AWS re:Invent

SEC405

# Scalable, Automated Anomaly Detection with Amazon GuardDuty and Amazon SageMaker

Jeff Puchalski
Senior Security Engineer
AWS Security

Neal Rothleder
Senior Security Architect
AWS Professional Services

AWS re:Invent

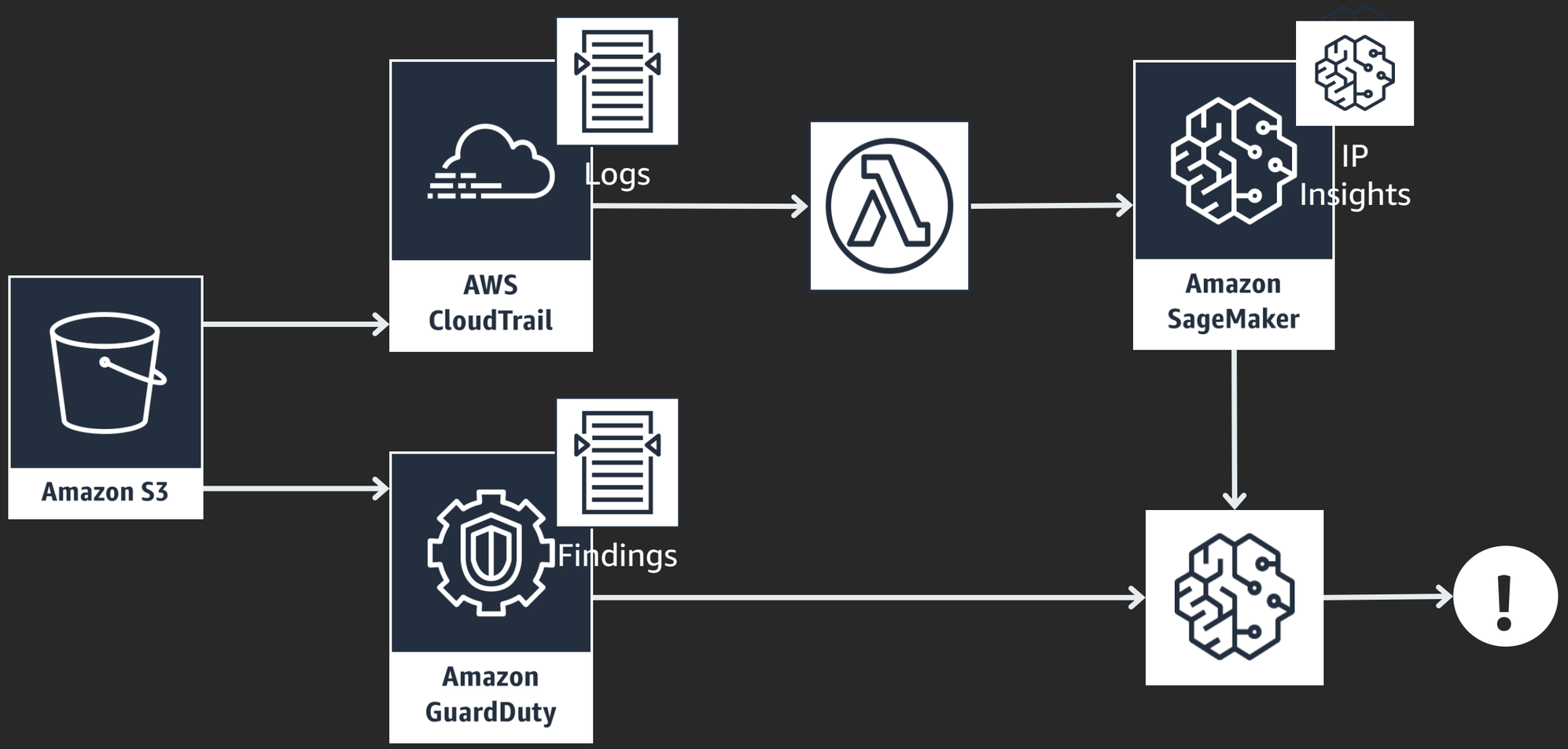


aws

# 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 re:Invent



aws

# 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 Amazon S3
  - Optional AWS KMS encryption
  - Optional log file integrity validation
- Optional data-level event logging for Amazon S3 object calls and Lambda invokes
- Can route events to Amazon CloudWatch Events

