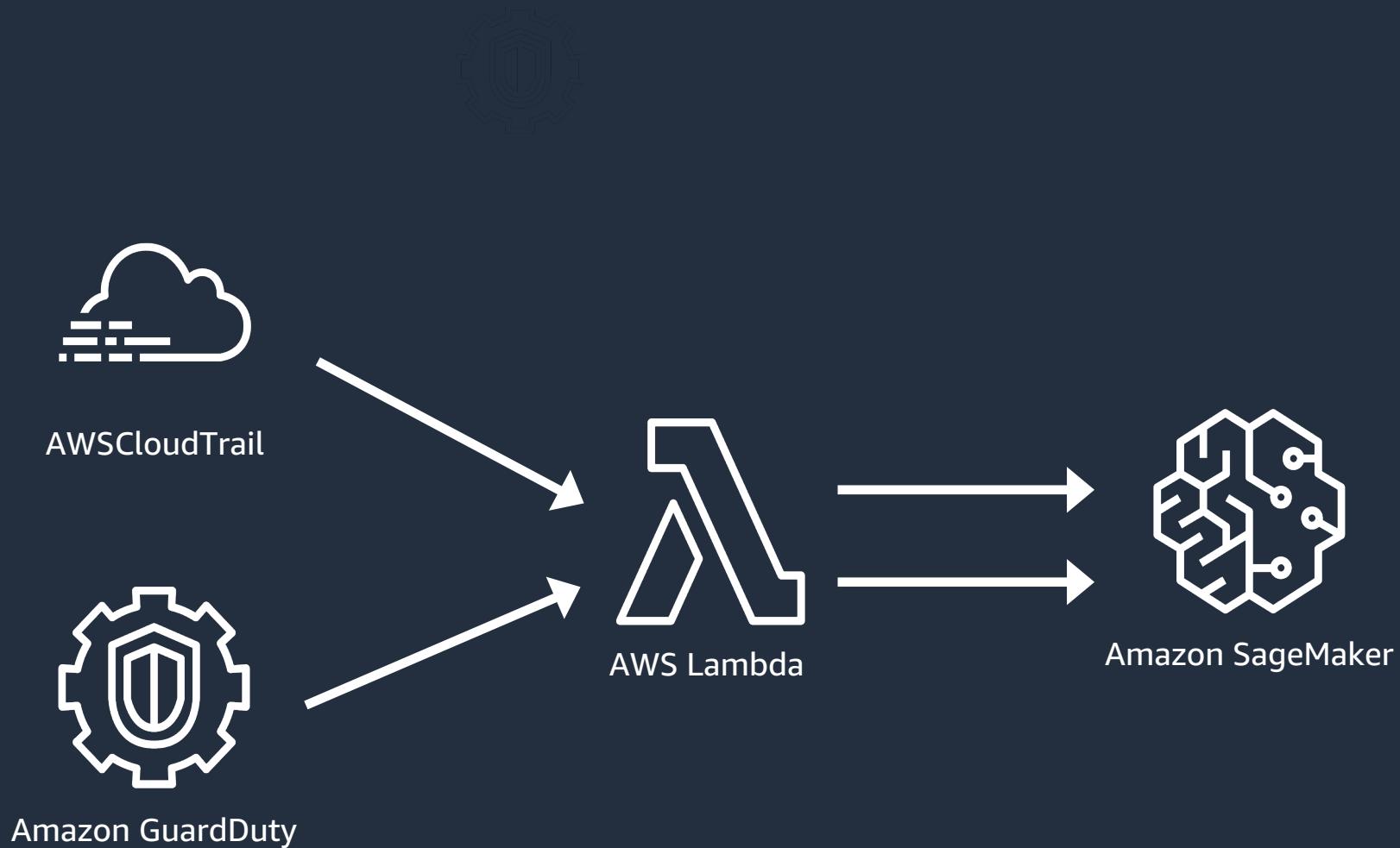


Amazon SageMaker

# Building Blocks



# Building Blocks



# Exercise Steps

Browse to <https://awssecworkshops.com/getting-started/>

Work through the steps

- If you already have an account you can skip the first step
- Skip “Create a Cloud9 instance” for this workshop

Next, open the menu and browse to:

Workshops -> Detection with Machine Learning

URL: <https://ml-threat-detection.awssecworkshops.com/>

# Putting This Into Practice

- Use this as-is: score specific console logins tagged by GuardDuty
- Tuning
  - Parameters and training sets
  - When to retrain the IP Insights model?
- Use IP Insights on different classes of behavior
  - Specific application usage (e.g., monitoring apps, bastion hosts)
- Thinking bigger – Adding additional detectors and models
  - Leverage GuardDuty, Macie findings
  - Build your own
  - Leverage broad classes of algorithms and prior work
  - Build application-specific models
- Remember the importance of Subject Matter Expertise

# Thank you!

<https://github.com/aws-samples/aws-security-workshops/detection-ml-workshop/>

Jeff: [jski@amazon.com](mailto:jski@amazon.com) (AppSec Team)

Neal: [nrothled@amazon.com](mailto:nrothled@amazon.com) (ProServe Team)

Baris: [barisco@amazon.com](mailto:barisco@amazon.com) (GuardDuty Team)

Rima: [tanashr@amazon.com](mailto:tanashr@amazon.com) (Security Hub Team)