# 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"}
    }
}]}
```

# 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

AWS re:Invent



aws

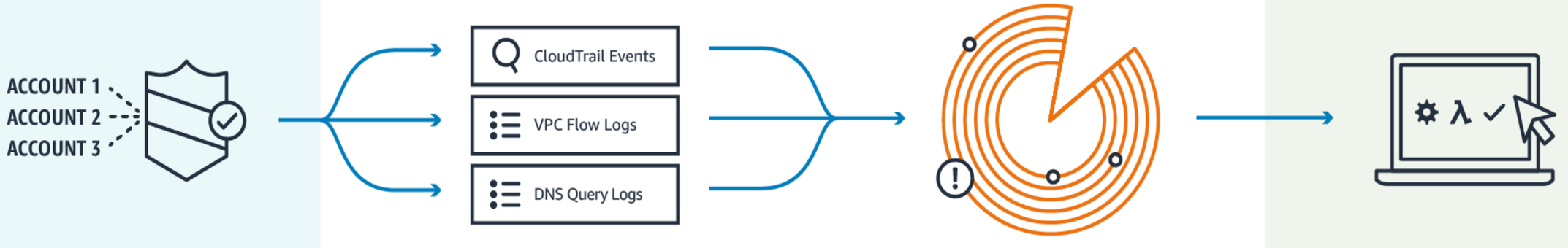
# Amazon GuardDuty

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




### Enable GuardDuty

With a few clicks in the console, monitor your AWS accounts without additional security software or infrastructure to deploy or manage

### Continuously analyze

Automatically analyze network and account activity at scale providing broad, continuous monitoring of your AWS accounts and workloads

### Intelligently detect threats

Utilize managed rule-sets, integrated threat intelligence, anomaly detection, and machine learning to intelligently detect malicious or unauthorized behavior

### Leverage actionable alerts

Review detailed findings in the console, integrate into event management or workflow systems, or trigger AWS Lambda for automated remediation or prevention

# 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 Amazon 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 AWS Management Console, SDKs, AWS 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

# 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

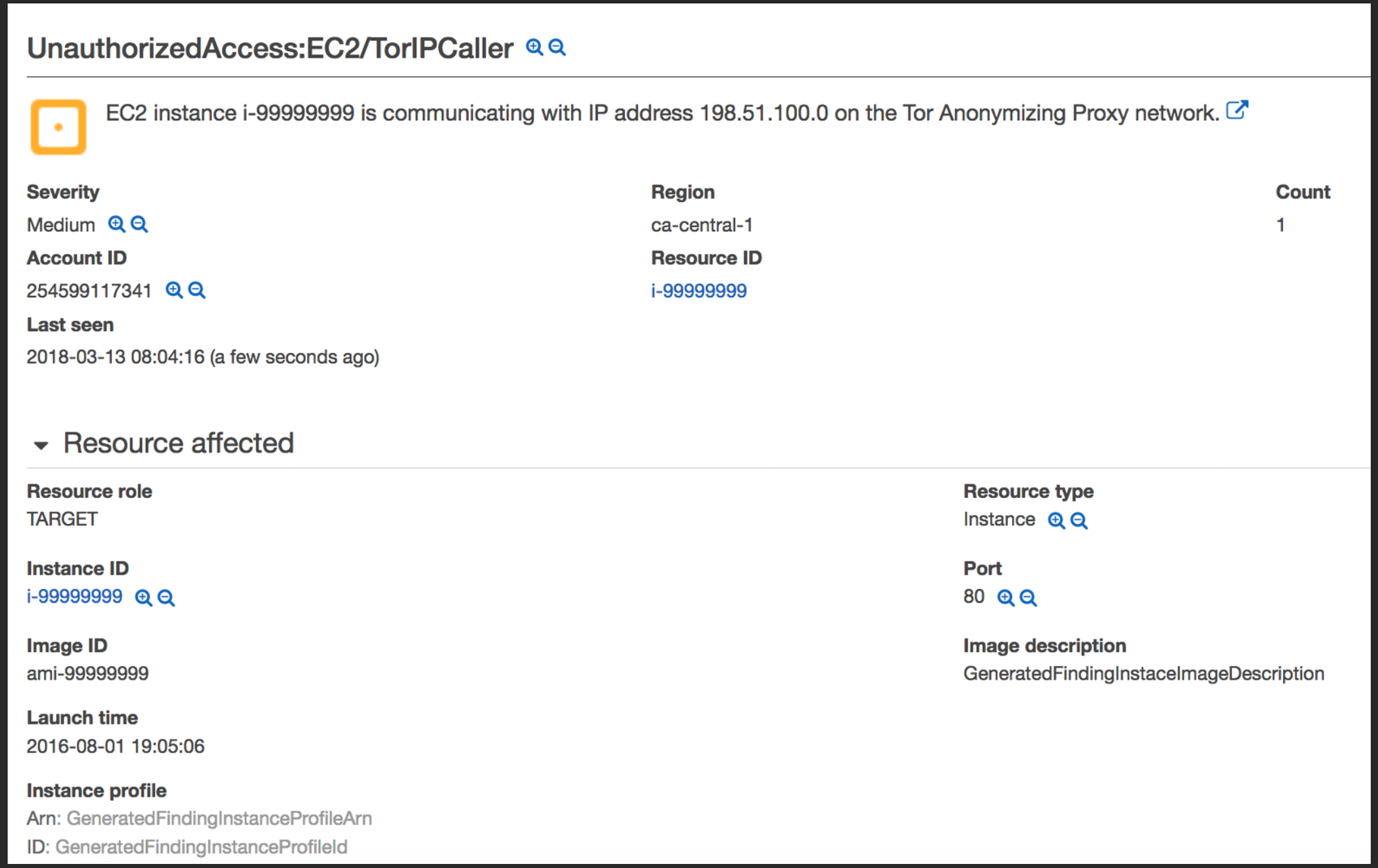
Recon: attack scoping vulnerabilities by probing ports, listening, database tables, etc.

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



# 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

AWS re:Invent



aws

# GuardDuty + Additional Machine Learning

GuardDuty assigns all findings of the same type with the same severity

If we can add some additional information, we can start to build a more context sensitive confidence score to individual findings

*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

AWS re:Invent



aws

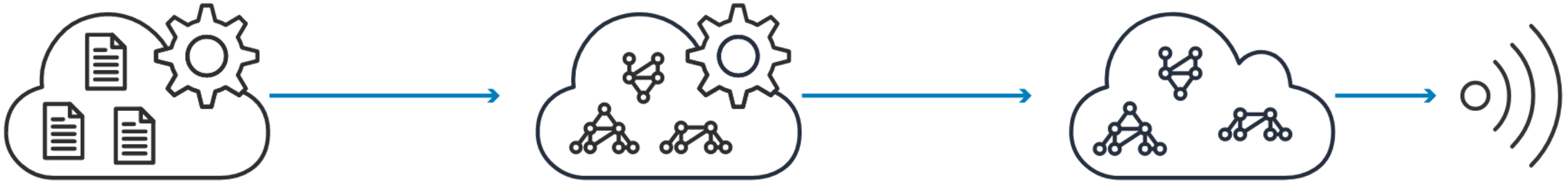
# Amazon SageMaker

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

# Amazon SageMaker Workflow



## Notebook

Availability of AWS and SageMaker SDKs and sample notebooks to create training Jobs and deploy models.

**Create notebook instance**

## Training

Train and tune models at any scale. Leverage high performance AWS algorithms or bring your own.

**Training jobs**

**Hyperparameter tuning jobs**

## Inference

Create models from training jobs or import external models for hosting to run inferences on new data.

**Models**

**Endpoints**

**Batch transform jobs**

# ML Model

AWS re:Invent



aws

# IP Insights Model

Learn a measure of association between principals 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

Use IP Insight ML model to augment threat detection in your AWS environment

Use Amazon SageMaker to identify users of your AWS accounts coming from anomalous IP addresses

Blend anomaly score with findings from Amazon GuardDuty to create an aggregated list of suspicious activity

# Building Blocks

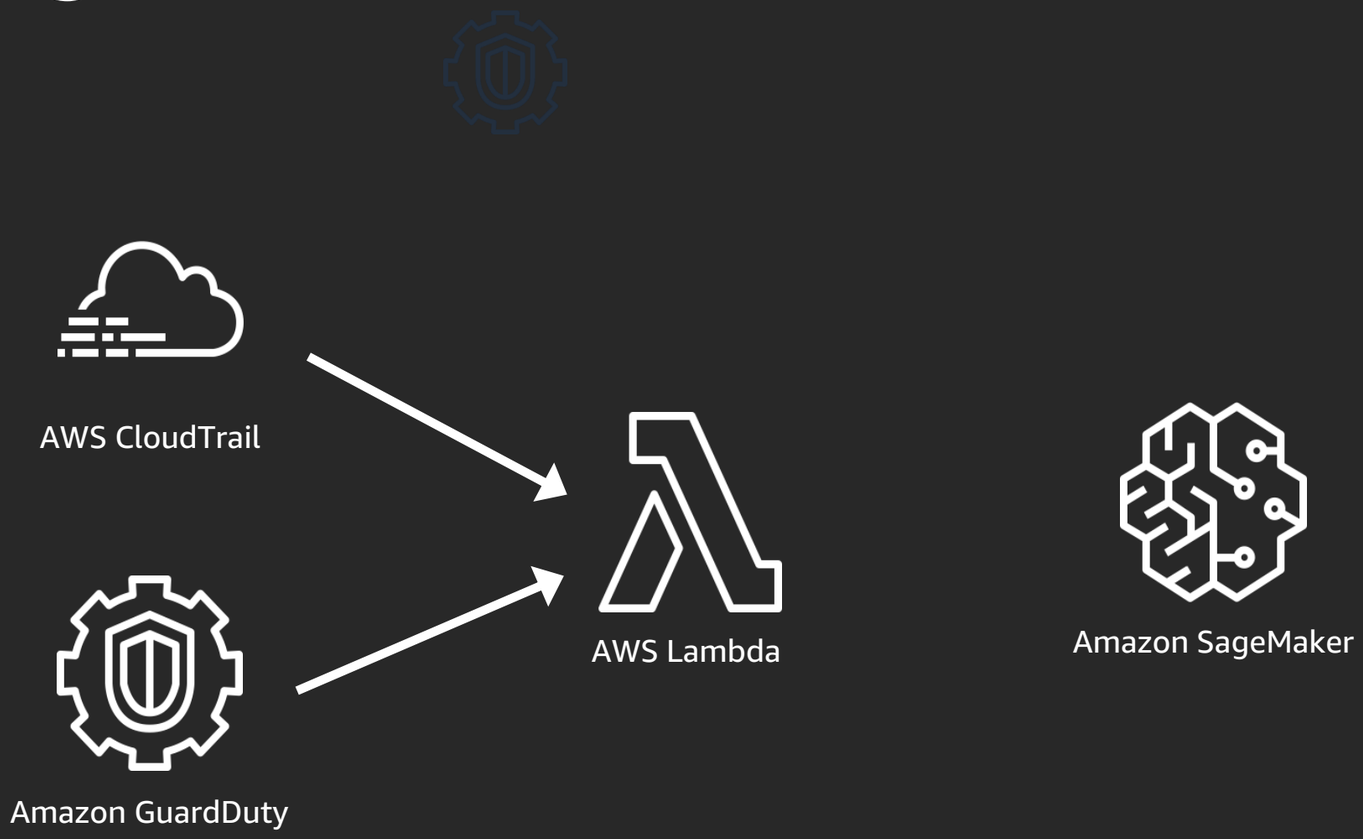
AWS CloudTrail

Amazon GuardDuty

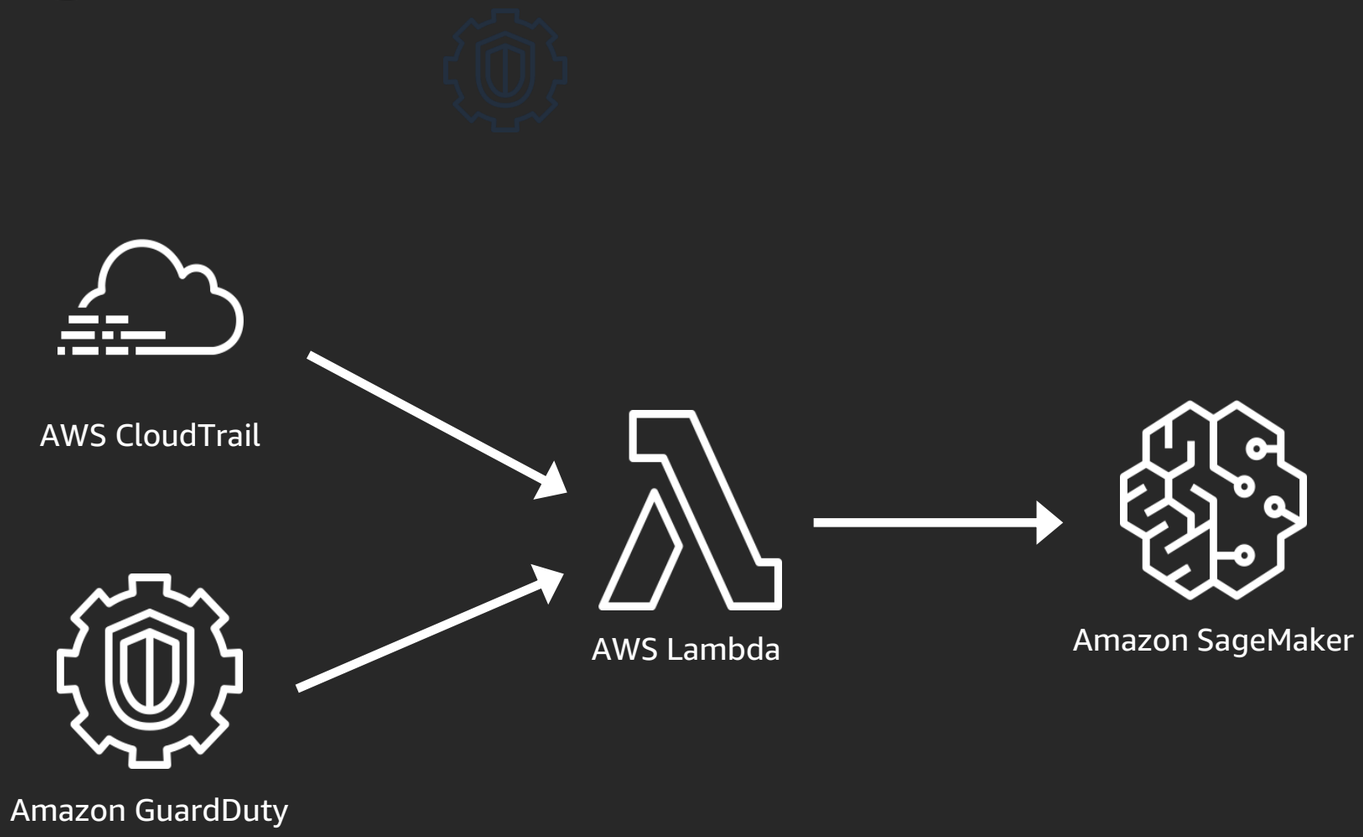
AWS Lambda

Amazon SageMaker

# Building Blocks



# Building Blocks



# Exercise Steps

## Browse to the workshop repo: https://amzn.to/sec405

- Repository for exercise hosted on GitHub
- Instructions are contained in the README file in the repo
- There is also an interactive notebook for the IP Insights algorithm that you will use in SageMaker

# 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 (AppSec Team)

Neal: nrothled@amazon.com (ProServe Team)
Rima: tanashr@amazon.com (ProServe Team)

Baris: barisco@amazon.com (GuardDuty Team)

Please complete the session survey in the mobile app.

AWS re:Invent



aws